Я.К.Трохименко, Ф.Д.Любич

ИНЖЕНЕРНЫЕ РАСЧЕТЫ НА ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРАХ

Библиотека инженера

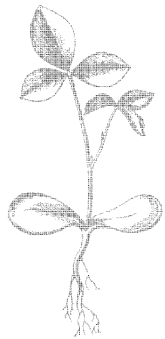
Библиотека инженера

Я.К.Трохименко, Ф.Д.Любич

Библиотека
инженера

Я.К.Трохименко, Ф.Д.Любич

ИНЖЕНЕРНЫЕ РАСЧЕТЫ НА ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРАХ



Киев «Техніка»
1985

32.973.1
Т76

Трохименко Я. К., Любич Ф. Д.

Т76 Инженерные расчеты на программируемых микрокалькуляторах.— К.: Техніка, 1985.— 328 с., ил.— (Б-ка инженера).— Библиогр.: с. 326.

В пер. тип 5 — 1 р. 30 к., в пер. тип 7 — 1 р. 40 к. 30 000 экз.

Рассмотрены особенности вычислений на микрокалькуляторах, погрешности результатов вычислений, составление, отладка и перевод программ на входных языках отечественных программируемых микрокалькуляторов. Описаны методика составления алгоритмов и программ для решения основных задач, связанных с инженерным проектированием: вычислениями по аналитическим выражениям, решением уравнений и систем уравнений, численным интегрированием, статистической обработкой результатов эксперимента, численной оптимизацией.
Рассчитана на широкий круг инженерно-технических работников.

Т $\frac{2405000000\text{-}019}{\text{М202(04)-85}}$ 50.85 32.973.1

Рецензенты:
д-р техн. наук Л. Я. Нагорный, Ю. М. Польский

Редакция литературы по энергетике,
электронике, кибернетике и связи
Зав. редакцией З. В. Божко

ПРЕДИСЛОВИЕ

Малогабаритные микро-ЭВМ, называемые микрокалькуляторами, успешно используются для решения повседневных инженерных задач. Доступность, определяемая портативностью, малой стоимостью машинного времени и простотой пользования, обеспечивает широкое применение микрокалькуляторов в качестве достаточно мощного персонального вычислительного средства. Особенно большие возможности связаны с использованием программируемых микрокалькуляторов, из которых многие по ряду показателей не уступают универсальным ЭВМ первых поколений. Однако эффективное применение этих микро-ЭВМ требует знания численных методов, теории погрешностей и особенностей программирования, что обусловливает необходимость в пособиях для широкого круга пользователей программируемых микрокалькуляторов.

Первая отечественная книга по применению микрокалькуляторов для инженерных расчетов [11] вышла в свет, когда промышленность выпускала лишь первый массовый программируемый микрокалькулятор типа «Электроника Б3-21», а его пользователи только начинали осваивать программирование решения сложных задач. В настоящее время ассортимент программируемых микрокалькуляторов существенно расширился, а их вычислительные возможности возросли.

Существенную помощь при инженерных расчетах оказывают пособия и справочники [13, 19], содержащие программы решения типовых задач. Однако самые обширные сборники программ не могут охватить многообразия задач, встречающихся в инженерной практике, а для эффективного использования таких программ необходимо знать методику их составления и уметь оценивать точность результатов вычислений. Поэтому по мере совершенствования микрокалькуляторов, усложнения их входных языков и соответствующего расширения вычислительных возможностей издание пособий по применению программируемых микрокалькуляторов становится еще более актуальным.

Настоящая книга, являясь практическим пособием по программированию решения инженерных задач, коренным образом отличается от книги [11] с близким названием не только текстом, включая программы, но и особенностями изложения. Основное внимание в настоящей книге уделено методике составления оптимальных программ решения типовых математических задач, к которым сводятся инженерные расчеты, с максимальным использованием вычислительных возможностей программируемых микрокалькуляторов. Объем книги ограничил возможность подробного рассмотрения методики составления алгоритмов и их последующей реализации на входных языках микро-

калькуляторов для всех приведенных программ. Однако приведенные примеры помогут читателю самостоятельно составлять эффективные программы решения инженерных задач как с помощью современных программируемых микрокалькуляторов, так и, с учетом непрерывного развития вычислительной техники, с помощью новых персональных вычислительных средств.

Отзывы и пожелания о книге просим направлять по адресу:
252601, Киев, 1, Крещатик, 5, издательство «Техніка».

Глава 1

ОСОБЕННОСТИ РАБОТЫ МИКРОКАЛЬКУЛЯТОРОВ

1. АЛГОРИТМЫ И ПРОГРАММЫ

Процесс решения новой задачи после ее возникновения (постановки) состоит из следующих основных этапов: 1) формализации условий задачи (как связаны по условиям задачи исходные данные и требуемый результат ее решения?); 2) выбора метода решения (как подойти к решению задачи?); 3) выбора способа решения (какие действия нужно выполнить?); 4) выполнения необходимых действий; 5) проверки приемлемости решения (соответствует ли полученный результат требуемому?).

При решении простых задач человек мысленно выполняет первые три этапа, выбирая наилучший способ решения на основании жизненного опыта и логических умозаключений. При решении сложных задач ему обычно приходится предварительно теоретически отыскивать наилучший путь решения задачи и лишь после тщательной проверки полученного результата приступать к практическому решению задачи.

Теоретическое решение задачи часто оказывается весьма трудоемким (например, при научных исследованиях или инженерном проектировании) и практически всегда связано с использованием математических моделей*. Основная задача инженерного проектирования заключается в составлении с помощью математического моделирования и в соответствии с требованиями, поставленными в техническом задании, теоретической модели (проекта) еще не существующего в природе объекта или способа (технологии) его изготовления. На математическом моделировании свойств компонентов проектируемых объектов и происходящих в них процессов основано решение и частных задач инженерного расчета. Преобразование математических моделей является основным содержанием математики, что обеспечивает сведение теоретического решения инженерных задач к последовательному решению типовых математических задач.

Этап *формализации* условий задачи заключается в составлении исходной математической модели (например, в виде системы уравнений или чертежа), отображающей с достаточной точностью связи между физическими (реально существующими) условиями задачи и требуемым результатом ее решения.

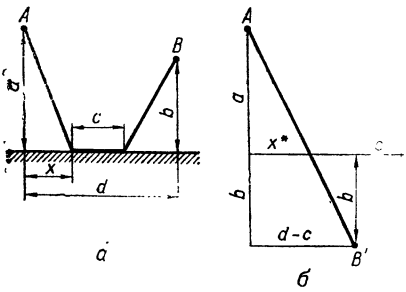
Выбор метода (подхода к решению определенного класса задач) сводится к выбору преобразования исходной математической модели, обеспечивающего выражение искомого результата через исходные данные. Если исходные данные и отношения между ними и результатом

---

* *Математическая модель* — совокупность алгебраических или математических знаков, символизирующих числа или пространственные формы и отношения между ними, которые отображают свойства реальных объектов.

выражены при помощи алгебраических или геометрических символов, то метод называют аналитическим или символьным. Такой метод используют для поиска заданных отношений между исходными объектами, а также для определения зависимостей результата от исходных данных и составления расчетных формул, по которым вычисляют искомый результат.

В тех случаях, когда для определения результата нельзя использовать аналитическое выражение, применяют численные методы. Прямые (точные) численные методы основаны на выполнении заранее точно известного количества действий (операций), требуемого для вычисления результата. При использовании методов последовательных приближений количество операций заранее неизвестно, а вычисления прекращают после получения результата с заданной точностью.



а б

Рис. 1

Удачный выбор метода может обеспечить существенное уменьшение трудоемкости решения задачи. В качестве наглядного примера рассмотрим задачу выбора кратчайшей трассы между пунктами A и B, расположенными на расстояниях a и b от прямолинейного участка берега моря и на расстоянии d вдоль берега, при условии, что отрезок трассы длиной c должен проходить вдоль берега.

По математической модели условий задачи в виде чертежа, показанного на рис. 1,*а*, несложно составить выражение для вычисления длины трассы

$$l = c + \sqrt{a^2 + x^2} + \sqrt{b^2 + (d - c - x)^2},$$

однозначно зависящей от расстояния x при заданных значениях a, b, c и d. В этом случае можно решить задачу методом последовательных приближений по значению x^*, соответствующему наименьшему из ряда вычисленных значений l.

Решение задачи таким методом связано со значительными затратами времени. Проще решить задачу, определив зеркальное отображение точки B и сместив его на расстояние c (рис. 1,*б*). В этом случае, соединив прямой (и, следовательно, кратчайшей) линией точки A и B', из подобия треугольников получим значение

$$x^* = a\,(d - c) / (a + b),$$

соответствующее минимальной длине трассы

$$l = c + a\sqrt{1 - (d - c)^2/(a + b)^2} + \sqrt{b^2 + (d - c)^2\,(1 - a/(a + b))^2}.$$

Один и тот же метод можно реализовать различными способами, выбор которых определяется критериями качества (оптимальности) результата решения задачи и пути его получения, а также возможностью выполнить требуемые действия. Способ решения конкретной задачи отображают алгоритмом — конечной последовательностью однозначных описаний операций (действий), выполнимых исполнителем алгоритма и приводящих к искомому результату. Если исполнитель не может выполнить операцию, то ее описание в алгоритме заменяют описаниями других выполнимых операций.

Алгоритм решения задачи представляют тремя основными способами — словесно-формульным описанием, схемой или программой. Словесно-формульное описание образовано последовательностью шагов (блоков, инструкций, операторов*) алгоритма в виде описаний операций с указанием при необходимости шага, выполняемого следующим.

В зависимости от вида операции различают обычные, условные и концевые блоки. Обычные блоки содержат описания операций идентификации переменных с помощью формул «принять $I = A$» или «вычислить $I = A$», называемых операторами присваивания. Такая формула означает, что в дальнейшем переменной, обозначенной символом (именем, идентификатором), записанным слева от знака равенства, присваивается значение, равное числу A или результату вычислений по выражению A, записанному справа от знака равенства. Например, оператор присваивания $x = x + 1$ означает, что в дальнейшем переменной x присваивается значение, на единицу большее предыдущего.

При отсутствии оговорок после обычного блока переходят к следующему по порядку блоку. Если необходимо перейти к блоку, следующему не по порядку нумерации, то переход называют безусловным, а описание оператора присваивания дополняют указанием на такой переход.

Условные блоки содержат описания операций проверки выполнения определенного условия в виде условных предложений, например, «если выполнено условие ..., то перейти к блоку ... , иначе к блоку ...». В этом описании, называемом условным оператором или оператором условного перехода, проверяемое условие часто обозначают формулой xAz, где A — знак отношения (равенства или неравенства) между переменной x и эталонным числом z. Переходы от условных блоков также называют условными.

Концевые блоки содержат описания операции прекращения вычислений, а концевой блок в начале описания алгоритма — описание исходных данных. Иногда исходные данные описывают отдельно, а указания на прекращение вычислений помещают в обычных или условных блоках.

Для бóльшей наглядности алгоритм представляют схемой, в которой обычные блоки обозначают прямоугольниками, концевые — овалами, а условные — близкими к ромбу фигурами с выходом «Да» или 1 при выполнении проверяемого условия и «Нет» или 0 при его невыполнении. Все блоки схемы соединяют линиями со стрелками, указывающими на очередность выполнения блоков.

При отсутствии условных блоков алгоритм отображается неразветвленной последовательностью обычных и концевых блоков схемы (рис. 2,*а*). С условными блоками связаны четыре основные структуры — простые разветвление (рис. 2,*б*) и цикл (рис. 2, *в*) и сложные

---

* *Оператор* — тот, кто (или то, что) управляет выполнением операций. В математике оператором называют обозначенное некоторым символом правило сопоставления элементов одного множества (например, исходных данных) элементам другого (например, результатов вычислений).

разветвление (рис. 2,*г*) и цикл (рис. 2,*д*) с блоками в обеих ветвях. Циклы используют для многократного выполнения определенной последовательности операций (итераций) с выходом из цикла после выполнения некоторого условия, проверяемого условным оператором. Схождение ветвей в сложных разветвлениях и циклах обеспечивается безусловными переходами.

Форма словесно-формульного описания или другого представления алгоритма зависит от решаемой задачи, искусства составителя алгоритма и возможностей его исполнителя. Так, для математика достаточно слов «вычислить факториал $a!$ небольшого неотрицательного числа a», тогда как для менее опытного исполнителя необходима более подробная инструкция, например:

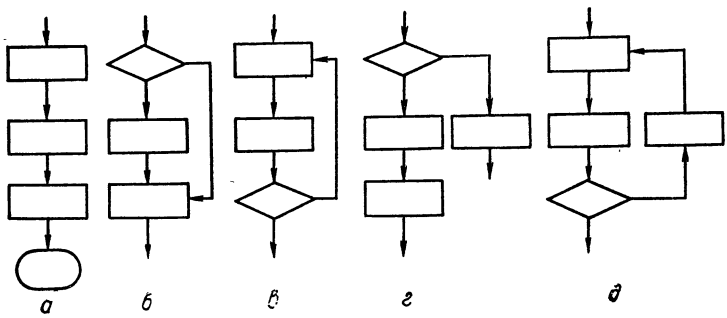


Рис. 2

1. Если $a = 0$ или $a = 1$, то принять $a! = 1$ и закончить вычисления, иначе перейти к п. 2.
2. Вычислить факториал по формуле $a! = a\,(a-1)(a-2)\ldots 3 \cdot 2$ и закончить решение задачи.

Исполнитель этого алгоритма должен выполнить вычисления по мысленно или на бумаге составленной формуле описанного вида для заданного числа. Можно избежать составления подобной формулы, организовав вычисления в цикле по формуле $P_{i+1} = P_i(m_i - 1)$; $i = 0,1,\ldots$ с начальными значениями $P_0 = m_0 = a$ и прекращением вычислений (выходом из цикла) при выполнении условия $m_i - 1 = 0$. Алгоритм решения задачи в этом случае можно представить следующим описанием:

1. Если $a = 0$, то перейти к блоку 2, иначе к блоку 4.
2. Принять $a! = 1$.
3. Закончить вычисления.
4. Принять $P = a$.
5. Принять $m = a$.
6. Вычислить $m = m - 1$.
7. Если $m = 0$, то перейти к блоку 8, иначе к блоку 10.
8. Принять $a! = P$.
9. Закончить вычисления.
10. Вычислить $P = Pm$ и перейти к блоку 6.

В этом описании блоки 1 и 7 — условные; 2, 4, 5 и 8 — обычные идентификационные; 6 и 10 — обычные функциональные; 3 и 9 — концевые. Условные переходы от блоков 1 и 7 и безусловный переход от блока 10 наглядно видны на схеме описанного алгоритма (рис. 3,*а*).

Представления алгоритма можно преобразовать различными способами, а описания операций упростить. Если договориться о том, что после выполнения условия, проверяемого условным оператором, переходят к следующему блоку, то после такого оператора достаточно указать на переход при невыполнении проверяемого условия. Текущую переменную (результат выполнения операции в обычном блоке) для упрощения описания алгоритма и стандартизации условных операторов целесообразно обозначать символом x. Можно сократить количество обозначений, приняв формулу $x = \text{ПN}$ при занесении текущей переменной x в N-е место памяти (при записи данных на листе бумаги или другом носителе информации) и формулу ИПN $= x$ при вызове переменной x из N-го места памяти. Удобно стандартизировать и описания вычислительных операций, выделив в отдельные блоки только описание выполняемой операции и переменные, над которыми выполняется операция. В этом случае необходимо лишь условиться о порядке следования блоков с описаниями операций (операторами) и описаниями переменных. Если воспользоваться обычным способом записи арифметических операций, то их описания должны находиться между описаниями переменных, над которыми выполняется операция. Однако можно условиться о порядке следования блоков, аналогичном вычислениям «в столбик», когда вначале выписываются два числа, а затем над ними выполняется арифметическая операция. Для такой последовательности, когда вначале следуют блоки переменных, а после них блок выполняемой операции, с учетом предыдущих замечаний составим следующий алгоритм вычисления факториала (рис. 3,*б*):

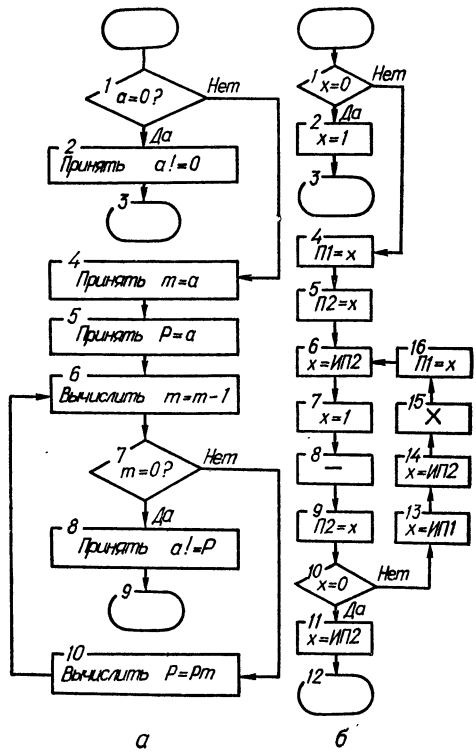


Рис. 3

1. Если условие $x = 0$ не выполняется, то перейти к блоку **4.**
2. Принять $x = 1$.
3. Закончить вычисления.

4. Принять $x = П1$.
5. Принять $x = П2$.
6. Принять $ИП2 = x$.
7. Принять $x = 1$.
8. Выполнить вычитание.
9. Принять $x = П2$.
10. Если условие $x = 0$ не выполняется, то перейти к блоку 13.
11. Принять $ИП1 = x$.
12. Закончить вычисления.
13. Принять $ИП1 = x$.
14. Принять $ИП2 = x$.
15. Выполнить умножение.
16. Принять $x = П1$ и перейти к блоку 6.

Исполнителем алгоритма может быть не только человек, но и автомат (например ЭВМ), конструкция которого обеспечивает выполнение требуемых операций по командам, соответствующим описаниям операций в алгоритме. Такой автомат должен содержать устройство, различающее вводимые команды и, следовательно, описание операций в алгоритме в этом случае должно быть составлено на языке с определенным запасом команд (операторов), различаемых автоматом, и синтаксическими правилами, определяющими допустимый порядок ввода команд. Подобные *алгоритмические* языки называют входными или внешними (в отличие от внутренних, элементами которых являются электрические сигналы), а составленные на этих языках описания алгоритмов — *программами*.

Основное отличие входных алгоритмических языков клавишных электронных вычислительных машин (ЭКВМ), к которым относятся микрокалькуляторы, и универсальных ЭВМ с большой емкостью запоминающих устройств связано со способами реализации операторов присваивания. Конструкции универсальных ЭВМ при вводе таких операторов обеспечивают автоматическое распределение в памяти и вызов из нее исходных данных и результатов вычислений с помощью идентификаторов. Это существенно упрощает составление программ, но не всегда обеспечивает достаточно полное использование емкости запоминающих устройств. Во входных языках ЭКВМ с малой емкостью запоминающих устройств для их полного использования засылка данных в память и вызов из нее реализуются при помощи специальных операторов обращения к памяти и составителю программ приходится самому распределять в памяти исходные данные и результаты выполнения операций. Поэтому по структуре программы для ЭКВМ близки к описанию алгоритма, схема которого показана на рис. 3,*б*.

Внешние отличия программ на алгоритмических языках клавишных и универсальных ЭВМ связаны со способом ввода информации. В программах на алгоритмических языках универсальных ЭВМ операторы, вводимые с помощью алфавитно-цифрового печатающего устройства, по форме близки к описаниям операций в словесно-формульных описаниях алгоритма (например, алгоритма, схема которого показана на рис. 3,*а*). В ЭКВМ операторы программы вводят нажатиями одной или нескольких клавиш, а программы записывают в виде последова-

тельности символов операторов, соответствующих сокращенным обозначениям на клавишах.

С учетом этих особенностей основные отличия программ от описаний алгоритмов определяются способом указания переходов. Для описания безусловных переходов в программах используют специальные операторы безусловного перехода, например, «ИДТИ НА m» или «GO TO m» в алгоритмических языках универсальных ЭВМ и «БП m» или «GTO m» во входных языках ЭКВМ, где m — указатель оператора (шага) программы, который выполняется после перехода. Указатели условных переходов содержатся непосредственно в условных операторах, например, «IF $x = 0$ THEN m_1 ELSE m_2» или «ЕСЛИ $x = 0$ ТО m_1 ИНАЧЕ m_2» в алгоритмических языках универсальных ЭВМ и «$x = 0\ m_2$» во входных языках ЭКВМ, причем в последних после символа проверяемого условия записывается указатель перехода при невыполнении (в некоторых языках — при выполнении) условия, а при его выполнении (невыполнении) выполняется оператор, следующий в программе за оператором условного перехода.

Во входных языках универсальных ЭВМ и некоторых ЭКВМ в качестве указателей перехода используют *метки* — допустимые правилами входного языка символы, записываемые также перед операторами (называемыми отмеченными), выполняемыми после перехода. В большинстве входных языков ЭКВМ используются указатели переходов в виде *адресов* — кодов порядковых номеров шагов программы, с которых продолжается ее выполнение после перехода.

Например, если обозначить оператор прекращения вычислений символом С/П, а операторы вычитания и умножения символами — и × и закодировать порядковые номера шагов программы адресами 00, 01, 02, ..., то алгоритм, схема которого показана на рис. 3,*б*, при записи операторов программы по строкам можно представить в следующем виде, близком к программным представлениям алгоритмов на входных языках ЭКВМ:

$x = 0$ 04 1 С/П П1 П2 ИП2 1 — П2
$x = 0$ 14 ИП1 С/П ИП1 × П1 БП 06

Если во входном языке в качестве указателей переходов используют метки (например, в виде символов MN), то рассматриваемый алгоритм можно представить программой

$x = 0$ М1 1 С/П М1 П1 П2 М2 ИП2 1
— П2 $x = 0$ М3 ИП1 С/П М3 ИП1 × П1
БП М2

При указателях переходов в виде меток составление программы несколько упрощается, так как не требуется определять адреса переходов, но число указателей переходов и, соответственно, длина программы увеличиваются.

Программу для наглядности переходов можно представить схемой, каждый блок которой соответствует одному или, для неразветвленных частей программы, нескольким шагам программы. Такая схема будет

представлять алгоритм вычислений с учетом особенностей входного языка. Число блоков подобного алгоритма и, соответственно, длина программы зависят от уровня входного языка, определяемого количеством операторов и сложностью управляемых ими операций. Например, если входной язык содержит оператор вычисления факториала, то приведенные программы можно заменить одним таким оператором.

2. СИСТЕМЫ КОМАНД И ВХОДНЫЕ ЯЗЫКИ МИКРОКАЛЬКУЛЯТОРОВ

Калькуляторами называли ЭКВМ первого поколения, выполнявшие при нажатии клавиш только арифметические операции, но это название сохранилось и для современных карманных и настольных калькуляторов (микрокалькуляторов), отличающихся от ЭКВМ других типов малыми размерами.

Различают *непрограммируемые* и *программируемые* микрокалькуляторы. Первые предназначены для автоматического выполнения операций над числами по командам, подаваемым нажатиями клавиш. Вторые, кроме такого *обычного* (ручного) режима, могут работать в *программируемом* режиме автоматических вычислений по программе, предварительно введенной нажатием клавиш в режиме *программирования*. В обоих случаях программа, представляющая алгоритм вычислений на входном языке микрокалькулятора, вводится нажатиями клавиш.

При нажатии клавиш вырабатываются электрические сигналы (команды), управляющие действиями микрокалькулятора в соответствии с символом команды. Система команд связана с внутренним языком микрокалькулятора и в общем случае содержит служебные команды, не связанные непосредственно с выполнением алгоритма вычислений. Поэтому следует различать систему команд и входной язык микрокалькулятора, определяющий не только выполняемые микрокалькулятором вычислительные операции, но и допустимую форму алгоритма вычислений, представляемого программой.

Входной язык характеризуется *алфавитом*, образованным множеством символов всех или части команд, обозначенных на клавиатуре микрокалькулятора, *словарным запасом*, образованным допустимыми и семантически (по смыслу) неделимыми сочетаниями элементов алфавита — словами или *операторами* входного языка, а также *грамматикой*, включающей *лексические* правила образования слов и *синтаксические* правила, определяющие допустимые последовательности слов. Несколько слов, приводящих к имеющему самостоятельное значение результату, образуют предложение, а одно или несколько предложений — программу.

Именам существительным и глаголам естественных языков во входных языках соответствуют *операнды* (числа, над которыми выполняются операции) и *операторы*, управляющие выполнением операций. Однако ввод операндов в вычисления также реализуется при помощи операторов и поэтому словами входного языка являются операторы,

занимающие один или два (операторы переходов в программах автоматических вычислений) шага программы.

Операторы (слова) входного языка в соответствии с их назначением можно разбить на несколько групп: набора десятичных представлений чисел, функциональных (вычислительных), обращения к памяти и «синтаксических», связанных с синтаксическими особенностями входных языков и обеспечивающих изменение порядка следования операндов. Во входных языках программируемых микрокалькуляторов имеется также группа операторов управления выполнением программы в режиме автоматических вычислений.

Группа операторов **набора десятичных представлений чисел** во всех микрокалькуляторах содержит операторы набора цифр от 0 до 9 и десятичного разделительного знака (точки или запятой) с одинаково расположенными нажимаемыми клавишами, а также оператор изменения знака числа, обозначаемый символами /—/, +/— или CHS (change sign — изменить знак). Если предусмотрена показательная форма представления чисел $a = M \cdot 10^n$, то имеется оператор ввода порядка, обозначаемый символами ВП, n, EEX или EE (enter exponent — ввести порядок), после которого набирают знаки порядка. К этой же группе относится и оператор набора постоянной $\pi = 3{,}1415926$.

Группа **функциональных** операторов содержит одноместные, двухместные и многоместные операторы, предназначенные для выполнения вычислений над одним, двумя и большим числом операндов соответственно. К *одноместным* относятся, в частности, операторы $1/x$, x^2 и $\sqrt{}$ для вычисления простейших степенных функций $y = x^{-1}$, $y = x^2$ и $y = x^{1/2}$, основанием которых является текущая переменная, тригонометрических, гиперболических и обратных им функций, а также операторы lg и ln для вычисления десятичных и натуральных логарифмов, операторы e^x и 10^x для вычисления антилогарифмов и оператор вычисления факториала (обозначаемый символом $x!$ или $n!$), модуля, целой и дробной частей числа и др.

Двухместными являются имеющиеся во входных языках всех микрокалькуляторов операторы +, —, × и ÷ для выполнения арифметических операций сложения, вычитания, умножения и деления соответственно, а также оператор вычисления степенной функции общего вида, обозначаемый символом X^y или Y^x. К двухместным относятся также операторы преобразования декартовых координат в полярные и некоторые другие.

К *многоместным* относятся имеющиеся в некоторых входных языках операторы вычисления вещественных корней квадратных уравнений, оператор LR для вычисления линейной регрессии, оператор вычисления детерминанта системы из двух линейных уравнений. Во входных языках некоторых программируемых микрокалькуляторов содержится оператор OPN, где N — номер относительно редко используемой многоместной операции, перечень которых приведен в инструкции по применению микрокалькулятора.

Группа операторов **обращения к памяти,** выполняющих функции операторов присваивания во входных языках универсальных ЭВМ, содержит операторы засылки текущей переменной в память, обозна-

чаемые символами ЗП, ЗАП, П, Р, $x \to m$, X → П или STO (storage — хранить), и операторы вызова из памяти, обозначаемые символами СЧ, ИП, F, $m \to x$, П → X или RCL (recall — вызвать). Входные языки некоторых микрокалькуляторов содержат операторы стирания содержимого памяти, обозначаемые символами СП или CLM (clearn memory — очистить память), а также оператор обмена, обозначаемый символами X ↔ П или EXC (exchange — обменять), который вызывает содержимое памяти, а на его место заносит текущую переменную. Если микрокалькулятор содержит несколько *регистров* памяти (устройств для хранения отдельных чисел), то оператор обращения к памяти дополняется командой набора номера нужного регистра.

К этой же группе можно отнести гибридные операторы, выполняющие двухместные операции над текущей переменной и содержимым регистра памяти с последующей засылкой в последний результат вычислений. К ним относятся операторы, обозначаемые символами П+, П—, П×, П÷, SUM (сумма), PRD (product — произведение), а также оператор П + x^2, добавляющий к содержимому регистра памяти квадрат текущей переменной.

Отдельную **«синтаксическую»** группу образуют операторы: вывода результатов (обычно используется символ =), разделения операндов (↑, B↑, E↑ или ENTER↑), перестановки операндов (X⇄Y, XY, ↔), стирания текущей переменной (C, Cx, CLX) и некоторые другие.

Основой синтаксиса входного языка микрокалькулятора служат алгебраические правила записи расчетных формул, используемых при вычислениях, после замены протяженных символов (например, дробных черт или радикалов) дискретными. Однако и такая запись, называемая *естественной* алгебраической, в большей степени отображает отношения между числами, чем способ вычисления результата. Например, формулу $y = a - b \ln (c/d)$ можно описать словами: «число y равно разности числа a и произведения числа b на логарифм отношения чисел c к d». При вычислении же числа y, равного правой части этой формулы, ее следует представить в естественной алгебраической записи

$$a - b \times \ln (c \div d) = \quad ,$$

отображающей при ее чтении (вводе) слева направо последовательность необходимых действий: «запомнить число a, знак вычитания, число b, знак умножения, знак вычисления алгоритма, открывающую скобку, число c, знак деления, число d; после ввода закрывающей скобки вычислить частное от деления числа c на число d, после ввода символа получения результата (=) вычислить логарифм этого частного, умножить результат на число b и вычесть полученное произведение из числа a».

Следовательно, при естественной алгебраической записи рассмотренной формулы для вычисления результата микрокалькулятор должен запомнить все операнды, набранные операторами набора чисел или вызванные из памяти, и функциональные операторы, введенные до закрывающей скобки, и лишь после этого начинать вычисления.

Кроме того, конструкция микрокалькулятора должна обеспечивать выполнение старших операций перед младшими, например, умножения и деления перед вычитанием и сложением. В связи с подобными требованиями к конструкции естественную алгебраическую запись используют редко и чаще прибегают к ее различным модификациям.

Во входных языках большинства микрокалькуляторов используют *простую* алгебраическую запись, где одноместные операторы записывают после операндов (которыми могут являться и результаты предыдущих вычислений), например,

$$a - b \times (c \div d) \ln = .$$

Конструкция микрокалькуляторов в этом случае упрощается, так как одноместные операторы могут выполняться непосредственно после их ввода и отпадает необходимость в их запоминании.

Для упрощения определения микрокалькулятором старшинства операций во входных языках часто используют простую *скобочную* запись, например,

$$a - (b \times (c \div d) \ln) = ,$$

при которой единственным признаком старшинства следующей операции является оператор открывающей скобки. Максимальное число открывающих скобок, вводимых перед закрывающей (число «скобок в скобках»), определяется конструкцией микрокалькулятора и соответствующими синтаксическими особенностями входного языка и на 2 меньше числа операндов, запоминаемых в процессе вычислений без обращения к памяти с помощью соответствующих операторов. Если допустима только одна открывающая скобка перед закрывающей, то рассматриваемое выражение следует преобразовать к виду

$$a - (c \div d = \ln \times b) = ,$$

при котором в процессе вычислений требуется запомнить одновременно не более трех операндов.

Во входных языках простейших микрокалькуляторов обычно используют простую *бесскобочную* запись с одновременным запоминанием не более двух операндов, участвующих в двухместных операциях. В этом случае приходится записывать вычисляемые выражения с учетом выполнения операций по старшинству, например

$$c \div d = \ln \times b \ /-/ + a = ,$$

и, при необходимости, использовать операторы обращения к памяти или записывать промежуточные результаты в вычислительном бланке.

В микрокалькуляторах с алгебраическим синтаксисом входного языка, при котором двухместные операторы вводят между операндами, двухместная операция не может быть выполнена до получения сигнала об окончании набора операнда и о том, что следующая операция не является старшей. Таким сигналом является ввод следующего двухместного оператора, если он не является старшим (для микрокалькуляторов, автоматически учитывающих старшинство операторов $\times$

и ÷ относительно операторов + и —), закрывающей скобки или оператора вывода результата =. Так как операторы скобок имеются только в языках со скобочной записью, то отличительной особенностью входных языков с алгебраическим синтаксисом является оператор вывода результата, обычно обозначаемый символом = .

Существенное упрощение конструкции микрокалькулятора и учета старшинства операций достигается при использовании обратной записи не только для одноместных, но и для двухместных операторов. При такой записи двухместных операций (ее называют обратной польской) двухместный оператор может быть выполнен микрокалькулятором без запоминания. Для разделения набираемых операндов используют оператор засылки ↑ (который в некоторых входных языках с обратным синтаксисом вводится и перед вызываемым из памяти операндом). Обратная польская запись соответствует реальному выполнению арифметических операций человеком, который вначале «в столбик» записывает оба многозначных операнда и лишь после этого выполняет над ними необходимые действия.

Форма записи программ на входных языках с *обратным синтаксисом* зависит от максимального числа операндов, запоминаемых микрокалькулятором в процессе вычислений (без ввода операторов вызова из памяти), и правил ввода оператора засылки ↑. Например, если этот оператор вводится только между набираемыми операндами (что характерно для большинства языков с обратным синтаксисом) и максимальное число одновременно запоминаемых операндов не менее четырех, то рассмотренное ранее выражение можно записать в обратной записи (если все операнды набираются)

$$a \uparrow b \uparrow c \uparrow d \div \ln \times -$$

или, если все операнды вызываются из памяти,

$$\text{ИП1 ИП2 ИП3 ИП4} \div \ln \times - .$$

Если максимальное число запоминаемых одновременно операндов не более трех и все операторы набираются, то

$$a \uparrow c \uparrow d \div \ln b \times - ,$$

а при минимальном числе (два) запоминаемых операндов и их наборе

$$c \uparrow d \div \ln b \times a \,/ - / + .$$

Правила обратной записи аналогичны синтаксическим правилам обычных языков для команд, требующих немедленного исполнения. Так, если алгебраическая запись $\sqrt{a}$ описывается предложением «извлечь корень из числа a», то обратной записи $a\ \sqrt{}$ соответствует команда «из числа a корень извлечь!». Аналогично, если запись $a + b =$ переводится предложением «число a сложить с числом b и записать результат», то записи $a \uparrow b +$ соответствует команда «число a и число b сложить!».

Построение синтаксиса входного языка на основе обратной записи функциональных операторов упрощает конструкцию микрокалькулятора, так как необходимо запоминать только операнды, но несколь-

ко затрудняет вычисления по готовым формулам, которые пользователю приходится мысленно или на бумаге представлять в обратной записи. Поэтому для массовых непрограммируемых микрокалькуляторов характерны входные языки с алгебраическим синтаксисом, тогда как входные языки с обратным синтаксисом характерны для программируемых микрокалькуляторов, предназначенных для более подготовленных пользователей. Иногда входные языки с алгебраическим синтаксисом используют в микрокалькуляторах, стоимость которых значительно больше изменений стоимости, связанной с использованием обратного синтаксиса.

Входные языки всех программируемых микрокалькуляторов содержат, кроме перечисленных, группу операторов **управления программой автоматических вычислений.** К ним относится оператор прекращения вычислений, обозначаемый символом С/П («стоп/пуск») или R/S (run/stop — пуск/стоп), операторы условных и безусловных переходов и, иногда, операторы установки флагов.

Оператор *безусловного* перехода общего вида БП *m* или GTO *m* занимает два шага программы автоматических вычислений, причем второй шаг занимает указатель *m* перехода в виде адреса или метки. Во входных языках с указателями переходов в виде метки последние набираются командами LB *N* (label — метка), где *N* — номер метки, или нажатиями клавиш с обозначениями букв A, B, C, ... Во входных языках с указателями переходов в виде адресов последние набираются цифрами или нажатиями других клавиш.

К операторам безусловного перехода относится и оператор *обращения к подпрограмме* ПП *m*, SBR *m* или SRT *m* (subroutine — процедура), занимающий два шага программы и обеспечивающий переход к многократно используемому фрагменту программы (подпрограмме). В конце каждой подпрограммы записывают оператор безусловного перехода В/О (возврата/очистки), обозначаемый во входных языках зарубежных микрокалькуляторов символами RES или RST (reset — восстановление). После этого оператора указатель перехода не записывают. Во входных языках программируемых микрокалькуляторов имеются *условные* операторы (операторы условного перехода) вида *xAz m*, где *A* — символ проверяемого условия (равенства или неравенства), *z* — эталонное число (которым может быть нуль, предыдущий операнд *y* или содержимое *t* специального регистра памяти), а *m* — указатель перехода при невыполнении (в некоторых языках — при выполнении) проверяемого условия. Если условие выполнено (не выполнено), то выполняется оператор, следующий в программе за условным оператором.

Условные операторы *цикла* LN *m*, также занимающие два шага программы, проверяют содержимое регистра *N* памяти, уменьшающееся на единицу после каждого выполнения такого оператора, на его равенство нулю. Если это равенство не соблюдается, то следующим выполняется оператор в предыдущей части программы, определяемый указателем *m* перехода. Если содержимое регистра *N* стало равным нулю, то выполняется оператор, следующий в программе после условного оператора цикла.

Иногда входные языки программируемых микрокалькуляторов содержат операторы установки *флага* Ф или ST FG (set flag — установить флаг) и условные операторы проверки флага Ф? *m* или IF FG *m* (if flag — установлен ли флаг). Установка флага (своеобразного «узелка на память») соответствует изменению состояния специального внутреннего двоичного устройства, а условный оператор проверки флага передает управление следующему оператору программы, если флаг не установлен, или другому оператору программы, указанному меткой или адресом *m*, если флаг установлен. Обычно оператор установки флага вводят в одну из ветвей после условного оператора, проверяющего выполнение некоторого условия, а операторы проверки флага вводят в тех случаях, когда дальнейшее выполнение программы зависит от выполнения ранее проверенного условия.

Во входных языках большинства программируемых микрокалькуляторов предусмотрена косвенная адресация условных и безусловных переходов, а иногда и косвенная адресация обращений к памяти. Операторы косвенных переходов или обращений к памяти содержат указатель номера регистра памяти, в котором соответственно хранится адрес перехода или номер регистра памяти, в который засылается или из которого вызывается текущая переменная. Над содержимым указанного регистра памяти могут выполняться автоматически или по обычным командам арифметические операции, что обеспечивает его изменение при повторных выполнениях операторов. Эти операторы обозначают в различных языках разнообразными символами, но они занимают один шаг программы, хотя набираются нажатиями нескольких клавиш.

Некоторые входные языки содержат оператор (PAUSE), обеспечивающий остановку на несколько секунд выполнения программы автоматических вычислений для регистрации высвечиваемого промежуточного результата.

Система команд любого микрокалькулятора содержит также *служебные* команды (символы которых не входят в алфавит входного языка), вводимые нажатием клавиш или установкой переключателей в определенное положение. К ним относятся команда включения микрокалькулятора, команды выбора размерности аргумента тригонометрических функций в градусах (degrees), радианах (radians) или десятичных градусах (grades), соответствующих делению прямого угла на 100 частей; команды выбора размерности операндов; команды перевода градусов, минут и секунд в десятичное представление градусов. К служебным относятся также имеющиеся в системе команд некоторых микрокалькуляторов с показательной формой представления чисел команды SCI (scientific — научный) для вывода результата с одной отличной от нуля цифрой в целой части мантиссы и ENG для вывода результата с порядком, кратным трем, что удобно при инженерных расчетах с единицами размерности, отличающимися в 1000 раз (например, граммы, килограммы и тонны).

В большинстве микрокалькуляторов для уменьшения общего числа клавиш предусмотрено использование *префиксных* команд (модификаторов или переключателей), относящихся к трем основным типам. Пре-

фиксные клавиши с символом команды-модификатора первого типа (F, P, *g*, *f*, 2 — *nd* и др.) нажимают для ввода команды или оператора, обозначенных над или под клавишей, нажимаемой следующей. При ошибочном нажатии префиксной клавиши для снятия совмещенного режима нажимают клавишу, обычно обозначаемую символом CF.

Префиксные команды второго типа обозначают символами, являющимися составными частями символов операторов входного языка. К таким командам относятся arc и hyp, совместный или раздельный ввод которых перед оператором вычисления тригонометрической функции обеспечивает вычисление соответствующей обратной тригонометрической, гиперболической или обратной гиперболической функции.

Префиксные команды третьего типа, обычно обозначаемые символами INV или F^{-1}, изменяют на обратное назначение операторов, вводимых следующими. Так, ввод INV tg обеспечивает вычисление arc tg x, ввод INV lg приводит к вычислению антилогарифма 10^x, а ввод INV $x = 0$ обеспечивает проверку условия $x \neq 0$.

Системы команд всех программируемых микрокалькуляторов содержат команды перехода в режим *программирования*, обозначаемые символами РП, ПРГ или LPN (learn — обучать), и в *рабочий* режим, обозначаемые символами РР, АВТ или USER (пользователь), а также предназначенные для редактирования вводимой программы команды смещения программы на шаг вперед, обозначаемой символами $\overrightarrow{\text{ШГ}}$ или SST (single step — одиночный шаг), и шаг назад, обозначаемой символами $\overleftarrow{\text{ШГ}}$ или BST (back step — шаг назад), команды стирания ошибочно введенного оператора программы, обозначаемой символом НОП или NOP (нет оператора), и некоторые другие.

Следует отметить, что команды перехода в заданный режим работы, часто вводимые при помощи префиксных клавиш, также являются модификаторами. Так, при нажатии клавиши С/П в режиме программирования в программу заносится оператор остановки, но нажатие этой клавиши в рабочем режиме обеспечивает выполнение программы автоматических вычислений в программируемом режиме, во время которого нажатие клавиши С/П приводит к аварийной остановке выполнения программы.

Таким образом, входной алгоритмический язык, являющийся средством представления алгоритма вычислений программой, выполнимой микрокалькулятором, охватывает лишь часть команд. Это различие особенно ощутимо при составлении программ автоматических вычислений для программируемых микрокалькуляторов. Однако выполнение вычислений в обычном режиме удобнее всего рассмотреть на примере непрограммируемых микрокалькуляторов различных типов.

3. НЕПРОГРАММИРУЕМЫЕ МИКРОКАЛЬКУЛЯТОРЫ

Непрограммируемые микрокалькуляторы подразделяются на простейшие (арифметические) и специализированные (например, инженерные или научные и коммерческие). *Простейшие* микрокалькулято-

ры предназначены для выполнения над заданными числами арифметических действий и, иногда, вычисления простейших функций — процентов, обратных величин, квадратных корней или квадратов чисел. *Специализированные* микрокалькуляторы при вводе соответствующих операторов дополнительно вычисляют элементарные и специальные (например, сложные проценты) функции, характерные для расчетов в определенной области деятельности пользователей микрокалькуляторов.

По сложности устройства и количеству элементов даже простейшие микрокалькуляторы превосходят универсальные ЭВМ первого поколения. Однако для изучения связи между нажатиями клавиш и результатами выполнения команд непрограммируемым микрокалькулятором достаточно воспользоваться его обобщенной функциональной схемой (рис. 4), не совпадающей со структурными схемами микрокалькуляторов конкретных типов, но отображающей взаимодействие их основных узлов.

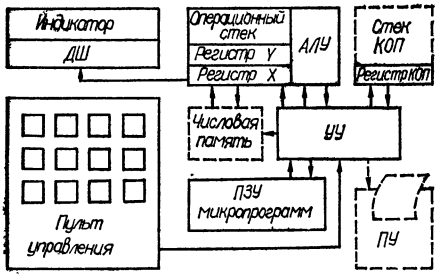


Рис. 4

В непрограммируемых микрокалькуляторах практически единственным устройством ввода информации является пульт управления с клавишами и переключателями для подачи команд, а основным и обычно единственным устройством вывода информации служит индикатор для высвечивания десятичных представлений чисел и иногда символов, характеризующих режим работы. Лишь некоторые настольные микрокалькуляторы дополнительно снабжены миниатюрным печатающим устройством *ПУ* для вывода результатов вычислений на бумажную ленту.

Индикатор содержит цифровые знакоместа, образованные из сегментов (обычно семи) на светодиодах, жидких кристаллах или газонаполненных трубках, а также отдельные сегменты для высвечивания запятой или точки, отрицательного знака и иногда дополнительных символов.

В простейших микрокалькуляторах числа обычно индицируются в естественной форме с запятой, фиксированной между целой и дробной частью. При r цифровых знакоместах индикатора, соответствующих цифровым разрядам, диапазон представления чисел в естественной форме ограничен по модулю пределами $A_{\min}=10^{1-r}$ и $A_{\max}=10^r-1$. В большинстве специализированных микрокалькуляторов предусмотрено высвечивание результата вычислений в стандартной (научной) показательной форме $x=M\cdot 10^n$ со значением мантиссы $1\leqslant M<10$. В этом случае при r цифровых знакоместах индикатора для мантиссы и m для порядка диапазон представления результатов вычислений ограничен по модулю более широкими пределами: $A_{\min}=10^{1-10^m}$ и $A_{\max}=(10-10^{1-r})\,10^{10^m-1}$. Результаты вычислений, отличающиеся от нуля, но меньшие по модулю $A_{\min}$, попадают в область *машинного нуля* и

на индикаторе высвечивается нуль.Если модуль результата вычислений больше A_{max}, как и при попытке выполнить некорректную операцию (например, вычислить логарифм или корень отрицательного числа), возникает переполнение запоминающих устройств микрокалькулятора, о чем свидетельствует определенный сигнал — высвечивание специальных символов, стирание индицируемого числа и т. п.

Основным функциональным узлом микрокалькулятора является *процессор* (вычислитель), к которому относится управляющее устройство *УУ*, синхронизирующее во времени и определяющее необходимые действия других устройств, два или более операционных регистров, арифметико-логическое устройство *АЛУ* и, в микрокалькуляторах с алгебраическим синтаксисом входного языка, один или несколько регистров *КОП* для хранения кодов операций.

Операционные регистры предназначены для хранения в процессе вычислений двухпозиционных (обычно двоично-десятичных) представлений операндов и результатов операций. Входной регистр (аккумулятор), называемый также индикаторным регистром или регистром *X*, соединен через дешифраторное устройство *ДШ* с индикатором. Дешифратор преобразует представление числа, хранящееся в регистре *X*, в напряжения, подаваемые на соответствующие сегменты знакомест индикатора для высвечивания десятичного представления содержимого регистра *X*. При вводе операторов набора чисел нажатием клавиш коды десятичных знаков заносятся в регистр *X*, а их десятичные представления высвечиваются на индикаторе.

Регистр *X* соединен также с одним или несколькими регистрами *числовой* памяти (запоминающего устройства для хранения чисел), если она предусмотрена. При вводе нажатием клавиш операторов засылки в память содержимое регистра *X* переписывается в соответствующий регистр памяти, а при вводе операторов вызова из памяти содержимое соответствующего регистра переписывается в регистр *X*.

Операционные регистры *X* и *Y* предназначены для хранения операндов, участвующих в двухместных операциях, а одноместные операции выполняются над содержимым регистра *X*. Если предусмотрена возможность вычислений со скобками, то имеются дополнительные операционные регистры для временного запоминания операндов (или результатов предыдущих операций), вводимых перед открывающими скобками. Число таких дополнительных регистров определяет допустимое число «скобок в скобках» при вычислениях. Операционные регистры соединены между собой так, что по специальным командам *УУ* содержимое каждого регистра засылается в соседний регистр. Такое соединение регистров называют магазинным или кольцевым *стеком.*

При смещении «вверх» *магазинного* стека содержимое каждого регистра засылается в соседний «верхний» регистр, причем прежнее содержимое последнего регистра стирается, а содержимое регистра *X* сохраняется (рис. 5,*а*). При смещении такого стека «вниз» содержимое последнего регистра обычно сохраняется (рис. 5,*б*) лишь в микрокалькуляторах с обратным синтаксисом входного языка. При смещении «вверх» *кольцевого* стека содержимое его последнего регистра

засылается в регистр X (рис. 5,*в*), а при смещении такого стека «вниз» содержимое регистра X засылается в последний регистр (рис. 5,*г*). Работу кольцевого стека удобно отображать схемой (рис. 5,*д*), в которой смещения стека «вверх» и «вниз» заменены его поворотами по и против часовой стрелки соответственно. Операционные стеки непрограммируемых микрокалькуляторов обычно работают только в магазинном режиме.

Арифметико-логическое устройство образовано логическими элементами, выполняющими над содержимым регистров X и Y элементарные операции *(микрооперации)* сложения и умножения содержимого двоичных разрядов, переноса единицы в старший разряд, сдвига содержимого регистров на один разряд, и содержит внутренние «буферные» регистры для временного хранения результатов промежуточных вычислений.

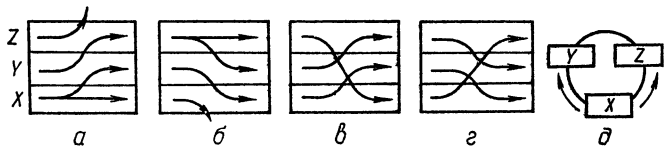


Рис. 5

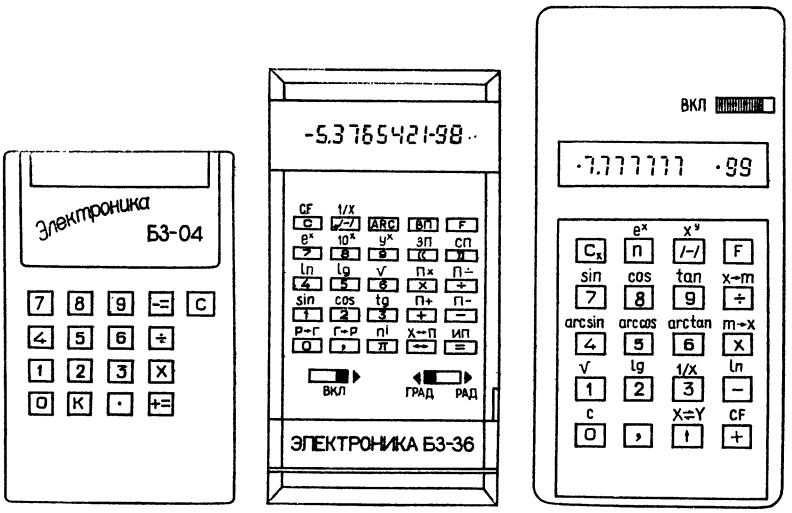
Выполнение определенных последовательностей микроопераций над содержимым регистров X и Y, а также буферных регистров обеспечивает вычисление результатов арифметических и элементарных или специальных функций, предусмотренных в словарном запасе входного языка. Последовательности кодов микроопераций, называемые *микропрограммами*, хранятся в постоянном запоминающем устройстве *ПЗУ* (см. рис. 4).

В микрокалькуляторах с обратным синтаксисом входного языка при вводе функционального оператора нажатием клавиш управляющее устройство вызывает из *ПЗУ* соответствующую микропрограмму, в соответствии с которой *АЛУ* выполняет необходимые вычисления и засылает результат в регистр X. Аналогично выполняются одноместные функциональные операторы в микрокалькуляторах с простым алгебраическим синтаксисом входного языка.

Большинство простейших и некоторые специализированные микрокалькуляторы отличаются входными языками с простым бесскобочным алгебраическим синтаксисом. В таких микрокалькуляторах имеется только два операционных регистра X и Y и один регистр *КОП* для хранения кодов двухместных операторов. При вводе первого двухместного оператора его код заносится в регистр *КОП*, а при вводе следующих их коды замещают в регистре *КОП* предыдущий, который засылается в *ПЗУ* для вызова соответствующей микропрограммы в *АЛУ*. Однако входные языки микрокалькуляторов с алгебраическим синтаксисом отличаются различными «жаргонными» особенностями, связанными с учетом старшинства двухместных операций, способами выполнения операций с постоянным операндом (константой), моментом поступле-

ния из *УУ* сигнала для смещения операционного стека «вверх» при вводе операторов и операндов и блокировками таких сигналов.

Первый отечественный массовый карманный арифметический микрокалькулятор типа «Электроника Б3-04» (рис. 6,*а*) отличается входным языком с простым бесскобочным алгебраическим синтаксисом и минимальным словарным запасом с оператором константы К и совмещенными операторами + = и — =. При вводе оператора К содержимое регистров *Y* и *КОП* фиксируется после выполнения первого функционального оператора и при последующих вводах операторов += или —= (определяющих знак результата) над содержимым ре-



а *б* *в*

Рис. 6

гистров *X* и *Y* выполняется операция, код которой хранится в регистре КОП, причем в качестве первого операнда используется содержимое регистра *X*, например, С К 10 ÷ 2 += (5) 3 + = (1,5) 4 += (2). Эту особенность можно, в частности, использовать для возведения содержимого регистра *X* в целую положительную или отрицательную степень предложениями С К *x* × += (x^2) += (x^3) += ... или С К *x* ÷ += (1) += (x^{-1}) += (x^{-2}) += ... Рассматриваемый микрокалькулятор отличается и рядом других особенностей (например, сложение не выполняется перед умножением), не представляющих интереса в настоящее время.

В современных микрокалькуляторах с алгебраическим синтаксисом входного языка для выполнения операций с константой достаточно повторно вводить оператор =, так как в этом случае содержимое регистров *Y* и *КОП* фиксируется и в двухместную операцию (код которой хранится в регистре *КОП*) первым вводится содержимое регистра *X*. Эта и другие «жаргонные» особенности выполнения комбинаций операторов, не допустимые в обычной записи формул, могут оказаться

полезными при вычислениях. Поэтому пользователю микрокалькулятора конкретного типа следует установить их экспериментально, определяя реакции микрокалькулятора на ввод различных комбинаций операторов по изменениям содержимого операционных регистров.

Некоторые простейшие (например, «Электроника Б3-35») и большинство инженерных микрокалькуляторов характеризуются простым скобочным алгебраическим синтаксисом входного языка, обеспечивающим выполнение операций со скобками. Операционные стеки таких микрокалькуляторов содержат дополнительные регистры, число которых равно допустимому числу открывающих скобок, не разделенных в программе операторами закрывающих скобок. Регистр КОП в этом случае заменяется стеком с таким же числом дополнительных регистров.

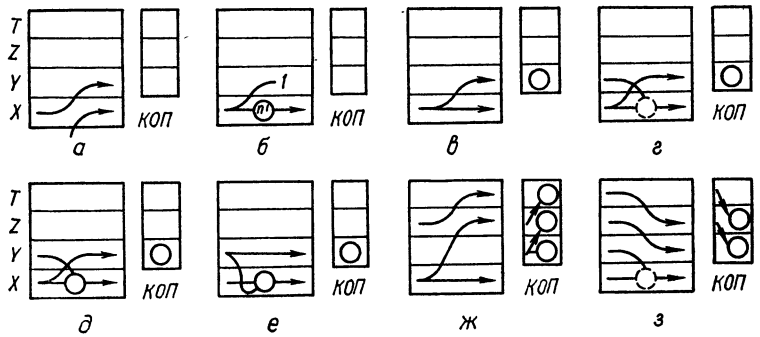


Рис. 7

Примером может служить инженерный микрокалькулятор «Электроника Б3-36» (рис. 6,*б*), допускающий вычисления с двойными скобками и, следовательно, содержащий четырехрегистровый операционный стек и трехрегистровый стек *КОП*. Проверка работы этого микрокалькулятора с помощью тестовых программ показывает, что она подчинена следующим правилам:

1. При наборе или вызове из памяти операнда в начале программы, после оператора = и после открывающей скобки его код заносится в регистр *X* без изменения содержимого остальных регистров. При вводе после выполнения двухместной операции операнд заносится в регистр *X*, прежнее содержимое которого засылается в регистр *Y* (рис. 7,*а*).

2. При вводе одноместного оператора результат операции над содержимым регистра *X* заносится в этот же регистр без изменения содержимого остальных регистров операционного стека. Исключение составляет оператор *n*! вычисления факториала, после выполнения которого прежнее содержимое регистра *Y* замещается единицей (рис. 7,*б*).

3. При вводе первого двухместного оператора в начале программы, после открывающей скобки и оператора = код вводимого оператора замещает прежнее содержимое входного регистра без изменения со-

держимого остальных регистров стека *КОП*, а содержимое регистра *X*, засылается также в регистр *Y* (рис. 7,*в*).

4. При вводе двухместного оператора после двухместного оператора или после двухместного оператора и операнда код вводимого оператора заносится во входной регистр стека *КОП* без изменения содержимого остальных его регистров, выполняется предыдущая двухместная операция (код которой до этого хранился во входном регистре стека *КОП*), результат заносится в регистр *X*, прежнее содержимое которого засылается в регистр *Y* (рис. 7,*г*).

5. При вводе оператора = после двухместного оператора и двухместного оператора и операнда выполняется двухместная операция, код которой хранится во входном регистре стека *КОП*, результат операции заносится в регистр *X*, прежнее содержимое которого засылается в регистр *Y* (рис. 7,*д*).

6. При вводе оператора = после оператора = выполняется двухместная операция, код которой хранится во входном регистре стека *КОП*, над содержимым регистров *X*, вводимым в качестве первого операнда, и *Y*, результат заносится в регистр *X* без изменения содержимого остальных регистров операционного стека (рис. 7, *е*).

7. При вводе оператора открывающей скобки содержимое регистров стека *КОП* и содержимое регистров *X*, *Z* и *T* операционного стека смещается «вверх» без изменения содержимого регистра *Y* (рис. 7,*ж*).

8. При вводе оператора закрывающей скобки выполняется последняя операция в скобках (код которой хранится во входном регистре *КОП*) над содержимым регистров *X* и *Y*, операционный стек и стек *КОП* смещаются «вниз» и результат операции заносится в регистр *X* (рис. 7,*з*).

Изменение содержимого регистров операционного стека и стека *КОП* в соответствии с этими правилами при вычислениях по несложной формуле показано в табл. 1А.

Входные языки большинства непрограммируемых микрокалькуляторов характеризуются алгебраическим синтаксисом. К немногим исключениям относится инженерный микрокалькулятор типа «Электроника Б3-19М» (см. рис. 6,*в*) с обратным синтаксисом входного языка, трехрегистровым операционным стеком и индикатором с семью цифровыми знакоместами для высвечивания мантиссы.

Правила работы микрокалькулятора при вводе операторов и операндов показаны на эквивалентных схемах (рис. 8). После выполнения двухместного оператора X^y результат вычислений заносится в регистр *X*, прежнее содержимое регистра *Y* засылается в регистр *Z*, а в регистр *Y* заносится число 2,302585. При выполнении одноместных операторов $1/x$ и $\sqrt{}$ содержимое регистров *Y* и *Z* не изменяется, но после выполнения остальных одноместных операторов прежнее содержимое регистра *Y* засылается также в регистр *Z*. Пренебрежение этой особенностью может привести к ошибкам при вычислениях по программам с одноместными и двухместными операторами.

Использование обратного синтаксиса входного языка упрощает конструкцию микрокалькулятора (отсутствует регистр или стек *КОП*),

1. Изменение содержимого регистров операционного стека и стека *КОП* при вычислениях по формуле $x = 2{,}5 - 2 \lg (8/5)$

А. Микрокалькулятор типа «Электроника Б3-36»

Оператор	Операционные регистры				Регистры *КОП*		
	X	*Y*	*Z*	*T*	1	2	3
2	2	0	0	0	0	0	0
,	2,	0	0	0	0	0	0
5	2,5	0	0	0	0	0	0
—	2,5	2,5	0	0	—	0	0
(	2,5	2,5	2,5	0	—	—	0
2	2	2,5	2,5	0	—	—	0
×	2	2	2,5	0	×	—	0
(	2	2	2	2,5	×	×	—
8	8	2	2	2,5	×	×	—
÷	8	8	2	2,5	÷	×	—
5	5	8	2	2,5	÷	×	—
)	1,6	2	2,5	0	×	—	0
lg	0,20412	2	2,5	0	×	—	0
)	0,40824	2,5	0	0	—	0	0
=	2,09176	0,40824	0	0	—	0	0

Б. Микрокалькулятор типа «Электроника Б3-19М»

Оператор	Операционный стек		
	X	*Y*	*Z*
2	2	0	0
	2,	0	0
5	2,5	0	0
↑	2,5	2,5	0
8	8	2,5	0
↑	8	8	2,5
5	5	8	2,5
÷	1,6	2,5	2,5
lg	0,2041199	2,5	2,5
2	2	0,2041199	2,5
×	0,4082398	2,5	2,5
—	2,09176	2,5	2,5

а данные табл. 2Б свидетельствуют об уменьшении числа операторов программы, что ускоряет вычисления.

Непрограммируемые микрокалькуляторы удобны для вычисления функций и результатов несложных последовательностей операций, но

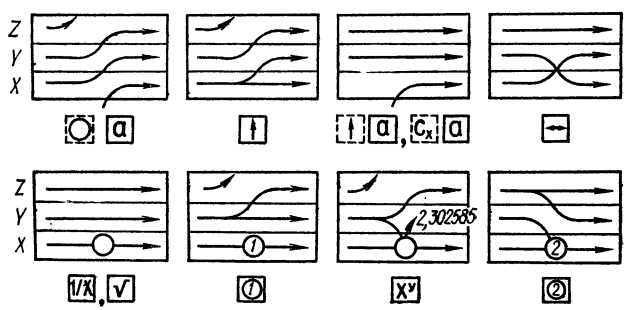


Рис. 8

их использование для сложных расчетов связано со значительными затратами времени и утомительно для пользователя, так как при увеличении числа операторов возрастает вероятность их ошибочного ввода, который не всегда удается своевременно обнаружить. Поэтому в дальнейшем будем рассматривать только использование программируемых микрокалькуляторов, удобных для простых вычислений в обычном режиме и более сложных инженерных расчетов в программируемом режиме.

4. ОСОБЕННОСТИ ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРОВ

Программируемые микрокалькуляторы отличаются от непрограммируемых *программной памятью* (запоминающим устройством для хранения программы автоматических вычислений), числовой памятью с относительно большим количеством регистров для хранения исходных данных и результатов промежуточных вычислений, а также различными вспомогательными устройствами для ввода программы автоматических вычислений в программную память, проверки правильности ввода программы и ее выполнения в автоматическом (программируемом) режиме. Основные из этих устройств обозначены на обобщенной схеме программируемого микрокалькулятора, показанной на рис. 9.

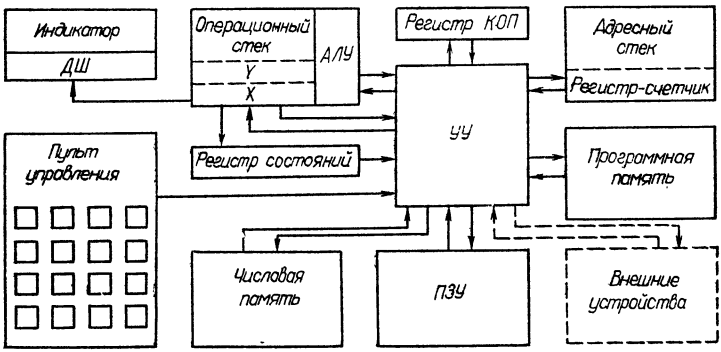


Рис. 9

В обычном режиме вычислений при нажатии клавиш код соответствующего оператора в виде электрического сигнала засылается управляющим устройством в регистр* *КОП* для идентификации. После нее код оператора набора десятичного знака пересылается в регистр *X* и на индикаторе высвечивается набранный знак. Аналогично пересылаются в соответствующие устройства для исполнения коды синтаксических операторов и операторов обращения к памяти. После ввода функционального оператора и его идентификации из *ПЗУ* вызывается и пересылается для исполнения в *АЛУ* нужная микропрограмма.

Автоматическое выполнение программы вычислений в основном обеспечивается программной памятью. При емкости не более 99 программных шагов (операторы программы занимают по одному шагу, кроме операторов условного и безусловного переходов, у которых первый шаг занимает команда проверки условия или безусловного перехода, а второй — указатель перехода) эта память состоит из отдельных устройств (ячеек) для временного хранения шагов программы и обычно имеет матричную структуру. Десятичные коды порядковых номеров (адреса) ячеек определяются номерами строк и столбцов матрицы, на

---

* В микрокалькуляторах с алгебраическим синтаксисом входного языка регистр *КОП* заменяется стеком, но в дальнейшем при отсутствии оговорок будем рассматривать микрокалькуляторы с обратным синтаксисом входного языка.

пересечении которых находится ячейка. Двузначный* десятичный код ab адреса ячейки образуется номером $a = 0, 1, 2, \ldots$ страницы (строки или столбца, в зависимости от порядка нумерации ячеек) и номером $b = 0, 1, 2, \ldots$ ячейки (столбца или строки) на этой странице. Если страницы содержат $m < 10$ ячеек, то последовательность адресов определяется в системе счисления кода b с основанием m и после адреса a $(m - 1)$ следует адрес $(a + 1)0$.

Работа программной памяти связана с внутренним адресным стеком (стеком возврата), входным регистром которого является регистр-счетчик адресов шагов программы. Содержимое этого регистра определяет ячейку программной памяти, связанную через управляющее устройство с регистром *КОП*. В рабочем режиме содержимое регистра-счетчика очищается (становится равным 00) при нажатии клавиши В/О**, а при нажатии клавиш БП a b в этот регистр засылается адрес ab.

В режиме программирования ввод шага программы нажатиями клавиш приводит к засылке его кода через регистр *КОП* в ячейку программной памяти, адрес которой равен содержимому регистра-счетчика, после чего это содержимое увеличивается на единицу и код следующего шага вводимой программы засылается в следующую ячейку. Если программная память заполнена, то ввод избыточного оператора приводит к формированию в регистре-счетчике адреса 00 и код этого оператора занимает место ранее введенного первого шага программы.

Для контроля правильности ввода программы на индикаторе в режиме программирования высвечиваются десятичные коды одного или нескольких шагов программы, введенных последними, и содержимое регистра-счетчика, на единицу большее адреса введенного последним шага программы. Редактирование вводимой программы обеспечивается внешними (вводимыми нажатиями клавиш) командами $\overrightarrow{\text{ШГ}}$ и $\overleftarrow{\text{ШГ}}$, изменяющими на единицу содержимое регистра-счетчика и, соответственно, высвечиваемые коды. Для вызова на индикатор кода нужного оператора из программной памяти можно также нажать клавиши В/О или | БП | a | b |, перейдя для этого в рабочий режим. Лишние операторы введенной в память программы устраняют вводом на их место оператора НОП, а в некоторых микрокалькуляторах предусматривают специальные команды для перемещения на заданное число шагов отдельных частей программы в программной памяти.

После ввода программы микрокалькулятор переводят в рабочий режим, вводят исходные данные в числовую память или операционный стек, а затем нажимают клавиши В/О или | БП | a | b |, если выполнение программы должно начинаться с адреса 00 или ab соответственно.

---

* В микрокалькуляторах с емкостью программной памяти от 99 до 999 шагов используют трехзначные адреса, а при большей емкости программной памяти она обычно не разбивается на отдельные ячейки и передача управления реализуется с помощью меток.

** Здесь и далее команды и операторы (за исключением операторов флагов) обозначены символами, принятыми для отечественных микрокалькуляторов.

Конструкцией программируемых микрокалькуляторов обычно предусмотрена возможность проверки выполнения программы по шагам нажатиями в рабочем режиме клавиши *ПП*, каждое из которых приводит к увеличению на единицу содержимого регистра-счетчика, вызову и выполнению соответствующего шага программы.

Для выполнения программы в автоматическом (программируемом) режиме нажимают клавишу С/П, что приводит к вызову в регистр *КОП* и исполнению шага программы, адрес которого равен исходному содержимому регистра-счетчика, автоматически увеличивающемуся на единицу после вызова этого и последующих шагов программы из программной памяти. При вызове в регистр *КОП* кода оператора С/П (в некоторых микрокалькуляторах после считывания последнего шага программы) содержимое регистра-счетчика фиксируется на адресе этого оператора, выполнение программы прекращается и на индикаторе высвечивается (или хранится в памяти, если это предусмотрено программой) результат вычислений.

Таким образом, выполнение неразветвленных программ в обычном и программируемом режимах отличается лишь тем, что коды шагов программы засылаются в регистр *КОП* с пульта управления или программной памяти. При выполнении разветвленной программы считывание из программной памяти кода команды БП безусловного перехода и его идентификация в регистре *КОП* приводит к засылке считываемого следующим адреса перехода в регистр-счетчик. В этом случае выполнение программы будет продолжаться с оператора, код которого хранится в программной памяти по адресу перехода, заслан-ному в регистр-счетчик.

При считывании из программной памяти команды ПП безусловного перехода к подпрограмме* считываемый следующим адрес перехода также засылается в регистр-счетчик, но в этом случае адресный стек смещается «вверх» и предыдущее содержимое регистра-счетчика, равное адресу указателя (адреса) перехода, засылается во второй регистр адресного стека. После этого выполнение программы также продолжается с шага подпрограммы, адрес которого заслан в регистр-счетчик.

После выполнения подпрограммы и считывания кода оператора возврата В/О из программной памяти адресный стек смещается вниз, причем к новому содержимому регистра-счетчика (ранее хранимому во втором регистре стека) добавляется единица и выполнение программы продолжается по этому адресу с шага, следующего за соответствующим оператором обращения к подпрограмме.

Число регистров адресного стека определяет допустимое число обращений из подпрограммы к подпрограмме (число *вложений* подпрограмм), так как при каждом таком обращении адресный стек смещается «вверх» и при избыточном числе вложений подпрограмм стек переполняется. Для иллюстрации переходов при вложении подпро-

---

* Подпрограмма оканчивается оператором В/О и может быть помещена в любой части программы, хотя чаще всего подпрограммы размещают после оператора С/П.

грамм рассмотрим часть операторов некоторой программы с указанными в скобках адресами шагов:

... ПП 25 ..... С/П ..... ПП 54 ... В/О ..... В/О
... (05)(06)(07) ..... (25) ... (36)(37)(38) ... (54) ...

При пуске нажатиями клавиш В/О и С/П такая программа будет выполнена с начального оператора до первого оператора ПП 25 обращения к подпрограмме, что приведет к засылке адреса 25 в регистр-счетчик, прежнее содержимое которого 06 сместится во второй регистр адресного стека и выполнение программы продолжится с шага, код которого хранится по адресу 25 в программной памяти. При считывании второго оператора ПП следующий за ним указатель перехода 54 будет заслан в регистр-счетчик, содержимое 37 которого сместится во второй регистр адресного стека, а предыдущее содержимое 06 второго регистра стека сместится в третий регистр. Выполнение программы продолжится с оператора, код которого хранится по адресу 54 до считывания кода оператора В/О. При считывании кода оператора В/О адресный стек сместится «вниз» и в его входной регистр-счетчик будет заслан увеличенный на единицу адрес $38 = 37 + 1$, с которого и продолжится выполнение программы. При считывании следующего кода оператора В/О адресный стек вновь сместится «вниз» и в регистр-счетчик будет заслан адрес $07 = 06 + 1$, с которого и продолжится выполнение программы до считывания оператора С/П.

Рассмотренная организация работы адресного стека обеспечивает возможность выполнения подпрограммы с любого ее шага. Более того, если в программе после оператора обращения к подпрограмме не записан оператор В/О, то первый из них можно использовать в качестве оператора безусловного перехода к любому последующему шагу программы. В свою очередь, если в программе перед оператором В/О не записан оператор перехода к подпрограмме, то после считывания оператора В/О в программируемом режиме выполнение программы будет продолжаться со второго шага с адресом 01.

Для конструктивной реализации операторов условных переходов вида *xAz* процессор микрокалькулятора дополняют *регистром состояний*, соединенным с регистром X операционного стека. В процессе выполнения программы в регистр состояний непрерывно заносится информация о знаке или равенстве нулю содержимого регистра X, для чего достаточно двух двухпозиционных (0 или 1) элементов регистра состояний, которые могут находиться в состояниях 00 ($x = 0$), 10 ($x > 0$) и 01 ($x < 0$). По состояниям каждого из этих элементов $E1$ и $E2$ и их сумме Σ можно определить выполнение условий $x = 0$ ($\Sigma = 0$), $x \neq 0$ ($\Sigma \neq 0$), $x > 0$ ($E1 = 1$), $x \leqslant 0$ ($E1 = 0$), $x < 0$ ($E2 = 1$) и $x \geqslant 0$ ($E2 = 0$).

Если входной язык микрокалькулятора содержит условные операторы с проверкой условий xAy или xAt, то регистр состояний соответственно может быть дополнен двухпозиционными ячейками для проверки знака или равенства нулю значений $x - y$ или $x - t$. Регистр состояний может быть также использован для контроля переполнения — если значение мантиссы или порядка содержимого регистра X

превышает допустимое, то в соответствующей ячейке регистра состояний нуль заменяется единицей и вырабатывается сигнал переполнения, подаваемый на индикатор.

При считывании из программной памяти кода команды *xAz* проверки условия устройство управления проверяет содержимое соответствующих элементов регистра состояния. Если проверяемое условие выполнено (в некоторых микрокалькуляторах — при невыполнении условия), то считываемый следующим электрический сигнал, кодирующий указатель перехода, блокируется и выполнение программы продолжается с оператора, записанного после оператора условного перехода. Если проверяемое условие не выполнено (выполнено), то считываемый после команды *xAz* код указателя перехода заносится в адресный регистр-счетчик и выполнение программы продолжается с оператора, код которого хранится по адресу, равному указателю перехода.

Значительно проще реализуются условные операторы проверки флага Ф *n ab* или IF FG *n ab*, где *n* — номер флага. В микрокалькуляторах, входной язык которых содержит подобные операторы, имеется специальный *регистр флагов*, число двоичных элементов которого равно числу флагов. При считывании из программной памяти кода оператора SФ *n* или STFG *n* установки *n*-го флага управляющее устройство изменяет состояние 0 соответствующего *n*-го элемента регистра флагов на состояние 1. При считывании кода оператора СФ *n* или CLFG *n* очистки *n*-го флага управляющее устройство изменяет состояние 1 соответствующего *n*-го элемента регистра флагов на состояние 0. При считывании оператора Ф *n a b* или IF FG *n ab* проверки флага управляющее устройство проверяет состояние *n*-го элемента регистра флага — если флаг установлен, то считываемый код адреса перехода блокируется и следующим выполняется оператор, записанный в программе после оператора проверки флага; если флаг не установлен (в некоторых входных языках —установлен), то считываемый адрес перехода засылается в адресный регистр-счетчик и выполнение программы продолжается с оператора, код которого хранится по этому адресу.

В микрокалькуляторах, входной язык которых допускает косвенную адресацию, предусмотрена возможность засылки в адресный регистр-счетчик не только адресов переходов из регистра *КОП*, но и содержимого любого регистра числовой памяти. В этом случае при считывании из программной памяти в регистр *КОП* кода оператора косвенного безусловного перехода КБПN или КПППN, занимающего один шаг программы, управляющее устройство засылает в адресный регистр-счетчик содержимое регистра *N* числовой памяти и выполнение программы продолжается с оператора, код которого хранится в программной памяти по адресу, равному содержимому регистра *N* числовой памяти. При этом считывание оператора КПППN приводит также к смещению «вверх» адресного стека и адрес оператора КПППN заносится в его второй регистр.

При считывании из программной памяти кода оператора К*x*A*z*N оператора условного косвенного перехода, занимающего одну ячейку программной памяти, управляющее устройство проверяет содержимое соответствующих элементов регистра состояний и при невыполнении

(выполнении) проверяемого условия целая часть содержимого регистра N числовой памяти засылается в адресный регистр-счетчик и выполнение программы продолжается с шага, адрес которого равен этой целой части. При выполнении (невыполнении) проверяемого условия содержимое регистра N не засылается в регистр-счетчик и выполнение программы продолжается с шага, следующего за оператором КxAzN.

При считывании оператора КПN косвенной засылки в память содержимое регистра X засылается в регистр числовой памяти, номер которого равен содержимому регистра N числовой памяти. При считывании оператора КИПN косвенного вызова из памяти в регистр X засылается содержимое регистра числовой памяти, номер которого хранится в регистре N. Косвенная адресация обеспечивает возможность выполнения операций над адресами переходов и номерами регистров числовой памяти. Обычно при косвенной адресации предусматривается автоматическое изменение (модификация) на единицу определенных регистров N числовой памяти при каждом выполнении оператора косвенной адресации.

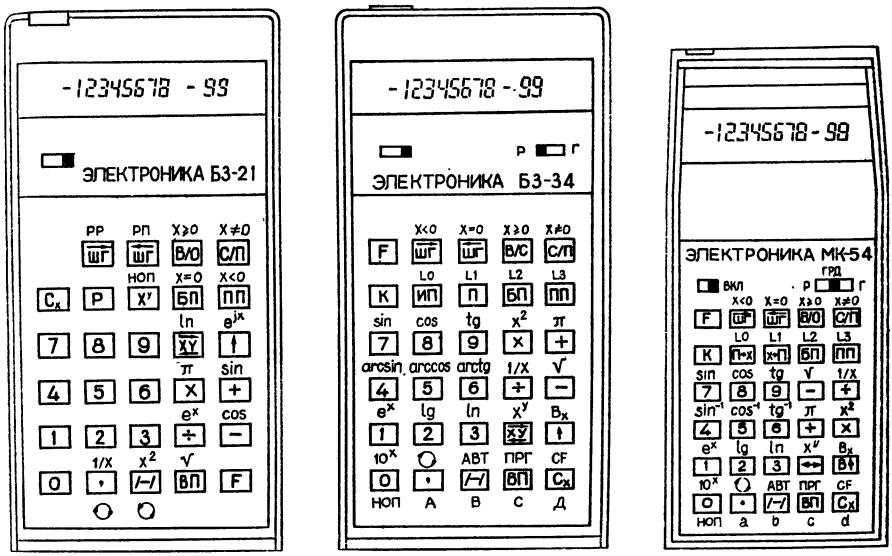
Организация управления программой может отличаться от описанной, характерной для большинства массовых программируемых микрокалькуляторов. В наиболее совершенных программируемых микрокалькуляторах, приближающихся по ряду показателей к универсальным ЭВМ второго поколения, числовая и программная память объединены, что позволяет изменять в определенных пределах распределение памяти, а для передачи управления используются метки. В таких микрокалькуляторах с емкостью памяти, обеспечивающей хранение нескольких сотен чисел и программ с несколькими тысячами шагов, организация управления программой близка к организации этого процесса в универсальных ЭВМ. Подобные микрокалькуляторы обычно снабжаются периферийными устройствами, аналогичными по назначению периферийным устройствам универсальных ЭВМ,— миниатюрными цифровыми или алфавитно-цифровыми печатающими устройствами для вывода информации на термочувствительную бумажную ленту, накопителями информации в виде магнитных карточек и полупроводниковых сменных запоминающих устройств для хранения программ вычислений или других данных и иногда имеют дополнительные программируемые устройства ввода-вывода информации (интерфейсы).

Однако такие сложные настольные и карманные микрокалькуляторы отличаются высокой стоимостью изготовления, а их программирование по сложности часто превосходит программирование универсальных ЭВМ. Поэтому в настоящее время наибольшее практическое значение имеют массовые инженерные программируемые микрокалькуляторы с емкостью программной памяти 50—250 шагов и 5—20 регистрами числовой памяти, отличающиеся небольшой стоимостью изготовления и простотой пользования.

5. МАССОВЫЕ ПРОГРАММИРУЕМЫЕ МИКРОКАЛЬКУЛЯТОРЫ

В массовых программируемых микрокалькуляторах входные языки с алгебраическим синтаксисом используются относительно редко и типичными являются входные языки с обратным синтаксисом одноместных и двухместных функциональных операций.

Первый отечественный массовый программируемый микрокалькулятор типа «Электроника Б3-21» (рис. 10,*а*) содержит программную память емкостью 60 шагов и индикатор для высвечивания чисел в естественной или стандартной (научной) показательной форме с восемью



а *б* *в*

Рис. 10

разрядами мантиссы* и двумя порядка. Числовая память состоит из запоминающего устройства с произвольной выборкой (*ЗУПВ*) из семи регистров с номерами *2*, ..., *8* (регистр *1* используется в качестве операционного регистра *Y*) и шести регистров, соединенных с регистром *X* в кольцевой стек памяти. Содержимое регистров этого стека (которые будем обозначать номерами CN согласно схеме, показанной на рис. 11) смещается по или против часовой стрелки при вводе соответственно операторов → или ← поворота стека памяти**. Засылка

* В большинстве микрокалькуляторов этого типа запятая высвечивается в цифровых разрядах и поэтому индицируется не более семи цифр мантиссы с дробной частью. В этом случае для определения восьмой цифры мантиссы достаточно вычесть содержимое старшего разряда. Например, при вызове π на индикаторе высвечивается 3,141592, но при вычитании числа 3 индицируется число $1{,}415927 \cdot 10^{-1}$ с искомой цифрой.

** Здесь и далее для упрощения записи операторы поворота стека, обозначаемые на клавиатуре (см. рис. 10,*а*) окружностями со стрелками, отображаются упрощенными символами → и ←. Аналогично оператор обмена содержимым регистров *X* и *Y* будем обозначать упрощенным символом XY без стрелок.

содержимого регистра X в регистр *ЗУПВ* с номером N обеспечивается вводом операторов PN, а вызов содержимого N-го регистра *ЗУПВ*— вводом операторов FN.

Использование кольцевого стека памяти во многих случаях позволяет существенно сократить длину программы, но иногда невозможность прямого вызова содержимого внутренних регистров этого стека затрудняет полное использование емкости числовой памяти.

Работа операционного стека рассматриваемого микрокалькулятора, образованного регистрами X и Y, подчинена простым и удобным правилам, связанным с обратным синтаксисом входного языка. При вводе оператора ↑ или оператора P1 содержимое операционного стека смещается «вверх» (рис. 12,*а*). При наборе операнда после любого оператора, кроме операторов ↑ и Cx, содержимое стека смещается

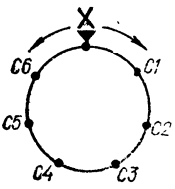


Рис. 11

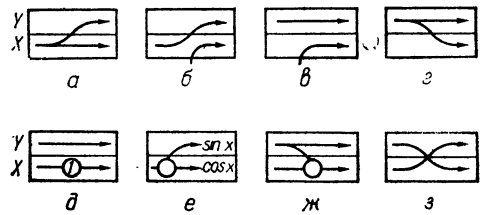


Рис. 12

«вверх», а набранный операнд заносится в регистр X (рис. 12,*б*). Операнд, вызываемый из памяти или набираемый после операторов ↑ или Cx, заносится в регистр X без изменения содержимого регистра Y (рис. 12,*в*), а при вводе оператора F1 операционный стек смещается «вниз» (рис. 12,*г*).

При вводе одноместных операторов ln, e^x, sin и cos (с предварительным нажатием префиксной клавиши P) или операторов $1/x$, x^2 и $\sqrt{}$ (с предварительным нажатием префиксной клавиши F) результат вычислений заносится в регистр X без изменения содержимого регистра Y (рис. 12,*д*). После выполнения одноместного оператора e^{jx}, вводимого с предварительным нажатием клавиши P, в регистрах Y и X хранятся соответственно значения $\sin x$ и $\cos x$ (рис. 12,*е*). В этом случае для вычисления значения $\operatorname{tg} x$ достаточно дополнительно нажать клавишу ÷ (аргумент тригонометрических функций должен быть выражен в радианах). После выполнения двухместных операторов, вводимых без нажатия префиксных клавиш, содержимое регистра Y сохраняется (рис. 12,*ж*), а после выполнения синтаксического оператора XY обмена операндов содержимое регистров X и Y меняется местами (рис. 12,*з*).

К операторам управления выполнением программы относятся операторы безусловного перехода и обращения к подпрограммам и условные операторы, занимающие два шага программы (первый шаг — команда БП или ПП или команда проверки условия $x = 0$, $x < 0$, $x \geqslant 0$, $x \neq 0$, второй шаг — указатель перехода по заданному адресу, набираемый нажатиями определенных клавиш), а также оператор В/О возврата и оператор С/П автоматического прекращения

выполнения программы, занимающие по одному шагу. Переход от условных операторов по заданному адресу реализуется при невыполнении проверяемого условия.

Переход в режим программирования и обратный переход в рабочий режим обеспечивают соответственно команды РП и РР, вводимые с нажатием префиксной клавиши Р. При попадании результата вычислений в область машинной бесконечности и попытке выполнения некорректной операции на индикаторе высвечивается символ переполнения — нули перед мантиссой или отрицательное значение нулевого порядка.

Программная память разбита на 10 страниц по 6 ячеек с адресами ab, образованными номерами $a = 0, 1, ..., 9$ страниц и номерами $b = 0, 1, ..., 5$ ячеек на странице. Адресный стек содержит 5 дополнительных (кроме регистра-счетчика) регистров, что обеспечивает до пяти вложений подпрограмм (внутренних обращений от подпрограммы к подпрограмме).

В режиме программирования для проверки правильности ввода программы на индикаторе в знакоместах порядка высвечивается содержимое регистра-счетчика шагов, равное адресу следующего шага программы, а в знакоместах мантиссы — десятичные коды команд или указателей перехода, занимающих три последние шага введенного фрагмента программы. При вводе указателей переходов нажатиями определенных клавиш (табл. 2) высвечиваются коды ab' указателей,

2. Коды адресов и операторов входного языка микрокалькулятора «Электроника Б3-21»

Код	Адрес	Оператор	Клавиши	Код	Адрес	Оператор	Клавиши	Код	Адрес	Оператор	Клавиши
01	00		Р 0	38		X^y	X^y	66	65	ВП	ВП
02	01		F 0	39		НОП	Р X^y	68		ПП	ПП
03	02	e^{ix}	Р ↑	41	40	Р4	Р 4	69		$x<0$	Р ПП
04	03	0	0	42	41	F4	F 4	71	70	Р7	Р 7
05	04		F ↑	43	42	→	Р ,	72	71	F7	F 7
06	05	↑	↑	44	43	4	4	73	72		Р Сх
11	10	Р1	Р 1	45	44	$1/x$	F ,	74	73	7	7
12	11	F1	F 1	46	45	,	,	75	74		F Сх
13	12		Р ХУ	48		В/О	В/О	76	75	Сх	Сх
14	13	1	1	49		$x \geqslant 0$	Р В/О	78		С/П	С/П
15	14		F ХУ	51	50	Р5	Р 5	79		$x \neq 0$	Р С/П
16	15	ХУ	ХУ	52	51	F5	F 5	81	80	Р8	Р 8
21	20	Р2	Р 2	53	52	←	Р /—/	82	81	F8	F 8
22	21	F2	F 2	54	53	5	5	83	82	cos	Р —
23	22	π	Р ×	55	54	x^2	F /—/	84	83	8	8
24	23	2	2	56	55	/—/	/—/	85	84		F —
25	24		F ×	58		БП	БП	86	85	—	—
26	25	×	×	59		$x=0$	Р БП	91	90		Р 9
31	30	Р3	Р 3	61	60	Р6	Р 6	92	91		F 9
32	31	F3	F 3	62	61	F6	F 6	93	92	sin	Р +
33	32	e^x	Р ÷	63	62		Р ВП	94	93	9	9
34	33	3	3	64	63	6	6	95	94		F +
35	34		F ÷	65	64	$\sqrt{}$	F ВП	96	95	+	+
36	35	÷	÷								

на единицу бóльшие ($b' = 1, 2, \ldots, 6$) адресов перехода* и сóвпадающие с кодами команд, вводимых нажатиями тех же клавиш. Это не приводит к недоразумениям, так как указатели переходов записаны в программе только после команд проверки условия, безусловного перехода и обращения к подпрограмме. Например, высвечивание на индикаторе в режиме программирования кодов

22	22	59	50

в соответствии с данными табл. 2 означает, что введено 30 шагов программы (следующий шаг будет иметь адрес 50), последними введены команда проверки условия $x = 0$ с кодом 59 по адресу 43, указатель перехода 22 по адресу 21 адрес указателя 44) и оператор F2 с кодом 22, занимающий шаг программы с адресом 45.

После перехода в рабочий режим, занесения исходных данных в операционные регистры и числовую память и, при необходимости, пошаговой отладке программы (с нажатием клавиши ПП) для пуска программы с начального шага нажимают клавишу В/О (очищается регистр-счетчик шагов) и клавишу С/П (регистр-счетчик запускается с адреса 00). Для выполнения программы с шага по адресу *ab* нажимают клавишу БП, клавиши набора адреса *ab* в соответствии с табл. 3 (адрес *ab* заносится в регистр-счетчик) и клавишу С/П (регистр-счетчик запускается с адреса *ab*). При необходимости аварийной остановки выполняемой программы нажимают клавишу С/П, что приводит (как и нажатие некоторых других клавиш) к остановке регистра-счетчика и высвечиванию на индикаторе содержимого регистра X.

В рассматриваемом микрокалькуляторе (см. рис. 10,*а*) полный цикл адресного регистра-счетчика составляет 96 шагов, что соответствует (в десятично-шестеричной системе счисления адресов) максимальному адресу 155, после чего регистр-счетчик автоматически счищается и формирование адресов продолжается с адреса 00. Если включить микрокалькулятори нажать клавишу С/П, запускающую регистр-счетчик в рабочем режиме, то управляющее устройство последовательно опросит ячейки программной памяти с адресами от 00 до 95, после чего будет сформирован адрес 100 (высвечиваемый на индикаторе как —0), который управляющее устройство по содержимому двух младших разрядов воспримет как 00, и опрос программной памяти продолжится с начальной ячейки. Однако после считывания содержимого ячейки с адресом 55 (соответствующего концу предельного цикла в 96 шагов) содержимое регистра-счетчика очищается и опрос продолжается, начиная снова с адреса 00 и до окончания цикла с последующими его повторениями.

---

* Высвечиваемые коды *ab'* пользователь может условно рассматривать как адреса 01, 02, ..., 95, —0 ячеек программной памяти (шагов программы),содержимому которых передается управление. В этом случае [11] высвечиваемое содержимое регистра-счетчика следует рассматривать как адрес шага, введенного последним, причем адреса вида *а*6 высвечиваются как $(a + 1)0$, например, адрес 26 высвечивается как 30, а 96 как —0.

Следовательно, при повторном пуске программы нажатием только клавиши С/П регистр-счетчик шагов будет запущен с адреса ячейки, в которой хранится код оператора С/П и, если он больше адреса 55, выполнение первых 13 шагов неразветвленной программы (при выполнении переходов содержимое регистра-счетчика изменяется) повторится дважды, что может привести к ошибочному результату выполнения программы.

При составлении и выполнении программ вычислений необходимо учитывать, что работа микрокалькуляторов первых выпусков с входным языком ЯМК21 отличается следующими особенностями:

1. При сложении числа, содержащего в мантиссе семь девяток и более четырех единиц в восьмом (неиндицируемом) для дробных мантисс разряде, и большего по порядку числа возникает ошибка. Для проверки микрокалькулятора следует сложить 9,9999999 и 10 — если высвечивается 120 (при вычитании 100), то при вычислениях необходимо учитывать эту особенность.

2. При выполнении одноместных операторов вычисления элементарных функций и оператора X^y в разряды порядка одного из регистров (номер которого зависит от вида функции и значений аргумента [11]) кольцевого стека числовой памяти заносится сигнал переполнения, что приводит к ошибкам при использовании этого регистра для хранения операндов. Для проверки микрокалькулятора следует очистить память, выключив питание, и после включения питания нажать клавиши | 2 | Р | + | Р |, |. Если высвечивается 0, то микрокалькулятор свободен от рассматриваемого недостатка, в противном случае (высвечивается 0—00) при вычислении функций между обращениями к стеку памяти следует учитывать его переполняемые регистры или не использовать стек памяти.

3. Переход к подпрограмме не выполняется, если код команды БП занесен в ячейку программной памяти с адресом 55, 65, 70, 80, 91 или 92, и выполняется оператор, код которого равен указателю перехода к подпрограмме. Для проверки следует в рабочем режиме нажать клавиши БП /—/ Р $\overleftarrow{\text{ШГ}}$ ПП 9 9 С/П Р $\overrightarrow{\text{ШГ}}$ БП 9 Р $\overleftarrow{\text{ШГ}}$, 1 1 С/П Р $\overrightarrow{\text{ШГ}}$ БП 5 С/П. Если на индикаторе высвечивается 11 то переход к подпрограмме выполняется нормально (проверяется ячейка с адресом 55), но при высвечивании 99 переход к подпрограмме с указанных адресов не выполняется.

В последующих выпусках микрокалькулятора эти недостатки устранены, но следует учитывать возможность следующих особенностей работы:

1. Разность чисел, отличающихся по порядку более чем на семь единиц, меньше большего числа на единицу последнего разряда мантиссы (например, $10-10^{-20}= 9{,}9999999$).

2. При считывании команды проверки условия $x \geqslant 0$ или $x < 0$ содержимое —0 регистра X воспринимается управляющим устройством как отрицательное число. Так как представление —0 может появиться только после выполнения оператора изменения знака /—/, то следует избегать ввода этого оператора перед операторами перехода по усло-

виям $x \geqslant 0$ и $x < 0$. Заметим, что представление —0 принимает форму 0 после ввода операторов ↑, FO, PN, XY и некоторых других.

Быстродействие микрокалькулятора зависит от выполняемых операторов, значений операндов и характеристик компонентов, причем время выполнения операций значительно возрастает при нулевых значениях операндов. Время выполнения отдельных операций микрокалькулятором рассматриваемого типа может отличаться на 20—40 % от указанного в пособии [13] в зависимости от характеристик компонентов. В частности, время выполнения операций на микрокалькуляторе с индикатором на светодиодах (красное свечение) примерно на 20 % меньше, чем при индикаторе с газонаполненными сегментами с зеленым свечением [19].

Элементная база и, соответственно, входной язык микрокалькулятора типа «Электроника Б3-21» использованы в программируемых микрокалькуляторах других типов (например, в настольном микрокалькуляторе типа «Электроника МК-46»). Поэтому в дальнейшем для краткости входные языки таких микрокалькуляторов будем называть входным языком ЯМК21.

Входным языком, также имеющим обратный синтаксис, с бóльшим словарным запасом и разнообразием грамматических правил характеризуется программируемый микрокалькулятор типа «Электроника Б3-34» (см. рис. 10,*б*), индикатор которого содержит 8 цифровых знакомест мантиссы и 2 порядка с высвечиванием результата вычислений в естественной или стандартной (научной) показательной форме.

Числовая память этого микрокалькулятора содержит 14 регистров с произвольной выборкой, обозначенных номерами $N = 0, 1, 2, \ldots, 9, A, B, C, Д$, содержимое которых вызывается в регистр X операторами ИПN, а содержимое регистра X засылается в регистры памяти операторами ПN (для набора номеров A, B, C и $Д$ нажимают клавиши, под которыми они обозначены).

Входной язык микрокалькулятора, кроме функциональных операторов входного языка ЯМК21 (исключая оператор e^{ix}), содержит функциональные операторы arcsin, arccos, arctg для вычисления обратных тригонометрических функций, операторы lg и 10^x — для вычисления десятичных логарифма и антилогарифма. Аргумент тригонометрических функций может быть задан в градусах или радианах в зависимости от установки переключателя Р—Г. Все операторы, обозначенные над клавишами (кроме оператора НОП, обозначенного под клавишей, и рассматриваемых далее операторов косвенной адресации), вводят с предварительным нажатием клавиши F, при ошибочном нажатии которой совмещенный режим снимают нажатием клавиши, над которой обозначена команда CF. Оператор НОП вводят с предварительным нажатием клавиши К.

Операционный стек рассматриваемого микрокалькулятора содержит регистры X, Y, Z и T, что обеспечивает вычисления со «скобкой в скобках» и дополнительный регистр $X1$ для хранения результата предыдущих вычислений. Работа операционного стека подчинена следующим правилам:

1. При вводе оператора ↑ содержимое стека смещается «вверх» с сохранением содержимого регистра X (рис. 13,*а*).

2. При наборе после операторов ↑ или Cx операнд заносится в регистр X без изменения содержимого других регистров (рис. 13,*б*).

3. При наборе операнда после остальных операторов и вызове из памяти содержимое стека смещается «вверх», а операнд заносится в регистр X (рис. 13,*в*).

4. При вводе оператора XY содержимое регистров X и Y меняется местами без изменения содержимого остальных регистров, кроме регистра $X1$, куда заносится прежнее содержимое регистра X (рис. 13, *г*).

5. При вводе любого одноместного функционального оператора результат операции над содержимым регистра X заносится в этот регистр, а его предыдущее содержимое засылается в регистр $X1$ (рис. 13,*д*).

6. При вводе двухместного оператора результат операции над содержимым регистров X и Y заносится в регистр X, прежнее содержимое которого засылается в регистр $X1$, а содержимое остальных регистров стека смещается «вниз» при сохранении содержимого регистра T (рис. 13,*е*) или, после выполнения оператора X^y, сохраняется.

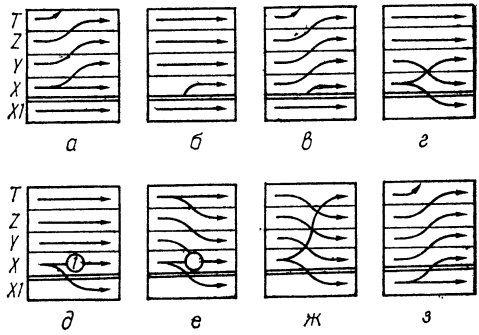


Рис. 13

7. При вводе оператора → содержимое регистра X в кольцевом режиме засылается в регистры T и $X1$, а содержимое остальных регистров смещается «вниз» (рис. 13,*ж*).

8. При вводе оператора BX содержимое стека смещается «вверх», а содержимое регистра $X1$ засылается также в регистр X (рис. 13,*з*).

Переход в режим программирования выполняется по команде ПРГ, а переход из режима программирования в рабочий режим — по команде АВТ. Отладка, редактирование и пуск программы выполняются с помощью тех же команд, что и для микрокалькулятора типа «Электроника Б3-21».

Программная память рассматриваемого микрокалькулятора содержит 98 ячеек и разбита на 10 страниц по 10 (8 на последней странице) ячеек с адресами ab = 00, 01, ..., 97, образованными в десятичной системе счисления номерами a = 0, 1, ..., 9 страницы и номерами b = 0, 1, ..., 9 ячеек на этой странице.

В программах автоматических вычислений операторы условных и безусловных (кроме В/О) переходов занимают два шага — первый шаг занимает команда безусловного перехода или проверки условия, а второй — указатель перехода в виде адреса оператора программы, которому передается управление (в условных операторах — при невыполнении проверяемого условия). В режиме программирования содержимое адресного регистра-счетчика и коды трех операторов,

введенных последними, высвечиваются на индикаторе, но коды указателей перехода равны адресам, а не отличаются от них на единицу, как в микрокалькуляторах с входным языком ЯМК21.

Входной язык рассматриваемого микрокалькулятора кроме операторов управления программой, имеющихся во входном языке ЯМК21, содержит *операторы косвенной адресации*, к которым относятся условные операторы цикла (в которых первый шаг занимает команда L0, L1, L2 или L3 проверки на равенство нулю содержимого регистра *0*, *1*, *2* или *3*, а второй — адрес перехода при невыполнении проверяемого условия) и занимающие по одному шагу операторы косвенных переходов и обращений к памяти, символы которых включают номер регистра N памяти. При каждом выполнении этих операторов содержимое регистра N модифицируется — уменьшается на единицу, если $N = 0, 1, 2$ или 3, увеличивается на единицу, если $N = 4, 5$ или 6, и не изменяется, если N — номер одного из остальных регистров памяти (регистры *A*, *B*, *C* и *Д* обозначаются соответственно номерами *10*, *11*, *12* и *13*). При модификации бо́льшего единицы дробного содержимого регистра N оно предварительно округляется до ближайшего меньшего целого числа, хранимого в младших десятичных разрядах регистра, старшие разряды которого заполняются нулями. Содержимое регистра N, меньшее единицы, не модифицируется, а соответствующий оператор в этом случае не может быть использован по прямому назначению.

Перед выполнением программы с условными операторами цикла в регистр N заносят требуемое число повторений вычислений (итераций) в цикле. При считывании команды LN управляющее устройство проверяет содержимое регистра N (уменьшающееся на единицу при каждом выполнении этой команды) по условию $x = 0$. Если это условие выполнено, то управление передается следующему оператору программы, иначе — оператору с адресом *ab*.

Операторы косвенных безусловных переходов КБПN и КППN и косвенных условных переходов Кx = 0N, К$x \neq$ ON, К$x \geqslant$ 0N, К$x <$ 0N, также вводимые нажатиями нескольких клавиш, занимают только по одному шагу программы. При каждом выполнении такого оператора в случае безусловного перехода и невыполнении проверяемого условия управление передается по адресу, равному модифицированному содержимому регистра N, а при выполнении проверяемого условия — следующему оператору.

Операторы косвенного обращения к памяти КПN и КИПN также занимают по одному шагу программы. При каждом выполнении оператора КПN косвенной засылки в память содержимое регистра X засылается также в регистр памяти, номер которого равен модифицированному содержимому регистра N (регистрам *A*, *B*, *C*, *Д* соответствуют номера *10*, *11*, *12*, *13*). При каждом выполнении оператора КИПN в регистр X засылается содержимое регистра памяти, номер которого равен модифицированному содержимому регистра N.

При попадании результата вычислений в область машинной бесконечности и попытке выполнения некорректной операции на индикаторе высвечивается символ ЕГГОГ (error — *ошибка*).

Отдельные выпуски микрокалькуляторов рассматриваемого типа отличаются особенностями*, которые необходимо учитывать при вычислениях:

1. В программируемом режиме работы первых выпусков микрокалькулятора не выполняются функциональные операторы перед оператором /—/ изменения знака. Для проверки следует в рабочем режиме последовательно нажать клавиши F ВП 8 ↑ × /—/ С/П F /—/ С/П. Если после выполнения программы высвечивается —64, то проверяемый микрокалькулятор лишен этой особенности. Высвечивание — 8 свидетельствует о том, что в программах автоматических вычислений следует избегать записи оператора /—/ непосредственно после функциональных операторов.

2. В программируемом режиме после выполнения подпрограммы, оканчивающейся некоторыми операторами, вместо оператора В/О выполняется следующий за ним оператор. Для проверки следует в рабочем режиме последовательно нажать клавиши В/О F ВП ПП 0 3 С/П 2 ↑ + lg В/О Сх С/П F /—/ В/О С/П. Если высвечивается нуль или символ переполнения, то в проверяемом микрокалькуляторе возврат от подпрограммы при ее окончании выбранным оператором (в данном случае lg) не выполняется. В этих случаях следует дополнить подпрограмму оператором (например, засылки в память), не изменяющим результата вычислений, или изменить структуру программы.

3. Оператор X^y выполняется неверно при некоторых значениях операндов и результата предыдущей операции (например, содержащего 5,7 или 9 единиц в восьмом разряде мантиссы). Для проверки следует нажать клавиши 5 5 5 5 ↑ × 4 ↑ F XY и, если высвечивается $256{,}00004 \approx 4^4$, то оператор X^y выполняется правильно. Если же высвечивается 39,062487, то следует изменить структуру программы вычислений (например, после выполнения предыдущей операции очистить регистр *X1* вводом операторов 0 и →).

4. Возникает ошибка при вычислении тригонометрических функций для значений аргумента, выраженных в градусах и превышающих 360°. Для проверки достаточно вычислить $\sin 930° = -0{,}5$. Если результат вычислений равен $-0{,}64278762 = \sin 940°$, то аргумент функции следует предварительно привести в интервал главных значений, например, вместо $\sin 930°$ вычислить $\sin (930—720°) = \sin 210° = -0{,}5$.

Следует также отметить, что в рассматриваемом микрокалькуляторе работа адресного регистра-счетчика определяется более сложными правилами, чем в микрокалькуляторах с входным языком ЯМК21. После выполнения программы без переходов (изменяющих содержимое регистра-счетчика) ее повторный пуск нажатием только клавиши С/П приводит к опросу следующих ячеек программной памяти до адреса 97, затем программа выполняется с адреса 00 до адреса 13, снова с адреса 00 до 47 и лишь после этого полностью выполняется с адреса 00. В этом случае результат вычислений может оказаться ошибочным.

* Эти особенности являются следствием не ошибок разработчиков микрокалькулятора, а их попыток найти компромисс между требованиями математического обеспечения и простоты конструкции.

Элементная база и соответственно входной язык микрокалькулятора типа «Электроника Б3-34» использованы в ряде других карманных и настольных программируемых микрокалькуляторов, например, типов «Электроника Б3-54», «Электроника МК-54», «Электроника МК-56». Их отличия в основном связаны с внешним оформлением, характеристиками некоторых компонентов и обозначениями команд на пультах управления. Так, команды, обозначенные на пульте микрокалькулятора типа «Электроника Б3-34» символами arcsin, arccos, arctg, ↑, $\overrightarrow{XY}$, П, ИП, на пультах микрокалькуляторов типов «Электроника $\overleftarrow{МК}$-54» (см. рис. 10,*в*) и «Электроника МК-56» обозначены соответственно символами $\sin^{-1}$, $\cos^{-1}$, tg^{-1}, В↑, ↔, $x \rightarrow П$, $П \rightarrow x$, причем два последних особенно неудобны при составлении программ. В некоторых микрокалькуляторах переключатель Р—Г заменен трехпозиционным переключателем, обеспечивающим задание аргумента тригонометрических функций также в десятичных градусах *(градах)*, соответствующих делению прямого угла на 100 частей. Однако эти отличия не затрагивают словарного запаса и грамматики входного языка, который в дальнейшем будем называть входным языком ЯМК34, используя в программах символы команд, характерные для микрокалькуляторов типа «Электроника Б3-34».

Входные языки с обратным синтаксисом характерны и для зарубежных массовых программируемых микрокалькуляторов, например, настольного микрокалькулятора ЕЛКА-59 (8 регистров памяти, 124 ячейки программной памяти) болгарского производства, карманных микрокалькуляторов HP25C и HP33E (8 регистров, 49 ячеек) или HP19C и HP29C (30 регистров, 98 ячеек), выпускаемых в ряде стран по лицензии фирмы Хьюллет—Паккард. Входные языки с обратным синтаксисом используются и в более сложных программируемых микрокалькуляторах, разработанных этой фирмой, например, HP67 и HP97 (26 регистров, 224 ячейки, периферийные устройства) и одном из наиболее сложных HP41C, емкость оперативной памяти которого может быть расширена до 32 кбайт [8].

Глава 2

ПОГРЕШНОСТИ РЕЗУЛЬТАТОВ ВЫЧИСЛЕНИЙ

1. ПРИЧИНЫ ПОГРЕШНОСТЕЙ ИНЖЕНЕРНЫХ РАСЧЕТОВ

Задача инженерного проектирования на ее теоретическом этапе предполагается оконченной, если расчетные характеристики, полученные по математической модели проектируемого объекта, совпадают в требуемых пределах с характеристиками, предусмотренными техническим заданием. Однако даже при полном совпадении расчетных характеристик с требуемыми они практически всегда отличаются от характеристик реального устройства, построенного в соответствии с проектом. Эти отклонения численных значений физических величин,

характеризующих свойства реального объекта, от их расчетных значений называют *погрешностями инженерных расчетов.*

В общем случае разность $\Delta_{\text{ист}} = a - a^*$ приближенного a и точного a^* значений называют *истинной* абсолютной погрешностью, а число $\delta_{\text{ист}} = \Delta_{\text{ист}} / |a^*|$ — истинной относительной погрешностью величины a. Однако точное значение a^* обычно неизвестно и поэтому степень отклонения от него известного приближенного значения оценивают симметричным относительно a интервалом $a - \Delta \leqslant a^* \leqslant a + \Delta$, в котором находится точное значение. Половину ширины этого интервала

$$\Delta \geqslant |a - a^*| \tag{2.1а}$$

называют *абсолютной погрешностью* приближенной величины a. Отношение абсолютной погрешности приближенной величины к ее модулю (часто выражаемое в процентах)

$$\delta = \Delta / |a| \geqslant |a - a^*| / |a^*| \tag{2.1б}$$

называют *относительной погрешностью* a — в этом случае точное значение находится в интервале $a(1 - \delta) \leqslant a^* \leqslant a(1 + \delta)$.

Точность приближенной величины a обычно оценивают по ее относительной погрешности δ, но во многих случаях (в частности, при близости a к нулю) для оценки точности удобнее использовать абсолютную погрешность Δ. Приняв δ или Δ достаточно большими, можно всегда определить интервал, в котором находится точное значение a^*, но оценка истинной погрешности будет тем точнее, чем ближе неравенства (2.1) к равенствам. Поэтому наименьшие по имеющейся информации оценки погрешностей, гарантирующие выполнение неравенств (2.1), называют *предельными* (хотя слово предельный в таких случаях обычно опускают).

Полная погрешность приближенной величины обычно складывается из частных погрешностей, вызванных различными причинами, и *мажоритарная* оценка полной погрешности соответствует сложению модулей ее составляющих, что обеспечивает выполнение условий (2.1) при всех возможных отклонениях величины a от ее точного значения. Погрешности физических величин часто обусловлены воздействием множества различных причин, примерно одинаково влияющих на полную погрешность, и вероятность наихудшего сочетания составляющих с одинаковыми знаками оказывается весьма малой. В таких случаях истинная погрешность значительно меньше ее мажоритарной оценки, и поэтому прибегают к *статистическим* оценкам по среднеарифметическим значениям $\overline{\Delta}$ или $\overline{\delta}$ погрешностей и их дисперсиям σ^2_Δ или σ^2_δ, определяя доверительный интервал $[\alpha, \beta]$, в котором с выбранной доверительной вероятностью $P_{\text{дов}}$ находится точное значение рассматриваемой величины. Границы доверительного интервала в общем случае определяют из соотношения

$$P_{\text{дов}} = \int_\alpha^\beta p(a)\,da, \tag{2.2}$$

где $p(a)$ — закон распределения полной погрешности величины a.

При симметричных законах распределения полной погрешности вместо выражения (2.2) для статистической оценки в неравенствах (2.1) принимают

$$\Delta = |\bar{\Delta}| + K\sigma_{\Delta};\ \delta = |\bar{\delta}| + K\sigma_{\delta}, \qquad (2.3)$$

где K — коэффициент, зависящий от заданной доверительной вероятности и вида закона распределения.

Для известных законов распределения полной погрешности можно составить по соотношению (2.2) или найти в литературе таблицы, связывающие значение коэффициента K с выбранным значением доверительной вероятности. В частности, для наиболее характерного нормального закона распределения значениям $P_{\text{дов}} = 0{,}8;\ 0{,}85;\ 0{,}9;\ 0{,}95;\ 0{,}98;\ 0{,}9973;\ 0{,}999$ соответствуют значения коэффициента K, равного 1,282; 1,439; 1,643; 1,960; 2,325; 2,576; 3,000; 3,290.

Точность величины a, отображенной числом, оценивают также по количеству верных цифр в его десятичном представлении. Первые слева k цифр в таком представлении называют *верными*, если абсолютная погрешность числа не превышает половины единицы младшего k-го разряда. Например, точное значение числа 32,075 со всеми верными цифрами находится в пределах от 32,0745 до 32,0755. Если дробная часть числа со всеми верными цифрами оканчивается нулями, то их сохраняют для указания на точность числа. Так, запись 435,800 означает, что это число определено с точностью до шести верных значащих* цифр и его точное значение заключено в пределах от 435,7995 до 435,8005.

Представление оканчивающихся нулями целых чисел в естественной форме (с фиксированной запятой) не позволяет непосредственно судить об их точности, поэтому более удобно представление чисел в показательной форме (с плавающей запятой). Так, по естественной записи числа 14 000 нельзя точно определить число верных цифр, тогда как представление этого числа в показательной форме $1{,}40 \times 10^3$ свидетельствует об определении числа с точностью до трех верных цифр, когда точное значение лежит в пределах от 13 950 до 14 050. Следовательно, верные цифры приближенного числа не всегда совпадают с цифрами его точного представления, а иногда полностью отличаются. Например, в приближении 3,9999 точного числа 4 первые четыре цифры верны, так как приближают все числа, лежащие в интервале от 3,9995 до 4,0005.

Количество верных цифр служит мерой как абсолютной, так и относительной погрешности приближенных представлений чисел. Если число представлено k верными цифрами, то при $k \geqslant 2$ по мажоритарной оценке

$$\Delta = 5 \cdot 10^{n-k};\ \delta = 5/M\ 10^k = 5 \cdot 10^{-k}/M, \qquad (2.4а)$$

---

* *Значащими* называют все цифры десятичного представления числа от 1 до 9, а также нули, расположенные между ними и справа от них в целой и, при указании точности, дробной части числа.

где M и n — мантисса и порядок числа в показательной форме. Если мантисса неизвестна, то используют более грубую оценку

$$\delta \leqslant 5/10^k = 5 \cdot 10^{-k}. \qquad (2.4б)$$

При одной верной цифре эти формулы приводят к слишком грубому приближению и в качестве мажоритарной оценки принимают $\delta \leqslant 1/M \leqslant 1$.

В свою очередь, по известной абсолютной или относительной погрешности представления числа количество его верных цифр определяют из соотношения $k = \mathrm{E}\,[\lg(5 / \delta M)]$ или, при неизвестной мантиссе, $k = \mathrm{E}\,[\lg(1/2\delta)]$, где E — символ целой части числа в скобках; $M\delta$ — относительная погрешность мантиссы M.

Если погрешность приближенного представления числа не превышает единицы k-го разряда, то его первые k цифр называют *верными в широком смысле*. Например, приближения 32,074 и 32,076 точного числа 32,075 содержат четыре верных (в узком смысле) цифры и пять — в широком смысле. Если абсолютная погрешность представления числа не превышает двух единиц последнего разряда, то содержащуюся в этом разряде цифру называют *сомнительной* (если нет уверенности, что она не превышает единицы).

В процессе проектирования инженер по условиям технического задания выбирает структурную схему проектируемого объекта, в соответствии с которой из математических моделей компонентов составляет математическую модель объекта и рассчитывает требуемые характеристики. Источниками погрешностей результатов инженерных расчетов являются погрешности математического моделирования и погрешности вычислений.

Свойства компонентов проектируемого объекта для заданных рабочих условий отображают числами, характеризующими отношения соответствующих физических величин к единицам их измерения. Зависимости таких чисел x от изменения рабочих условий, полученные в результате наблюдений или измерений, моделируют некоторыми функциями $x(t)$, где t — независимая координата отсчета, обычно имеющая размерность времени. В окружающем мире любое явление есть причина и следствие других явлений, но причиной свойства явления, описанного переменной $x(t)$, будет не координата отсчета t, а некоторое другое свойство того же или другого явления, описанное переменной $q(t)$ в той же системе отсчета. Причинно-следственная связь между *воздействием* (причиной) $q = q(t)$ и *реакцией* на это воздействие (следствием) $x = x(t)$ моделируется функцией

$$x = x(q) \equiv aq, \qquad (2.5)$$

которую называют *динамической* или *временно́й* (если координата отсчета имеет размерность времени) характеристикой канала связи между воздействием и реакцией на это воздействие.

Свойства канала связи в этой формуле определены *параметром* $a = x(q)/q$, но формально в таком канале над переменной q выполняется некоторая операция, результатом которой является переменная x. Поэтому букву a в формуле (2.5) можно также рассматривать как сим-

вол *оператора*, ставящего число x в соответствие числу q. Математическая модель этого оператора (параметра) зависит от конкретных свойств моделируемого канала причинно-следственной связи. Вследствие материальности таким каналам присущи все основные свойства каналов передачи энергии — нелинейность, параметричность, случайность, инерционность и распределенность.

Энергия, поступающая от источника воздействия q, изменяет свойства канала, что проявляется в зависимости оператора $a = a(q)$ от уровня воздействия и нелинейности уравнения (2.5). Свойства канала связи зависят также от множества сторонних (по отношению к рассматриваемому воздействию q) воздействий z, явно не учитываемых в формуле (2.5). Поэтому оператор $a = a(q, z)$ является сложной функцией многих переменных, а канал связи с учетом влияния сторонних воздействий на оператор (параметр) a называют *параметрическим* или активным. В таком канале реакция поддерживается энергией, поступающей как от источника воздействия q, так и от сторонних воздействий z, являющихся внутренними (так как они учитываются изменениями оператора a) для такого канала.

Всеобщая взаимосвязь явлений в природе проявляется в зависимости любой реакции от бесконечного числа воздействий, что приводит к *случайным* изменениям оператора (параметра) реального канала связи и возможности определения лишь его среднестатистического значения $a = \bar{a}(q, z)$.

Процессы накопления энергии в физических каналах связи приводят к их *инерционности*, проявляющейся в зависимости оператора канала и, следовательно, реакции от значений воздействия не только в данный, но и в предыдущие моменты времени. Реакция таких каналов численно определяется решением дифференциально-интегрального (относительно независимой переменной t) уравнения. Ограничения скорости распространения энергии проявляются в задержке во времени и *распределенности* реакции (отображаемой распределенностью параметров) вдоль протяженных каналов связи. Поэтому в общем случае реакция канала связи определяется решением нелинейного параметрического дифференциального уравнения со случайными коэффициентами и частными производными по времени и пространственным координатам.

Физический объект или его компонент с n независимыми *входами* (местами определения воздействий и реакций) рассматривают как автономную физическую систему, моделируемую системой в общем случае операторных уравнений

$$q_j = \sum_{i=1}^{n} a_{ji} x_i; \quad j = 1,2, \ldots, n \tag{2.6}$$

или, в сокращенной матричной записи,

$$Q = AX,$$

где Q и X — векторы-столбцы переменных q_j и x_i на входах; A — квадратная матрица операторов (параметров) a_{ji}, отображающих свойства каналов причинно-следственных связей между переменными на входах.

Систему уравнений (2.6) можно преобразовать в другую эквивалентную (отображающую свойства моделируемого объекта на заданных входах с такой же точностью) модель в виде систем уравнений, графов, схем замещения или частично-упорядоченных множеств чисел [14], а также расчетных формул для вычисления нужных характеристик объекта. Эти характеристики отображают как *буквенными* моделями (формулами), *графическими* или *табличными* моделями численных зависимостей, так и алгоритмами их вычисления при представлении программами, называемыми *цифровыми* моделями.

Свойства проектируемого и еще не существующего в природе объекта инженер может описать только его математической моделью, составленной из математических моделей компонентов с известными свойствами. Однако в связи с бесконечным многообразием свойств физических объектов точное математическое моделирование компонентов принципиально невозможно. Обычно учитывают лишь наиболее существенные свойства каналов причинно-следственных связей в компонентах, пренебрегая теми из них, которые в заданных рабочих условиях мало влияют на зависимости реакций от воздействий.

В частности, каналы малой протяженности обычно рассматривают как *сосредоточенные*, а их параметры — как *детерминированные*, учитывая случайный характер последних лишь при необходимости. При достаточно медленных изменениях воздействий пренебрегают инерционностью каналов связи, описывая их *статическими* характеристиками, отображающими зависимости реакций только от уровня воздействий. При малых изменениях воздействий каналы связи моделируют *линейными* уравнениями с постоянными параметрами или операторами.

Погрешности *математических моделей* компонентов, связанные с подобными рассмотренным идеализациям их реальных свойств, разделяются на погрешности структуры, параметров и определения воздействий. Погрешности *структуры* математической модели связаны с невозможностью полного отображения всех причинно-следственных связей в реальном объекте. Учет большого числа этих связей приводит к усложнению модели и ее последующего использования, а составление достаточно точных и простых моделей физических объектов является сложной задачей, связанной с трудоемкими научными исследованиями. Поэтому в инженерной практике для типовых компонентов используют стандартные модели, структура которых описана в учебниках, инженерных справочниках и паспортных данных компонентов. Задача инженера в этом случае — выбрать стандартную модель, структура которой отображает свойства компонента в заданных рабочих условиях с достаточной точностью.

Модели с различной структурой отличаются количеством и численными значениями параметров, характеризующих свойства причинно-следственных связей в компоненте. Погрешности *параметров* определяются отклонениями принятых для модели значений параметров от соответствующих численных значений физических величин в реальном компоненте. Эти погрешности минимальны при непосредственном измерении параметров каждого из выбранных для проектируемого объекта

компонентов в заданных рабочих условиях. Однако при проектировании серийных изделий приходится учитывать технологический разброс параметров однотипных компонентов и выбирать при моделировании номинальные (средние) значения параметров с учетом их предельных погрешностей (допусков).

Погрешности *определения воздействий* связаны с отклонениями принятых при расчете значений воздействий от их реальных значений. Эти погрешности вместе с погрешностями параметров обычно называют погрешностями *исходных данных*.

Однако даже при использовании достаточно точных математических моделей точность результата инженерных расчетов может оказаться низкой в связи с влиянием операционных и методических погрешностей вычислений. Основная причина *операционных* погрешностей связана с ограничением разрядности чисел при выполнении вычислительных операций. Представление любого числа, содержащее теоретически бесконечное (например, $1/3 = 0{,}33333...$ или $\sqrt{2} = 1{,}414213...$) или большее, чем принято при выполнении операции, число цифр, округляется и, следовательно, становится приближенным. При обычных вычислениях операционные погрешности можно уменьшить, увеличив допустимое количество цифр (разрядность) десятичных представлений чисел, но автоматические вычисления в ЭВМ и, в частности, микрокалькуляторах, выполняются с фиксированной разрядностью, что приводит к операционным погрешностям. При выполнении нескольких операций погрешности вычислений накапливаются и практически всегда оказываются бóльшими погрешностей исходных данных.

Методические погрешности вычислений возникают при использовании методов (например, при последовательных приближениях, численном интегрировании, суммировании бесконечных рядов), обеспечивающих получение точного результата после выполнения теоретически бесконечного числа операций. Так как практически выполнимо конечное число операций, то и возникает методическая (остаточная) погрешность результата вычислений, называемая также погрешностью ограничения. Ее можно уменьшить, увеличив число операций, но в этом случае возрастет операционная составляющая результата вычислений.

Влияние погрешностей модели и погрешностей вычислений на результат инженерного расчета зависит также от выбора алгоритма решения задачи инженерного проектирования. Однако стремление достичь максимальной точности результата за счет повышения точности модели и точности вычислений связано со значительными затратами времени и не всегда оправдано.

Точность определения физических величин, соответствующих численным результатам инженерных расчетов, зависит от возможной точности их измерения и точности изготовления соответствующих физических объектов. Поэтому требуемая точность результатов инженерного расчета зависит только от точности определения соответствующих физических величин. Практически число верных цифр численного результата инженерного расчета должно быть равно числу верных цифр

в численном представлении соответствующей физической величины. В связи с этим полученные результаты расчета округляют до этого (или на единицу большего) числа верных цифр, так как представление результатов с большим числом значащих цифр не имеет смысла и может ввести лишь в заблуждение.

Следовательно, инженер должен уметь оценивать точность (например, число верных цифр) результата расчета и в соответствии с его требуемой точностью оценивать допустимые погрешности модели и вычислений.

2. ОПЕРАЦИОННЫЕ ПОГРЕШНОСТИ АВТОМАТИЧЕСКИХ ВЫЧИСЛЕНИЙ

Максимальное количество цифр мантиссы* и порядка десятичных представлений чисел, высвечиваемых на индикаторе микрокалькулятора, определяется постоянными числами r и l соответствующих цифровых знакомест индикатора. Набираемые исходные данные и высвечиваемые результаты вычислений не могут содержать более r и l цифр мантиссы и порядка, что и является основной причиной операционных погрешностей при автоматическом выполнении операций микрокалькулятором.

Результат операции, отличающийся от нуля, но меньший по модулю нижней границы $A_{\min}$ диапазона представления чисел (определяемого числами r и l), попадает в область машинного нуля и на индикаторе высвечивается нуль с абсолютной операционной погрешностью, равной истинному значению результата выполненной операции. Результат операции, бóльший по модулю верхней границы $A_{\max}$ диапазона представления чисел, содержит больше r и l цифр, что приводит к переполнению запоминающих устройств микрокалькулятора и прекращению вычислений. Результат операции, попадающий в диапазон представления чисел, но содержащий большее число цифр мантиссы, автоматически округляется до значения, содержащего только r цифр, что приводит к операционной погрешности округления.

Эта погрешность зависит от способа округления, определяемого конструкцией микрокалькулятора. Простейший и технически наиболее просто реализуемый способ округления заключается в сохранении первых r цифр мантиссы без учета влияния отбрасываемых цифр в младших разрядах дробной части. Погрешность округления отбрасыванием зависит от способа представления чисел в микрокалькуляторе, но в общем случае сохраняются верные цифры в широком смысле, так как абсолютная погрешность результата округления не превышает единицы последнего сохраняемого разряда.

В естественной форме представления чисел, больших единицы по модулю (в показательной форме их порядок $n \geqslant 0$), при округлении отбрасыванием сохраняется r разрядов, причем цена (число, отображаемое единицей разряда) младшего сохраняемого разряда равна

---

* Мантиссой можно назвать и естественное представление числа с фиксированной запятой с учетом показателя порядка $n = 0$.

$1 \cdot 10^{n-r+1}$. Поэтому абсолютная погрешность результата по мажоритарной оценке

$$\Delta = 10^{n-r+1}, \tag{2.7а}$$

а относительная погрешность при значении M мантиссы

$$\delta = 10^{1-r}/M \tag{2.7б}$$

или, если мантисса неизвестна,

$$\delta = 1 \cdot 10^{1-r}. \tag{2.7в}$$

В естественной форме представления чисел, меньших единицы по модулю (их порядок n отрицателен), $|n|$ левых разрядов окажутся занятыми нулями (например, для 0,0038745 порядок $n = -3$), абсолютная погрешность $\Delta = 1 \cdot 10^{1-r}$ не зависит от n, а относительная погрешность $\delta = 1 \cdot 10^{1-r-n}/M$ будет тем большей, чем меньше n. Так, при $n = -7$ и $r = 8$ сохраняется только одна значащая цифра результата и относительная погрешность достигает 100 %, если отбрасываемая часть результата близка к 10^{-7}, а цифра мантиссы — к единице.

Для микрокалькуляторов с показательной формой представления чисел при округлении отбрасыванием независимо от знака порядка мажоритарными являются оценки операционных погрешностей

$$\Delta < 1 \cdot 10^{n-r+1};\ \delta < 1 \cdot 10^{1-r}/M < 1 \cdot 10^{1-r}. \tag{2.8}$$

Результаты ручных операций с помощью карандаша и бумаги обычно округляют более точными способами, называемыми симметричным и несимметричным округлением *по дополнению*. При *несимметричном* округлении сохраняемая часть представления числа не изменяется, если первая отбрасываемая цифра меньше 5, и увеличивается на единицу последнего сохраняемого разряда в противном случае. *Симметричное* округление отличается тем, что при равенстве отбрасываемой части половине единицы последнего сохраняемого разряда его нечетное содержимое увеличивается на единицу, а четное не изменяется. Округленное по дополнению десятичное представление числа содержит все верные (в узком смысле) цифры, так как предельная погрешность округления не превышает половины единицы последнего сохраняемого разряда.

Если в микрокалькуляторе используется один из рассмотренных способов округления или близкий к ним, то его несложно определить простыми проверками. Так, для микрокалькулятора с $r = 8$ при округлении отбрасыванием

$$10000001 + 0{,}5 = 10000001;\ 10000002 + 0{,}5 = 10000002,$$

при симметричном округлении

$$10000001 + 0{,}5 = (10000002);\ 10000002 + 0{,}5 = (10000003)$$

и при несимметричном округлении

$$10000001 + 0{,}5 = (10000002);\ 10000002 + 0{,}5 = (10000002).$$

Подобные тестовые примеры несложно составить и для проверки способа округления микрокалькулятором результатов других арифметических операций.

В микрокалькуляторах с входными языками ЯМК21 и ЯМК34 результаты умножения и деления округляются отбрасыванием, но для округления результатов сложения и вычитания использованы различные способы. В первых выпусках микрокалькуляторов «Электроника Б3-21» и в микрокалькуляторах с входным языком ЯМК34 в результате сложения последовательно по несимметричному дополнению округляется содержимое младших разрядов суммы, не попадающих в разрядную сетку, например,

$$10000000 + 0{,}444444 = (10000000); \quad 10000000 + 0{,}44445 = (10000001).$$

Вследствие этого при вычитании получается, например,

$$10000000 - 0{,}55 = (10000000); \quad 10000000 - 0{,}56 = (9999999).$$

Этот способ округления приводил в первых выпусках микрокалькулятора «Электроника Б3-21» к рассмотренному ранее неконтролируемому переполнению при сложении некоторых чисел, например, $10 + 9{,}9999999 = (120)$. В более поздних выпусках микрокалькулятора этого типа применен близкий к несимметричному способ округления, при котором в процессе вычитания числа, более чем на $r = 8$ отличающегося по порядку от вычитаемого, происходил «заем» единицы последнего разряда результата, например, $100 - 4 \cdot 10^{-66} = (99{,}999999)$.

Необходимо учитывать, что при сложении и вычитании операндов a и b, попадающих в диапазон представления чисел микрокалькулятора, но отличающихся по порядку на $r = 8$ или более, меньший операнд b попадает в область относительного машинного нуля и не влияет на результат операции $a + b = (a)$, равный большему операнду. Подобная ситуация возникает при суммировании убывающих по модулю членов ряда, когда значение очередного члена ряда оказывается относительным машинным нулем относительно суммы предыдущих членов и результат сложения перестает изменяться. В таких случаях для сокращения времени счета в качестве критерия автоматического прекращения вычислений следует выбирать не малость очередного члена ряда, а малость разности между суммой предыдущих членов ряда и ее суммой с очередным членом ряда.

Оценку погрешностей округления в необходимых случаях уточняют статистическими методами. Для статистических оценок нужно знать закон распределения погрешности однократного округления, по которому можно найти среднее и дисперсию погрешности округления множества результатов арифметических операций. Если предполагать распределение содержимого младших отбрасываемых разрядов равновероятным в пределах одной единицы последнего сохраняемого разряда, то среднее значение и дисперсия абсолютной погрешности округления отбрасыванием $\bar{\Delta} = \Delta/2$, $\sigma_\Delta^2 = (\Delta/12)^2$, а при симметричном и несимметричном округлении $\bar{\Delta} = 0$, $\sigma_\Delta^2 = (\Delta/12)^2$, где Δ — мажо-

ритарная оценка предельной абсолютной погрешности числа, отображаемого единицей последнего сохраняемого разряда.

Для статистической оценки относительной погрешности, зависящей и от мантиссы округляемого числа, должен быть известным закон распределения мантиссы M, который при $1 \leqslant M < 10$ близок к логарифмическому и $p(M) = 1/M \ln 10$ [17]. В этом случае совместный закон распределения мантиссы и младших отбрасываемых разрядов при их независимости для округления отбрасыванием

$$p(\delta_н) = \begin{cases} 9/\ln 10 & \text{при } 0 \leqslant |\delta_н| \leqslant 0{,}1; \\ (1 - \delta_н)/(\delta_н \ln 10) & \text{при } 0{,}1 \leqslant |\delta_н| \leqslant 1, \end{cases}$$

где нормированное значение относительной погрешности $\delta_н = \delta 10^{r-1}$. В этом случае $\bar{\delta}_н = 0{,}45/\ln 10 \approx 0{,}195$; $\sigma_{\delta_н} = ((0{,}165/\ln 10) - (0{,}45/\ln 10)^2)^{1/2} \approx 0{,}183$; $\sigma^2_{\delta_н} \approx 0{,}0335$.

При несимметричном и симметричном округлениях по дополнению

$$p(\delta_н) = \begin{cases} (1 - 2|\delta_н|)/(2|\delta_н| \ln 10) & \text{при } 0{,}5 \leqslant |\delta_н| \leqslant 0{,}05; \\ 9/\ln 10 & \text{при } 0 \leqslant |\delta_н| \leqslant 0{,}05, \end{cases}$$

когда $\bar{\delta}_н = 0$ и $\sigma_{\delta_н} = (0{,}33/8 \ln 10)^{1/2} \approx 0{,}134$.

В некоторых задачах закон распределения содержимого старших разрядов мантиссы представления числа можно полагать равновероятным. В этом случае при округлении отбрасыванием

$$p(\delta_н) = \begin{cases} 11/2 & \text{при } 0 \leqslant |\delta_н| \leqslant 0{,}1; \\ (1 - \delta^2_н)/18\,\delta^2_н & \text{при } 0{,}1 \leqslant |\delta_н| \leqslant 1 \end{cases}$$

и $\bar{\delta}_н = \ln 10/18 \approx 0{,}128$; $\sigma_{\delta_н} = (1/30 - (\ln 10/18)^2)^{1/2} \approx 0{,}13$, а при несимметричном и симметричном округлениях

$$p(\delta_н) = \begin{cases} (1 - 4\,\delta^2_н)/72\,\delta_н & \text{при } 0{,}05 \leqslant |\delta_н| \leqslant 0{,}5; \\ 11/2 & \text{при } 0 \leqslant |\delta_н| \leqslant 0{,}05 \end{cases}$$

и $\bar{\delta}_н = 0$, $\sigma_{\delta_н} = (1/120)^{1/2} \approx 0{,}09$.

Подобным образом можно определить статистические оценки погрешностей округления и при других законах распределения мантиссы и содержимого отбрасываемых разрядов.

Автоматическое вычисление элементарных функций в инженерных и программируемых микрокалькуляторах выполняется по микропрограммам, постоянно хранимым в *ПЗУ*. Обычно в этих микропрограммах вычисление функций реализовано методами последовательных приближений с определенной методической погрешностью. Однако подобные погрешности вычисления функций по команде, подаваемой нажатиями клавиш, следует рассматривать как операционные, так как их нельзя уменьшить увеличением числа более простых операций, и для пользователя вычисление функции вводом соответствующего оператора входного языка является операцией.

Оценки предельных погрешностей автоматического вычисления элементарных функций, относящиеся к паспортным данным микрокаль-

кулятора, обычно указаны в инструкции по его эксплуатации. Обычно погрешность автоматического выполнения арифметических операций и вычисления функций не превышает единицы последнего разряда мантиссы и лишь для некоторых функций погрешность лежит в пределах двух единиц последнего младшего разряда представления мантиссы на индикаторе.

Следует учитывать, что цена единицы младшего разряда мантиссы зависит от ее значения и формы представления числа. Так, при вычислении $\sin x$ или $\cos x$ предельная погрешность, оцениваемая единицей младшего разряда микрокалькулятора с $r = 8$, соответствует абсолютной погрешности результата вычисления функции $\Delta \leqslant 1 \times \times 10^{-7}$ независимо от значения аргумента, тогда как относительная погрешность увеличивается при уменьшении результата вычисления функции. Например, значение функции $\sin 9{,}999999 \cdot 10^{-6} = 0{,}0000099$, вычисленное с абсолютной погрешностью $\Delta = 1 \cdot 10^{-7}$, будет иметь относительную погрешность $\delta = 1 \cdot 10^{-7}/9{,}9 \cdot 10^{-6} \approx 0{,}01$ или 1 %.

Микрокалькуляторы вычисляют различные функции с разными погрешностями, зависящими от значения аргумента. Так, микрокалькулятор «Электроника Б3-21» почти для всех значений аргумента $x > 0$ вычисляет восемь цифр значения экспоненциальной функции, например $e^{\pi} = 23{,}140687$ и $e^{11} = 598{,}74129$. Однако значение $e^{10} = 22026{,}46$ содержит лишь семь значащих цифр. При $x < 0$ этот микрокалькулятор вычисляет значения e^x с семью или даже с шестью значащими цифрами, например, $e^{-\pi} = 0{,}4321392$; $e^{-10} = 0{,}4539994 \times \times 10^{-4}$; $e^{-11} = 0{,}167017 \cdot 10^{-4}$. В этих случаях можно полагать, что относительная погрешность результата вычисления e^x при положительных значениях аргумента ограничена значением 10^{-7}, а при отрицательных — значением 10^{-8}.

Для грубой оценки погрешностей вычисления функций микрокалькулятором можно сравнивать вычисляемые значения с известными или, когда это возможно, сравнить результат последовательного вычисления функции и обратной ей функции с исходным значением аргумента. При такой проверке для микрокалькулятора с входным языком ЯМК21 получим $\pi/6$ sin (0,5); $\pi/3$ cos (0,5); 2 ↑ 3 X^y (8,9999794); 3 ↑ 2 X^y (7,9999994); 2 ln (0,693147) e^x(1,9999997); 2 $\sqrt{}$ (1,4142135) x^2(1,9999998).

При подобной проверке для микрокалькулятора с входным языком ЯМК34 получим 2 sin (0,90929745) arcsin (1,1415926) sin (0,90929745); 2 cos (—0,41614688) arccos (2); 2 tg (—2,1850397)arctg (— 1,1415926) tg (— 2,1850397); 2 ln (0,69314717) e^x(2) e^x (7,3890557) ln (2); 2 lg (0,30102999) 10^x(2) 10^x(100) lg (2); 2 $\sqrt{}$ (1,4142135) x^2 (1,9999998) $\sqrt{}$ (1,4142134) x^2(1,9999995); 2 ↑ 3 X^y(8,9999984); 3 ↑ 2 X^y (7,9999993).

Для вычисления функции с максимально достижимой точностью нужно исследовать погрешность ее вычисления микрокалькулятором, сравнивая результаты вычислений с данными достаточно точных таблиц или результатами точных вычислений. После этого следует составить таблицы или графики для коррекции результата вычисления функции микрокалькулятором в заданном интервале изменения аргумента.

3. ОЦЕНКА ПОГРЕШНОСТИ РЕЗУЛЬТАТОВ РАСЧЕТА

Результат вычислений по любому алгоритму можно представить в виде некоторой функции $f = f(x_1, x_2, \ldots, x_m)$ исходных данных и результатов выполнения каждой операции. Если эти переменные x_i определены с абсолютной погрешностью Δx_i, то результат вычислений раскладывается в многомерный ряд Тейлора

$$f = f_0 + \sum_{i=1}^{m} \frac{\partial f}{\partial x_i} \Delta x_i + \frac{1}{2} \sum_{\substack{i=1 \\ j=1}}^{m} \frac{\partial^2 f}{\partial x_i \partial x_j} \Delta x_i \Delta x_j + \ldots, \qquad (2.9)$$

где f_0 — значение функции при $\Delta x_i = 0$.

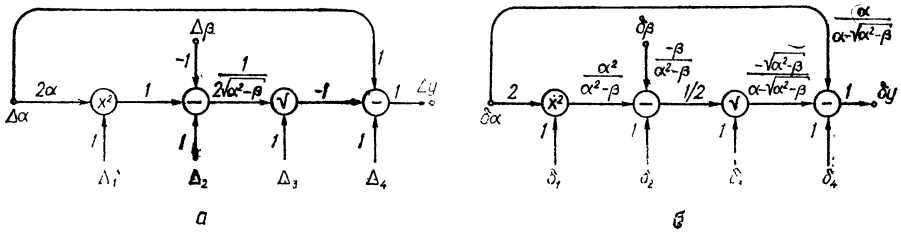


Рис. 14

При достаточно малых погрешностях Δx_i этот ряд ограничивают линейными членами и оценивают абсолютную погрешность результата вычислений по формуле

$$\Delta f = f - f_0 \approx \sum_{i=1}^{m} f'_i \Delta x_i, \qquad (2.10)$$

где производные $f'_i = \partial f / \partial x_i$ определяют зависимости погрешности результата вычислений от абсолютных погрешностей исходных данных и результатов выполнения каждой операции.

Для оценки точности результата вычислений по его абсолютной погрешности удобно использовать граф накопления погрешностей, вершины которого (кроме вершин-источников погрешностей и вершины-стока погрешности результата вычислений) обозначают символами выполняемых операций или их результатов. Дугам графа приписывают веса f'_i, равные производным результата операции по соответствующему операнду, кроме дуг операционных погрешностей с единичным весом. Вершинам-источникам сопоставляют соответствующие операционные погрешности или погрешности исходных данных.

Значения производных f'_i для основных операций, выполняемых микрокалькуляторами, приведены в табл. 3. Погрешность результата вычислений определяют по такому графу, как сумма всех произведений исходных погрешностей на веса дуг, соединяющих вершины-источники с вершиной результата вычислений. Например, по графу накопления абсолютных погрешностей (рис. 14, *а*) при вычислениях

5. Коэффициенты пропорциональности малых погрешностей операндов погрешностям результатов выполнения операций

Результат операции	Коэффициенты пропорциональности	
	$f_i' = \partial f/\partial x_i$	$S_i = (x_i/f)\, f_i'$
$f = x_1 + x_2$	$f_1' = f_2' = 1$	$S_1 = x_1/(x_1 + x_2)$, $S_2 = x_2/(x_1 + x_2)$
$f = x_1 - x_2$	$f_1' = 1$, $f_2' = -1$	$S_1 = x_1/(x_1 - x_2)$, $S_2 = -x_2/(x_1 - x_2)$
$f = x_1 x_2$	$f_1' = x_2$, $f_2' = x_1$	$S_1 = S_2 = 1$
$f = x_1/x_2$	$f_1' = 1/x_2$, $f_2' = -x_1/x_2^2$	$S_1 = 1$, $S_2 = -1$
$f = x_1^{x_2}$	$f_1' = x_2 x_1^{x_2 - 1}$, $f_2' = x_1^{x_2} \ln x_1$	$S_1 = x_2$, $S_2 = x_2 \ln x_1$
$f = 1/x$	$-1/x$	-1
$f = x^2$	$2x$	2
$f = \sqrt{x}$	$1/2\sqrt{x}$	$1/2$
$f = e^x$	e^x	x
$f = 10^x$	$10^x \ln 10$	$x \ln 10$
$f = \ln x$	$1/x$	$1/\ln x$
$f = \lg x$	$1/x \ln 10$	$1/\ln 10 \ln x$
$f = \sin x$	$\cos x$	$x/\operatorname{tg} x$
$f = \cos x$	$-\sin x$	$-x \operatorname{tg} x$
$f = \operatorname{tg} x$	$1/\cos^2 x$	$2x/\sin 2x$
$f = \arcsin x$	$1/\sqrt{1 - x^2}$	$x/(\sqrt{1 - x^2} \arcsin x)$
$f = \arccos x$	$x/\sqrt{1 - x^2}$	$1/(\sqrt{1 - x^2} \arccos x)$
$f = \operatorname{arctg} x$	$1/(1 + x^2)$	$x/((1 + x^2) \operatorname{arctg} x)$

по формуле $y = \alpha - \sqrt{\alpha^2 - \beta}$ несложно определить абсолютную погрешность

$$\Delta y = -(2\alpha\Delta\alpha + \Delta_1 - \Delta\beta + \Delta_2)/2(\sqrt{\alpha^2 - \beta}) + \Delta\alpha - \Delta_3 + \Delta_4.$$

Мажоритарная оценка погрешности суммы соответствует сложению модулей составляющих, но если один и тот же источник приводит к появлению нескольких составляющих, их суммируют с учетом знаков. Например, для рассматриваемого случая

$$\Delta y \leqslant |\Delta\alpha(1 - \alpha/\sqrt{\alpha^2 - \beta}) + (|\Delta\beta| + |\Delta_1| + |\Delta_2|)2\sqrt{\alpha^2 - \beta} + |\Delta_3| + |\Delta_4|.$$

Абсолютные операционные погрешности согласно формулам (2.7а) и (2.8) зависят от порядков результатов промежуточных вычислений. При использовании представлений чисел в показательной форме (с плавающей запятой) мажоритарная оценка относительной погреш-

ности результата вычислений не зависит от порядка промежуточных результатов. Поэтому в большинстве случаев точность результата вычислений удобнее оценивать по относительным погрешностям.

Разделив правую и левую части равенства (2.10) на значение результата вычислений, получим формулу для вычисления его относительной погрешности

$$\delta f = \frac{\Delta f}{f} \approx \sum_{i=1}^{m} \frac{\partial f}{\partial x_i} \frac{x_i}{f} \frac{\Delta x_i}{x_i} = \sum_{i=1}^{m} S_i \delta x_i, \qquad (2.11)$$

где коэффициенты $S_i = f'_i x_i / f$ называют *чувствительностями* (их выражения для основных операций, выполняемых микрокалькуляторами, приведены в табл. 3) результата операции к изменениям операндов x_i.

Накопление относительных погрешностей удобно оценивать по графу, структура которого аналогична графу накопления абсолютных погрешностей, но с весами дуг, равными чувствительностям S_i результатов операций. Для $y = \alpha - \sqrt{\alpha^2 - \beta}$ по такому графу (рис. 14, *б*) получим

$$\delta y \leqslant |-\alpha\delta\alpha/\sqrt{\alpha^2-\beta}| + (|-\beta\delta\beta| + |-\delta_1\alpha^2|)/2\sqrt{\alpha^2-\beta^2}(\alpha - \sqrt{\alpha^2-\beta}) + (|-\delta_2| + 2|-\delta_3|)\sqrt{\alpha^2-\beta}/|2(\alpha-\sqrt{\alpha^2-\beta})| + \delta_4.$$

Если исходные данные предполагаются точными ($\delta\alpha = \delta\beta = 0$), а погрешности округления $\delta_i = 10^{1-r}$, то при $\alpha^2 > \beta$ мажоритарная оценка $\delta y \leqslant 10^{1-r}(1 + (4\alpha^2 - 3\beta)/(2\sqrt{\alpha^2-\beta}(\alpha - \sqrt{\alpha^2-\beta})))$.

Остановимся подробнее на оценке погрешности суммы k чисел, которую обычно [7] находят по графу распространения относительной погрешности (рис. 15, *а*) как

$$\delta y = \sum_{i=1}^{k} x_i \delta x_i / y + \sum_{j=1}^{k-1} \delta_j \sum_{i=1}^{j+1} x_i / y,$$

где y — значение суммы и, если погрешности сложения одинаковы,

$$\delta y = \delta x + \delta \sum_{j=1}^{k-1} \left(\sum_{i=1}^{j+1} x_i / y\right).$$

Рассмотрим частный случай сложения близких по модулю и одинаковых по знаку слагаемых, характерный, в частности, для задач статистической обработки массивов данных при $\bar{x} \gg \sigma$. В этом случае $y = k\bar{x}$, $\sum_{i=1}^{j+1} x_i = (i+1)\bar{x}$, $\delta y \approx \delta x + \delta \sum_{i=1}^{k-1} i/k = \delta x + (k^2 + k - 2)\delta/2k$ и при $k \gg 1$ можно принять

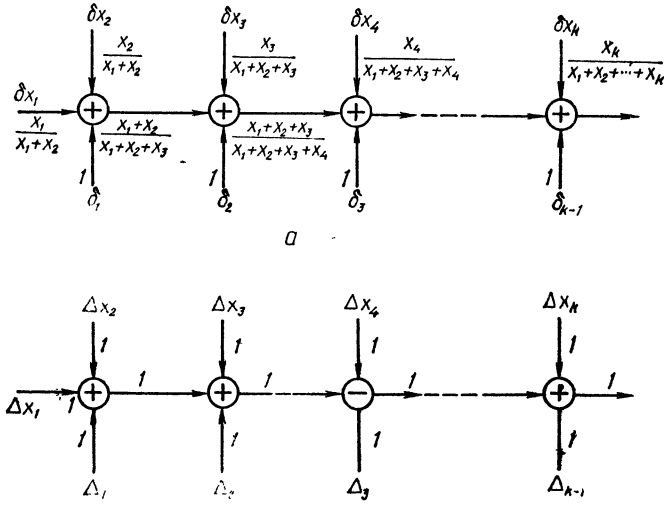
$$\delta y \approx \delta x + k\delta/2.$$

Попробуем уточнить эту оценку по абсолютным значениям погрешностей, получив с помощью графа их распространения (рис. 15,*б*)

оценку абсолютной погрешности суммы

$$\Delta y = \sum_{i=1}^{k} \Delta x_i + \sum_{j=1}^{k-1} \Delta_j.$$

Полная сумма близких по величине слагаемых $y \approx \overline{M}k\,10^n$, а ее порядок больше порядка слагаемых на величину $\Delta n = \mathrm{E}(\lg Mk)$, где символом E обозначена целая часть содержимого скобок. В этом случае последние операции сложения выполняются при округлении отбрасыванием с абсолютной погрешностью $\Delta' = 10^{n-r+1+\Delta n}$, а число



а

б

Рис. 15

таких сложений $k' = k - 10^{\Delta n}/M$. Предыдущие k'' сложений выполняются с меньшей на единицу порядка погрешностью округления $\Delta'' = \Delta'/10$, а их число $k'' = 10^{\Delta n}(1 - 10^{-1})/M = 0{,}9 \cdot 10^{\Delta n}/M$ до первых операций сложения, выполняемых без округления. Следовательно, погрешность суммы, вызванная погрешностями округления $\Delta y = (k - 10^{\Delta n}/M)\,10^{n-r+1+\Delta n} + 0{,}9 \cdot 10^{\Delta n} \cdot 10^{n-r+1+\Delta n}/10\,M + 0{,}09 \times$ $\times\, 10^{\Delta n} \cdot 10^{n-r+1+\Delta n}/100\,M + \ldots + 9 \cdot 10^{-\Delta n} \cdot 10^{n+r+1+\Delta n}/10\,M\,\Delta n =$ $= 10^{n-r+1+\Delta n}(k - (1 - 0{,}09 - 0{,}009 - \ldots - 9 \cdot 10^{-2\Delta n})\,10^{\Delta n}/M) \approx$ $\approx 10^{n-r+1+\Delta n}(k - 0{,}9 \cdot 10^{\Delta n}/M)$. Для упрощения этого выражения обозначим $\Delta n = \mathrm{E}(\lg Mk) = \lg Mk - \alpha$, где $0 < \alpha < 1$. Тогда $10^{\Delta n} = 10^{\lg Mk - \alpha} = Mk/M_c$, где $M_c = 10^{\alpha}$ — мантисса суммы.

Переписав полученное выражение для полной погрешности в виде $\Delta y \approx 10^{n-r+1} Mk\,(k - 0{,}9\,k/M_c)/M_c = Mk^2\,10^{n-r+1}\,(1 - 0{,}9\,M_c)/M_c$, найдем наихудшее значение M_c, максимизирующее полную погрешность. Оно оказывается равным 1,8 и, следовательно, $\Delta y \leqslant Mk^2 \times$ $\times\, 10^{n-r+1}/3{,}6$, что соответствует относительной погрешности суммы

$$\delta y \leqslant k\,10^{1-r}/3{,}6. \qquad (2.12a)$$

Полученная оценка почти вдвое меньше общепринятой [17], что позволяет более точно оценивать погрешность рассматриваемой суммы чисел. Если в микрокалькуляторе предусмотрено симметричное или несимметричное округление по дополнению, то эту оценку можно уменьшить еще в два раза. Оценка (2.12а), оставаясь мажоритарной, дает более точно верхнюю границу возможной погрешности, так как учитывает влияние мантисс промежуточных сумм и числа сложений, выполняемых с различной погрешностью округления. При необходимости эту оценку можно уточнить по значению мантиссы M_c суммы согласно соотношению

$$\delta y = k\,10^{1-r}/[M_c(1 - 0{,}9/M_c)] = k\,10^{1-r}/K_c. \qquad (2.12б)$$

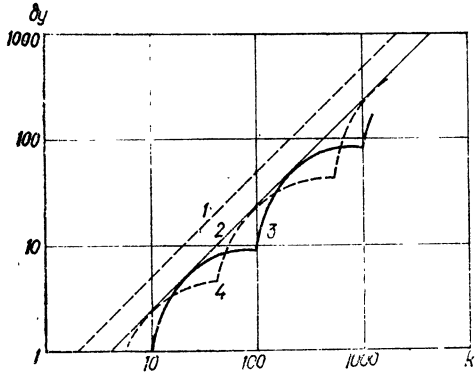


Рис. 16

В качестве иллюстрации на рис. 16 показаны в логарифмическом масштабе графики зависимости погрешности рассмотренной суммы от числа $k - 1$ погрешностей округления, построенные согласно общепринятой мажоритарной оценке (кривая *1*), по формуле (2. 12а) с учетом влияния мантисс промежуточных сумм (кривая *2*) и при значении мантисс сумм $M_c = 1$ (кривая *3*) и $M_c = 2$ (кривая *4*).

Процесс суммирования близких по величине слагаемых распадается на несколько этапов вычисления промежуточных сумм с различными абсолютными погрешностями, и необходимо определить допустимость статистического оценивания операционных погрешностей в зависимости от числа слагаемых. Эти оценки допустимы, когда полная погрешность суммы равна сумме достаточно большого числа примерно равных независимых погрешностей округления. Однако от таких оценок следует воздержаться, если основной вклад в погрешность результата вносят лишь несколько последних операций сложения.

Для проверки возможности статистического оценивания выберем число k_1 слагаемых, при котором в последний раз увеличивается порядок накопленной промежуточной суммы, и обозначим символом q порядок произведения k_1M. Тогда при достаточно большом k_1 справедливо соотношение $k_1M \approx 10^q$, и можно раздельно оценить операционную составляющую погрешности результата $\Delta_1 = 10^{n-r+2} \times$ $\times (\mathrm{E}(100/M) - \mathrm{E}(10/M)) + 10^{n-r+3}(\mathrm{E}(1000/M) - \mathrm{E}(100/M) + \ldots +$ $+ 10^{n-r+q}(\mathrm{E}(10^q/M) - \mathrm{E}(10^{q-1}/M)) \approx 0{,}909\,k_1^2\,M\,10^{n-r}$ при сложении k_1 членов суммы и составляющую $\Delta_0 = k_0\,10^{n-r+q+1}$ при сложении $k_0 = k - k_1$. Эти составляющие равны, когда $k_0 = (k_1^2 M\,0{,}909/M \times$ $\times\,10\,k_1)/10 \cdot 10 = 0{,}0909\,k_1$, но $k_1 = k - k_0$ и $k_0 \approx 0{,}083\,k$. Следовательно, при $k > 100$ и любом сочетании мантисс слагаемых и их

числа основной вклад в погрешность результата вносят более восьми независимых составляющих, что полагают достаточным основанием для статистического оценивания погрешностей.

Применение статистической оценки в рассматриваемом случае связано с определением среднего значения ожидаемой погрешности $\bar{\delta}$ и ее дисперсии σ^2 с учетом $\bar{\delta} = \delta/2$ при округлении отбрасыванием и $\bar{\delta} = 0$ при симметричном или несимметричном округлении по дополнению. Из теории вероятности следует, что дисперсия суммы независимых случайных величин равна сумме их дисперсий и равенство дисперсий абсолютных погрешностей в пределах каждой группы слагаемых можно учесть согласно соотношению

$$\sigma_\Delta^2 = (k - 10^{\Delta n})/M)\, 10^{2(n-r+1+\Delta n)}/12 + 0{,}9 \cdot 10^{\Delta n}\, 10^{2(n-r+1+\Delta n)}/12 \approx$$
$$\approx k^3 M^2 (M_c - 1)\, 10^{2(n-r+1)}/12\, M_c^3.$$

Так как изменения мантисс текущих сумм учтены при выводе исходного соотношения, при получении этой формулы принималась во внимание только случайность значений содержимого разрядов частных сумм, отбрасываемых при округлении.

Полученное выражение максимально при $M_c = 1{,}5$ и при округлении отбрасыванием $\delta \approx (k/7{,}2 + \sqrt{k}/3)\, 10^{1-r}$, а при несимметричном и симметричном округлениях по дополнению $\delta \approx \sqrt{k}\, 10^{1-r}/3$. В этих формулах учтено значение коэффициента K_c, определенного в формуле (2.12б), равное трем, что при нормальном законе распределения полной погрешности соответствует доверительной вероятности 0,997.

Рассмотрим некоторые характерные случаи суммирования чисел. Если слагаемые близки по модулю, но их знаки чередуются, то относительную погрешность результата суммирования y оценивают по формуле

$$\delta y = \sum_{i=1}^{k} |x_i|\, \delta x_i/y + \sum_{j=1}^{k-1} \delta_j \sum_{i=1}^{j+1} x_i/y.$$

При исходных предположениях сумма четного числа слагаемых намного меньше каждого из них, а нечетная — близка к среднему значению одного слагаемого. Поэтому при равных значениях $\delta x_i = \delta x$ и $\delta_j = \delta$ погрешность суммы четного числа слагаемых $\delta y_{\text{чет}} \approx \approx (kx\delta x/y + (k-2)\bar{x}\delta)/2y$, а для нечетного $\delta y_{\text{неч}} \approx k\delta x + (0{,}5k - 1)\delta$. Подобные оценки завышены, особенно при четном k, в чем можно убедиться, рассматривая процесс суммирования более подробно.

Суммирование четного числа близких по модулю членов с различными знаками (вычитание близких чисел) выполняется без погрешности округления с точностью исходных погрешностей операндов и $\delta y_{\text{чет}} = k\bar{x}\delta x/y$. Суммирование нечетного числа $k-1$ членов обычно также выполняется без операционной погрешности* в связи с малостью

---

* Исключение возможно при близости мантисс слагаемых к 10, когда может измениться порядок нечетных сумм по сравнению с порядком слагаемых, например, при сложении 9,9998738 + (—9,6385778) + 9,6934831 = 10,3247791.

результата и $\delta y_{\text{неч}} \approx k\delta x$. В рассматриваемом случае к погрешностям операндов применимо статистическое оценивание, причем (даже в том случае, когда операнды округляются перед вычислениями), вследствие чередования знаков погрешности компенсируются и среднее значение результирующей погрешности можно принять равным нулю. Однако дисперсию погрешности следует удвоить, так как погрешности однократного округления распределяются в пределах двух единиц младшего разряда представления округленного числа. Не следует выполнять усреднения по старшим разрядам, так как получаемая оценка основана на абсолютных погрешностях суммирования, не зависящих от содержимого старших разрядов. Таким образом при четном числе членов $\overline{\delta y}_{\text{чет}} \approx K_{\text{с}} \sqrt{k}\bar{x}\delta x/y\sqrt{\sigma}$ и нечетном $\overline{\delta y}_{\text{неч}} \approx$ $\approx K_{\text{с}}\delta x/\sqrt{\sigma}$, где для $P_{\text{дов}} = 0{,}997$ следует принять $K_{\text{с}} = 3$.

Рассмотрим часто встречающееся вычисление геометрической прогрессии

$$y = a_0 + a_0 q + \ldots + a_0 q^{k-1} = \sum_{i=0}^{k-1} a_0 q^i.$$

Составив граф распространения относительной погрешности при $\delta_i = \delta$ и $\delta x_i = \delta x$, получим

$$\delta y = \delta x + ((1+q) + (1+q+q^2) + \ldots + (1+q+ \\ + \ldots + q^{k-1}))\, a_0\delta/y = \delta x + \delta(k - 1 - q^2(1-q^{k-1})/(1-q))/(1-q^k). \quad (2.13)$$

Если $0 < q < 1$, $k \gg 1$ и $q^k \ll 1$, то эту мажоритарную оценку можно упростить, приняв $\delta y \approx \delta x + k\delta$.

Следует учитывать, что реальное число k убывающих членов суммы ограничено разрядностью r' микрокалькулятора, так как при отличии очередного члена на r' порядков от накопленной суммы он попадает в область относительного машинного нуля. Предельное число членов для убывающей прогрессии

$$k_{\max} = \mathrm{E}(1 - q - \lg(1-q))/(\lg q) + 1$$

и, например, при $r = r' = 8$ и $q = 0{,}9$ предельное число $k_{\max} = 132$. В этом случае возникает методическая погрешность $\delta_{\text{м}} = y_{\text{ост}}/y = q^{k_{\max}} \approx 10^{1-r}/(1-q)$. При $k \gg 1$ допустимо использование статистических оценок, так как вклад погрешностей округления большинства операций сложения оказывается примерно равноценным.

Рассмотрим суммирование членов геометрической прогрессии, начиная с меньших. Приняв в формуле (2.13) $|q| > 1$, можно при $|q^k| \gg 1$ и $|q^k| \gg k$ принять упрощенное выражение для оценки $\delta y \approx$ $\approx |(1-q)/(1-|q|)|\,\delta x + \delta|q|/(|q|-1)$, но при $q \approx 1$ такая оценка становится завышенной. В рассматриваемом случае также допустимо статистическое оценивание, расчетные формулы для которого несложно вывести по аналогии с предыдущими.

При вычислениях с ограниченной разрядностью погрешность суммы зависит от порядка слагаемых. Например, при вычислении суммы 10 членов геометрической прогрессии с $a_0 = 1$ и $q = (1/3)$, на-

чиная со старших членов, согласно формуле (2.13) получим $\delta y \approx$ $\approx \delta x + \delta (9 - (1 - (1/3)^9)/(1 - 1/3)9)/(1 - (1/3)^{10}) \approx \delta x + 8{,}83\delta$. Если же изменить порядок суммирования, начиная с его меньших членов (что эквивалентно суммированию членов возрастающей прогрессии с $a_0 = (1/3)^9$ и $q = 3$), то по той же формуле получим оценку погрешности $\delta y \approx \delta x + \delta (9 - 9(1 - 3^9)/(1 - 3))/(1 - 3^{10}) \approx$ $\approx \delta x + 1{,}5\delta$, примерно в шесть раз меньшую.

В практических задачах встречается суммирование членов знакочередующихся рядов, вначале возрастающих по модулю, а затем убывающих с некоторого члена. В этом случае допустимо статистическое оценивание полной погрешности суммы y по формуле

$$\delta y \approx K_c |x_{max}| (\delta x + \delta)/y,$$

где $|x_{max}|$ — модуль наибольшего члена ряда, а коэффициент K_c зависит от выбранной доверительной вероятности и скорости изменения модуля членов ряда (для нижней границы ожидаемой ошибки округления можно принять $K_c = 1$). Проверка этой оценки при $K_c = 1 \ldots 3$ на многочисленных примерах подтверждает ее близость к реальным значениям погрешности результата.

Приведенные результаты анализа влияния погрешностей округления можно распространить и на суммирование произвольных последовательностей слагаемых, воспользовавшись приведенными оценками для близких по структуре задач. Подобным образом можно анализировать распространение погрешностей исходных данных и операционных погрешностей и для других вычислительных схем. Однако этот анализ применим лишь в том случае, когда ограничение ряда (2.9) приводит к допустимой методической погрешности формул (2.10) и (2.11).

Способы оценки методической погрешности зависят от порождающего ее вычислительного процесса и особенностей задачи. Так, при суммировании членов бесконечного сходящегося ряда эта погрешность равна сумме остаточных членов ряда, но способ ее оценки зависит от свойств ряда. Если ряд с убывающими по модулю членами знакопеременный, то методическая (остаточная) погрешность суммирования не превышает первого из отбрасываемых членов, но для рядов с другими свойствами способы оценки остатка более сложные [4, 6, 7]. В частности, погрешности ограничения ряда (2.9) зависят от погрешностей исходных данных.

При вычислениях на ЭВМ точность результатов вычислений часто оценивают, выборочно сравнивая их с результатами повторных вычислений при кратном увеличении разрядности. Однако такая проверка обеспечивает лишь оценку влияния погрешностей округления, тогда как при инженерных расчетах обычно преобладает влияние погрешностей исходных данных.

Влияние относительно больших погрешностей исходных данных, при которых формулы (2.10) и (2.11) недостоверны, обычно оценивают статистическими методами, но при небольшом числе исходных данных применяют и прямые способы оценки погрешности результата. К ним относится метод, основанный на вычислении предельных значений

результата $f_{\min}$ и $f_{\max}$ при соответствующих наихудших сочетаниях граничных значений $x_{j\min}$ и $x_{j\max}$ неточных исходных данных. Для этого определяют знаки производных $f' = \partial f/\partial x_j$ и при вычислении $f_{\min}$ принимают значения $x_{j\min}$, если $f' > 0$ или $x_{j\max}$ в противном случае, а при вычислении $f_{\max}$ принимают противоположные граничные значения каждой переменной x_j. Точное значение результата f находится между значениями $f_{\min}$ и $f_{\max}$, погрешности вычисления которых оценивают согласно формулам (2.10) или (2.11) по малым погрешностям вычислений и остальных исходных данных.

Для примера оценим точность результата вычислений по формуле $y = a - \ln(b/c)$ при номинальных значениях $a = 3$, $b = 20$, $c = 1$, допуске 20 % на эти значения ($\Delta a = 0{,}6$; $\Delta b = 4$; $\Delta c = 0{,}2$), предельной погрешности вычисления логарифма $\Delta = 1 \cdot 10^{-5}$ и предельной погрешности результатов арифметических операций $\Delta = 1 \cdot 10^{-7}$. Составив согласно данным табл.3 граф накопления абсолютных погрешностей (рис. 17,*а*), соответствующий формуле (2.10), найдем с его помощью мажоритарную оценку $\Delta y \leqslant \Delta a + \Delta_3 + \Delta_2 + (c/b)(\Delta_1 + \Delta b/c + b\Delta c/c^2) \approx 1{,}1$, откуда при значении $y = 0{,}0042678$ при номинальных значениях исходных данных получаем граничные значения $y_{\min} = -1{,}0957322$ и $y_{\max} = 1{,}1042678$.

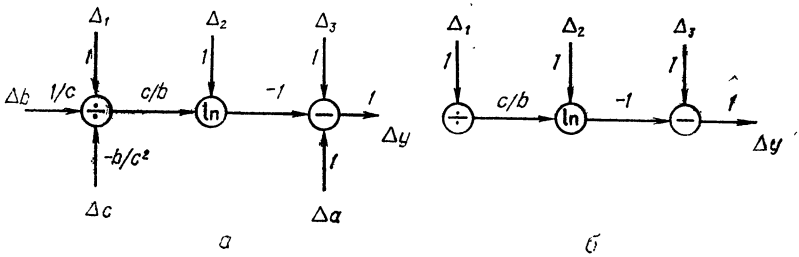


Рис. 17

По методу наихудших сочетаний непосредственно находим $y_{\min} = a_{\min} - \lg(b_{\max}/c_{\min}) = -0{,}7011973$ и $y_{\max} = a_{\max} - \lg(b_{\min}/c_{\max}) = 0{,}7097329$. С помощью графа накопления абсолютных погрешностей при номинальных значениях исходных данных ($\Delta a = \Delta b = \Delta c = 0$) находим мажоритарную оценку погрешности (рис. 17, *б*) вычисления этих граничных значений: $\Delta y_{\text{гр}} \leqslant \Delta_3 + \Delta_2 + \Delta_1 c/b \approx 1 \cdot 10^{-5}$.

Вычисление погрешностей согласно формуле (2.10) в рассмотренном случае привело к завышенной оценке. Однако и метод наихудших сочетаний дает завышенную оценку по сравнению со статистическими методами, так как вероятность того, что исходные данные одновременно принимают расчетные граничные значения, пренебрежимо мала.

Особого внимания требует оценка точности решения (корней) систем уравнений (2.6), используемых для моделирования свойств физических объектов. Если коэффициенты и свободные члены таких уравнений заданы точно, то при единственности решения системы оно соответствует (с точностью до ошибок вычислений) равенству нулю всех невязок уравнений

$$F_j = \sum_{i=1}^{n} (a_{ji}x_i - q_j),\ j = 1,2,\ \ldots,\ n$$

при замене переменных x_i вычисленными значениями корней. Однако при погрешностях исходных данных корни системы уравнений находятся в некоторой области пространства переменных x_i и задача оценки погрешности решения системы заключается в определении границ этой области. Эта задача может оказаться особенно существенной для систем уравнений, точность решения которых сильно зависит от погрешностей исходных данных. Такие системы, как и матрицы их коэффициентов, называют *плохо обусловленными*.

Причины плохой обусловленности удобно рассмотреть на примере систем из двух линейных уравнений, наглядно отображаемых прямыми линиями на плоскости переменных. При точно заданных исход-

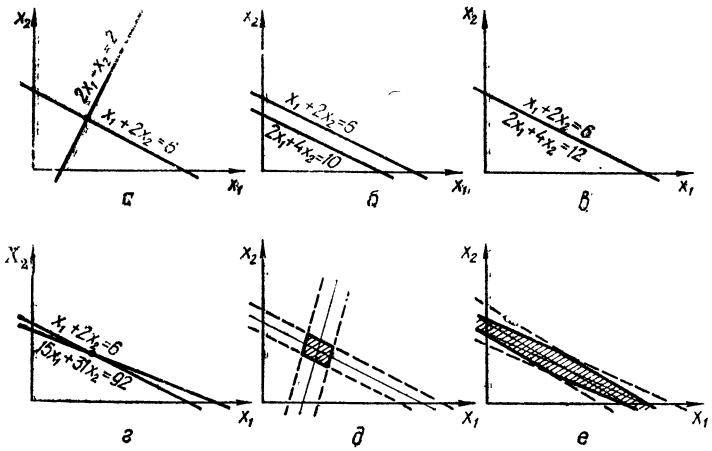


Рис. 18

ных данных невырожденная система линейных уравнений имеет единственное решение, соответствующее пересечению линий уравнений (рис. 18,*а*), а вырожденная система — ни одного решения, если эти линии параллельны (рис. 18,*б*), или бесконечное число решений, если они совпадают (рис. 18,*в*).

Изменение свободных членов уравнений приводит к параллельному смещению прямых, отображающих уравнения, и для хорошо обусловленной системы — к некоторой конечной области решений (рис. 18,*д*). Область решения расширяется при погрешностях коэффициентов уравнений, приводящих к изменению наклона соответствующих прямых линий. Плохая обусловленность характерна для почти вырожденных систем уравнений, линии которых почти совпадают (рис. 18, *г*). При точных исходных данных и отсутствии погрешностей вычислений такая система имеет единственное решение, но при небольших погрешностях вычислений и, особенно, исходных данных область решений значительно расширяется, а при некоторых значениях исходных данных решений становится бесконечно много или они отсутствуют.

Например, плохо обусловленная система уравнений

$$x_1+2x_2=6,\quad 15x_1+31x_2=92$$

имеет корни $x_1 = 2$, $x_2 = 2$, но при изменении свободных членов менее чем на 1 %, например, приводящем к системе

$$x_1 + 2x_2 = 5{,}96; \quad 15x_1 + 31x_2 = 92{,}38,$$

решение определяется значениями $x_1 = 0$, $x_2 = 2{,}98$ (рис. 18, *е*).

В общем случае точность решения систем уравнений оценивают аналитическими методами [1, 4], но для линейных систем уравнений такая оценка возможна и в процессе вычислений [7].

4. ПОВЫШЕНИЕ ТОЧНОСТИ РЕЗУЛЬТАТА ВЫЧИСЛЕНИЙ

Удачный выбор алгоритма и программы вычислений, тщательная подготовка исходных данных и контроль накопления погрешностей во многих случаях позволяют существенно повысить точность инженерного расчета. При вычислениях на микрокалькуляторах и ЭВМ других классов приходится прежде всего учитывать, что справедливые для операций над точными числами законы ассоциативности, дистрибутивности и коммутативности нарушаются при ограничении разрядности операндов и результатов промежуточных вычислений. Поэтому полная погрешность результата вычислений может зависеть от последовательности выполнения операций или порядка ввода операндов в вычисления. Подобная ситуация рассматривалась при оценке операционной составляющей результата суммирования, обобщая которую, можно сформулировать следующее правило: для уменьшения операционной погрешности суммирование следует начинать с меньших чисел. Например, вычисляя сумму чисел $0{,}00877 + 0{,}00959 + 10{,}00388 + 100{,}44785 + 100000{,}48$ в порядке записи слагаемых на микрокалькуляторе с $r = r' = 8$ и округляя отбрасыванием, получаем 100110,95. Если же изменить порядок суммирования на обратный, то получим 100110,92. Эта незначительная для нескольких слагаемых погрешность может существенно накопиться при большом числе и, особенно, различных (не чередующихся) знаках слагаемых.

Точность суммирования можно повысить еще больше, разбивая слагаемые на группы чисел, близких по порядку, и складывая результаты сложения в группах, начиная с меньших по модулю.

Перестановка сомножителей при вычислениях на микрокалькуляторе с показательной формой представления чисел обычно меньше влияет на погрешность произведения, так как изменяется лишь характер округления мантисс частных произведений, если они попадают в диапазон представления чисел микрокалькулятора. Так, на микрокалькуляторе со стандартным (научным) показательным представлением чисел получим:

$$1{,}19 \cdot 10^{-5} \times 1{,}239 \cdot 10^{-1} \times 1{,}3459 \cdot 10^{4} \times 1{,}45679 \cdot 10^{2} = (2{,}8908663);$$
$$1{,}45679 \cdot 10^{2} \times 1{,}19 \cdot 10^{-5} \times 1{,}3459 \cdot 10^{4} \times 1{,}239 \cdot 10^{-1} = (2{,}8908662);$$
$$1{,}45679 \cdot 10^{2} \times 1{,}3459 \cdot 10^{4} \times 1{,}239 \cdot 10^{-1} \times 1{,}19 \cdot 10^{-5} = (2{,}8908661).$$

При большом числе операций умножения и показательной форме представления чисел погрешность округления накапливается, но переста-

новка сомножителей незначительно изменяет окончательный результат, в чем можно убедиться, анализируя граф накопления погрешностей при последовательном умножении. В значительно большей степени изменение порядка ввода сомножителей влияет на результат при вычислениях на микрокалькуляторе с естественной формой представления чисел, когда порядок промежуточных произведений оказывается отрицательным, что приводит к увеличению относительных погрешностей промежуточных произведений. Действительно, выполнив на таком микрокалькуляторе описанные выше операции умножения, получим соответственно произведения 2,7449711; 2,8907327 и 2,8908661, отличающиеся уже во втором знаке.

Порядок промежуточных произведений можно увеличить, начиная умножение с наибольших сомножителей, но в этом случае возникает опасность переполнения запоминающих устройств микрокалькулятора. Поэтому обычно придерживаются следующего правила: умножают наибольший множитель на наименьший, и если результат меньше единицы, умножают его на наибольший из оставшихся множителей, в противном случае — на наименьший, продолжая подобный выбор сомножителей до получения окончательного результата.

Нарушение этого правила при умножении чисел различного порядка может привести к принципиальным ошибкам (вследствие попадания промежуточных результатов в область машинных нуля или бесконечности) даже при вычислениях на микрокалькуляторах с показательной формой представления чисел. Свидетельством служат следующие результаты умножения на микрокалькуляторе со стандартным представлением чисел при восьми разрядах мантиссы и двух разрядах порядка:

$$2 \cdot 10^{-96} \times 0{,}0001 \times 3 \cdot 10^{95} \times 140000 = (0);$$
$$3 \cdot 10^{95} \times 140000 \times 2 \cdot 10^{-96} \times 0{,}0001 = (\infty);$$
$$3 \cdot 10^{95} \times 2 \cdot 10^{-96} \times 140000 \times 0{,}0001 = (8{,}4).$$

Умножать числа различного порядка на микрокалькуляторе с естественной формой их представления практически возможно лишь после их записи в показательной форме. Тогда окончательный результат определяется произведением мантисс чисел и сложением их порядков. Этот способ, позволяющий избежать как переполнения, так и увеличения операционной погрешности или попадания результата в область машинного нуля, целесообразно использовать и в тех случаях, когда промежуточные результаты не выходят за пределы диапазона представления чисел в микрокалькуляторе. Например, непосредственно вычисляя на восьмиразрядном микрокалькуляторе $0{,}0135 \times \times 0{,}0105 \times 0{,}00543 = (0{,}0000004)$, получим результат с большой относительной погрешностью. Если же записать $1{,}35 \cdot 10^{-2} \times 1{,}05 \times \times 10^{-2} \times 3{,}43 \cdot 10^{-3}$, умножить мантиссы и сложить порядки, то результат вычислений $4{,}8620250 \cdot 10^{-7}$ будет содержать семь верных цифр вместо одной в результате непосредственного умножения.

Значительные отличия порядков операндов чаще всего связаны с неудачным выбором единиц измерения соответствующих физических величин. Поэтому перед началом вычислений целесообразно (а иног-

да — необходимо) выполнить нормирование переменных и коэффициентов, входящих в расчетные выражения. Нормирование заключается в замене искомого результата, переменных или параметров их линейными функциями и позволяет уравнять порядки (приблизив их к нулю) входящих в расчетные формулы чисел. Для физических величин нормирование имеет смысл перехода к более удобным единицам измерения. Нормирование не только позволяет избежать переполнения или ошибочного попадания в область машинного нуля, но, как правило, обеспечивает уменьшение операционных погрешностей.

Например, перед решением дифференциального уравнения $d^2x / dt^2 - 30000\,(1 - 25x^2)dx / dt + 10^{10}x = 0$ целесообразно использовать подстановку $\tau = t\ 10^5$, соответствующую переходу от основной единицы времени в секундах к меньшей в 10^5 раз единице измерения (десятки миллисекунд). В этом случае

$$dx / dt = (dx / d\tau)(d\tau / dt) = 10^5 dx / d\tau; \quad d^2x / dt^2 = (d((dx / d\tau)(d\tau / dt)) / d\tau) \times d\tau / dt = 10^{10} d^2x / d\tau^2$$

и после подстановки производных и деления правой и левой частей уравнения на 10^{10} получим $d^2x / d\tau^2 - 0{,}3\ (1 - 25x^2)\ dx / d\tau + x = 0$. Если исследуется только характер изменения процесса во времени, то целесообразно сократить число коэффициентов заменой $z = 5x$. Тогда $dx / d\tau = 0{,}2dz / d\tau$; $d^2x / d\tau^2 = 0{,}2d^2z / d\tau^2$ и, умножив на 5 правую и левую части уравнения, получим

$$d^2z / d\tau^2 - 0{,}3\ (1 - z^2)dz / d\tau + z = 0.$$

При решении заданного уравнения в такой форме операционные погрешности снижаются в связи с уменьшением числа операций, а также количества коэффициентов и их величин.

Операционные погрешности результатов расчета часто удается уменьшить, выбрав расчетную формулу с наименьшим числом операций. Типичным примером может служить вычисление степенного многочлена

$$A(p) = \sum_{i=0}^{n} a_i p^i = a_n p^n + a_{n-1} p^{n-1} + \cdots + a_1 p + a_0$$

после его преобразования по формуле

$$A(p) = ((\ldots(a_n p + a_{n-1})\, p + a_{n-2})\, p + \cdots + a_1)\, p + a_0,$$

обеспечивающей минимальное число операций и, соответственно, минимальную операционную погрешность результата. Точность этой формулы не изменяется и при ее представлении в итерационной форме

$$A_k = A_{k-1} p + a_{n-k};\ k = 1, 2, \ldots, n, \qquad (2.14)$$

когда для вычисления A_1 принимают $A_0 = a_n$, а полученное после выполнения n итераций значение A_n равно искомому значению многочлена.

При подобных преобразованиях алгоритма вычислений следует также избегать выполнения операций, результаты которых отличаются высокой чувствительностью к погрешностям операндов. К таким операциям, как видно из табл. 3, относятся вычисление arc sin x и arc cos x при $x \approx 1$, а также часто встречающаяся операция вычита-

ния при $x_1 \approx x_2$. В таких случаях переход к расчетным формулам даже с большим числом операций, но позволяющих избежать резкого увеличения погрешности отдельных результатов промежуточных операций, может обеспечить меньшее значение полной погрешности окончательного результата вычислений. Например, для уменьшения погрешности результата вычитания близких (при $a \gg \alpha$) чисел целесообразно устранить вычитания, преобразовав расчетные формулы:

$$(a+\alpha)^2 - a^2 = \alpha(2a+\alpha);$$
$$\sqrt{a+\alpha} - \sqrt{a} = \alpha/(\sqrt{a+\alpha} + \sqrt{a});$$
$$a - \sqrt{a^2+\alpha} = -\alpha/(a + \sqrt{a^2+\alpha});$$
$$1 - a/(a+\alpha) = \alpha/(a+\alpha).$$

В качестве примера рассмотрим часто встречающееся на практике вычисление вещественных корней квадратного уравнения $a_2x^2 + a_1x + a_0 = 0$ по формуле вида

$$x_{1,2} = \pm (|\alpha| \pm \sqrt{\alpha^2+\beta}),$$

где знак перед скобкой выбирают соответствующим знаку α.

Больший по модулю вещественный (при $\alpha^2 + \beta > 0$) корень целесообразно вычислять по исходной формуле

$$x_1 = \pm (|\alpha| + \sqrt{\alpha^2+\beta}),$$

но при $\alpha^2 \gg |\beta|$ вычисление меньшего корня по формуле

$$x_2 = \pm (|\alpha| - \sqrt{\alpha^2+\beta})$$

будет сопровождаться значительной относительной погрешностью. Для ее оценки предположим, что исходные данные заданы точно ($\delta\alpha = \delta\beta = 0$), а погрешности округления результата каждой операции одинаковы и равны 10^{1-r}. Тогда по графу накопления относительной погрешности (см. рис. 14,*б*) получим

$$\delta x_2 \leqslant |10^{1-r}(1 + (4\alpha^2+3\beta)/2\sqrt{\alpha^2+\beta}\,(|\alpha| - \sqrt{\alpha^2+\beta})|$$

и при $\alpha^2/|\beta| = z \gg 1$ предельная относительная погрешность $\delta x_2 \leqslant 10^{1-r}4z$. Так, при $r = 8$, $z = 10^6$ по этой оценке относительная погрешность корня может достигать 40 %.

Поэтому при $z \gg 1$ целесообразно вычислять меньший корень по преобразованной формуле

$$x_2 = \pm \beta/(|\alpha| + \sqrt{\alpha^2+\beta}) = \beta/x_1,$$

что уменьшит погрешность.

Например, при решении уравнения $x^2 + 20000x + 8 = 0$ на микрокалькуляторе с $r = 8$ и округлением результата вычитания отбрасыванием по исходной формуле получим $x_1 = -10000 - \sqrt{10000^2 - 8} = -20000$; $x_2 = -10000 + \sqrt{10000^2 - 8} = 0$ при невязке уравнения $A(x_{1,2}) = 8$. При вычислении меньшего корня по преобразованной формуле получим $x_2 = -8/20000 = -4 \cdot 10^{-4}$ с невязкой $A(x) = 0$. Следовательно, абсолютная погрешность меньшего корня, вычисленного по исходной формуле, составляет $\Delta x_2 = 4 \cdot 10^{-4}$, а относительная погрешность $\delta x_2 = \Delta x_2/x_2$ стремится к бесконечности.

Если условие $z \gg 1$ не выполняется, вычисление меньшего корня по исходной формуле может оказаться более точным, так как при вычислении этого корня по преобразованной формуле погрешность результата будет зависеть как от погрешности вычисления большего корня, так и от погрешности округления результата деления. Следо-

вательно, в рассмотренном случае целесообразно разветвлять алгоритм вычисления меньшего корня в зависимости от выполнения условия $z \gg 1$, что обеспечит снижение погрешностей результата вычислений при решении с помощью этого алгоритма различных уравнений второго порядка. Разветвление алгоритма в зависимости от величин исходных данных позволяет уменьшить погрешность результата вычислений и во многих других случаях.

В качестве примера рассмотрим вычисление функции $y = \sin x / x$ на микрокалькуляторе «Электроника Б3-21» в произвольном интервале значений аргумента. В подобных случаях рекомендуют разветвлять алгоритм вычислений, выделяя значение $x = 0$, для которого $y(0) = 1$. Однако, если значение аргумента отличается от нуля, но достаточно мало́, то относительная погрешность результата может оказаться недопустимо большой. Причина заключается в том, что, хотя погрешность округления $\Delta \ll 10^{-7}$, микрокалькулятор заданного типа вычисляет функцию $\sin x$ при малых значениях аргумента в естественной форме с относительно большой операционной погрешностью $\delta_ф \approx \approx 10^{-7}/\sin x$. Например, при $x = 0{,}0001^0$ на этом микрокалькуляторе получим | 0,0001 | ↑ | π | × 1 8 0 | ÷ | ↑ | sin | XY | ÷ | (0,9740282) с погрешностью $\delta \approx 3\,\%$.

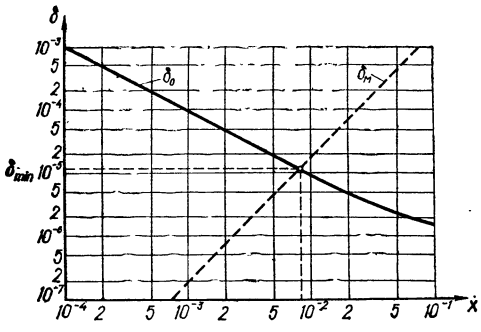


Рис. 19

Полная операционная погрешность результата вычисления функции $y = \sin x/x$ определяется выражением $\delta \leqslant \delta_0 + \delta_ф = 1 \times \times 10^{-7}(1 + |\sin x|)/|\sin x|$ и при малых значениях аргумента в основном зависит от операционной погрешности вычисления $\sin x$. Если при $x \ll 1$ не вычислять $\sin x$ вводом оператора sin, а воспользоваться приближением $\sin x \approx x$ с методической погрешностью $\delta_м \approx x^2/6$, то приближенное значение $y = 1$ с такой же методической погрешностью будет лучшим при всех малых значениях x, для которых $\delta_м \leqslant \delta_ф$. Из графиков зависимости методической погрешности $\delta_м$ и погрешности вычисления функции $\delta_ф$ от значения аргумента x в радианах (рис. 19) следует, что целесообразно разветвить алгоритм вычисления $y = \sin x/x$ с непосредственным вычислением этой функции при $x > 8 \cdot 10^{-3}$ и выбором $y = 1$ при $x \leqslant 8 \cdot 10^{-3}$. Такое разветвление обеспечит погрешность $\delta \leqslant 1{,}2 \cdot 10^{-5}$, меньшую погрешности результата непосредственного вычисления функции при малых значениях аргумента, во всем интервале изменения x.

Решение многих задач связано с применением численных методов, вызывающих методическую погрешность результата вычислений, слагающуюся с операционной составляющей. Теоретически методическую погрешность можно уменьшить до сколь угодно малой величины, увеличивая число операций, но это уменьшение при вычислениях на микрокалькуляторе ограничено как ростом операционной составляющей при увеличении числа операций, так и фиксированной разрядностью результата вычислений.

При разрядности r мантиссы, высвечиваемой на индикаторе, минимальная методическая погрешность не может быть меньше числа,

соответствующего единице младшего разряда мантиссы, но часто оказывается значительно большей.

В качестве примера рассмотрим процесс решения уравнения $x - 7/(1 + x^2)^{1/4} = 0$ методом *простых итераций*. Этот метод заключается в представлении исходного уравнения $f(x) = 0$ в форме $x_{i+1} = \varphi(x_i)$, выборе начального значения $x_i = x_0$ и последующих вычислениях правой части формулы с подстановками $x_i = x_{i+1}$ до получения значений x_{i+1} и x_i с заданным одинаковым числом значащих цифр, принимаемых в качестве верных и определяющих точность вычисления корня.

Переписав рассматриваемое уравнение в виде $x_{i+1} = 7/(1 + x_i^2)^{1/4}$ и приняв $x_0 = 1$ в качестве начального приближения, при вычислениях на восьмиразрядном микрокалькуляторе с оператором $\sqrt{}$ и округлением отбрасыванием последовательные приближения $x_1 = 5{,}8862754$; $x_2 = 2{,}8647619$, ..., $x_{14} = 3{,}6145184$. Повторяя итерации достаточно большое число раз и прекращая вычисления при совпадении в очередных результатах всех значащих цифр, можно свести методическую погрешность к одной единице последнего разряда мантиссы. Однако, продолжая вычисления, получим $x_{22} = 3{,}6146014$; $x_{23} = 3{,}6146018$; $x_{24} = 3{,}6146016$; $x_{25} = 3{,}6146018$ с последующим периодическим повторением двух значений, отличающихся двумя единицами последнего разряда.

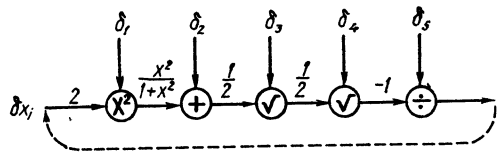


Рис. 20

Причина заключается в операционной погрешности вычислений на каждой итерации $\delta x_{i+1} = \delta x_i(-x_i^2/2(1 + x_i^2)) + \delta_1(-x^2/4(1 + x^2)) + \delta_2(-1/4) + \delta_3(-1/2) - \delta_4 + \delta_5$, которую несложно оценить по графу накопления относительной погрешности (рис. 20). Для достаточно больших i можно принять $x_i \approx 3{,}61$ и уточнить погрешности δ_i по мантиссам $1 \leqslant M(a) < 10$: $\delta_1 = 10^{-7}/M(x^2) \approx 10^{-7}/1{,}3 \approx \approx 0{,}77 \cdot 10^{-7}$; $\delta_2 = 0$ (добавление единицы к $x^2 \approx 1\,3$ не изменяет порядка результата); $\delta_3 = 10^{-7}/M\left(\sqrt{1 + x_i^2}\right) \approx 10^{-7}/3{,}75 \approx 0{,}27 \cdot 10^{-7}$; $\delta_4 = 10^{-7}/M \times \times \left(\sqrt{\sqrt{1 + x_i^2}}\right) \approx 10^{-7}/1{,}94 \approx 0{,}52 \cdot 10^{-7}$; $\delta_5 = 10^{-7}/M\left(7/\sqrt[4]{1 + x^2}\right) \approx 10^{-7}/3{,}61 \approx \approx 0{,}28 \cdot 10^{-7}$. Тогда $\delta x_{i+1} \approx -0{,}46\delta x_i - 0{,}56 \cdot 10^{-7}$ с сохранением знаков, так как при округлении отбрасыванием знаки погрешностей одинаковы.

Допустим, что после k-й итерации погрешность δx_k равна нулю. Тогда на следующей итерации полная погрешность результата $\delta x_{k+1} \leqslant -0{,}56 \cdot 10^{-7}$ и $\Delta x_{k+1} \leqslant -0{,}56 \cdot 10^{-7} x_k = -0{,}56 \cdot 10^{-7} \cdot 3{,}61 \approx -2 \cdot 10^{-7}$, на следующей итерации $\delta x_{k+2} = 0{,}56 \cdot 10^{-7} \cdot 0{,}46 - 0{,}56 \cdot 10^{-7} \approx -0{,}3 \cdot 10^{-7}$ и $\Delta x_{k+2} \leqslant 1 \cdot 10^{-7}$, а в дальнейшем результаты с такими погрешностями будут повторяться.

Обобщая полученные результаты, приходим к выводу, что при использовании итерационных алгоритмов с одинаковыми итерациями достижимая точность результата определяется операционной погрешностью результата вычислений на одном цикле (итерации). Исходя из этой погрешности и следует выбирать критерий прекращения вычислений в цикле. Так, для рассмотренного уравнения вида $x - A/\sqrt[4]{1 + x^2} = 0$ независимо от коэффициента A операционная погрешность результата выполнения итерации $|\delta_1|/4 + |\delta_2|/4 + |\delta_3|/2 + |\delta_4| + |\delta_5|$ или, приняв $\delta_i = 10^{-7}$, получим $\delta_{\text{ит}} \leqslant 3 \cdot 10^{-7}$. Следовательно, вычисления следует прекращать при выполнении условия $|(x_{i+1} - x_i)/x_i| \leqslant 3 \cdot 10^{-7}$, тогда как критерий прекращения $|(x_{i+1} - x_i)/x_i| \leqslant$

$\leqslant 1 \cdot 10^{-7}$ оказывается невыполнимым. Подобная оценка критерия прекращения вычислений особенно существенна при составлении программ автоматических вычислений, так как при неудачном выборе этого критерия программа „зацикливается" и ее выполнение можно прекратить только аварийной остановкой нажатием клавиши С/П.

Если в методах последовательных приближений (к которым можно отнести и последовательное суммирование членов бесконечного убывающего по модулю членов ряда) методическая погрешность уменьшается при увеличении числа последовательно выполняемых циклов (итераций), то во многих других методах уменьшение методической погрешности связано с увеличением числа «параллельно» выполняемых циклов. Типичным примером может служить численное интегрирование, методическая погрешность которого тем меньше, чем на большее число частей разбит интервал интегрирования. Так как искомый интеграл равен сумме «элементарных интегралов», вычисляемых на каждом участке разбиения практически с одинаковой погрешностью, то увеличение числа разбиений приводит к практически пропорциональному росту операционной составляющей результата интегрирования.

При необходимости получения результата с высокой точностью при числе верных цифр, превышающем разрядность r мантиссы, высвечиваемой на индикаторе, следует выполнять вычисления с повышенной разрядностью, разбивая операнды на блоки с числом разрядов, меньшим r.

Для алгебраического суммирования (сложения или вычитания) многозначных или отличающихся по порядку операндов их представления следует разбить на блоки по r—1 разрядов, записав

$$a = \ldots + a_2 10^{2(r-1)} + a_1 10^{r-1} + a_0;$$
$$b = \ldots + b_2 10^{2(r-1)} + b_1 10^{r-1} + b_0,$$

после чего выполнить на микрокалькуляторе суммирование отдельных блоков одинакового порядка

$$a + b = \ldots + (a_2 + b_2) 10^{2(r-1)} + (a_1 + b_1) 10^{r-1} + (a_0 + b_0),$$

предварительно уравняв порядки операндов. Например, при сложении чисел 13678500231 и 12,135689821 следует уравнять их порядки, записав, например $a = 13678500231000000000 \cdot 10^{-9}$ и $b = 1213135689821 \cdot 10^{-9}$ и, разбив на блоки по 7 разрядов мантиссы слагаемых, вычислить $a + b = (136785 \cdot 10^{14} + (23100 + 1213) \cdot 10^{7} + 5689821) \cdot 10^{-9} = 13678500243135689821 \cdot 10^{-9} = 13678500243,135689821$.

При умножении на 8-разрядном микрокалькуляторе операндов с восемью значащими цифрами мантиссы результат может содержать не более восьми из 15 или 16 верных цифр результата. При необходимости сохранения всех верных цифр такого произведения его также можно вычислить по частям, представив сомножители в показательной

форме с целыми мантиссами, разбитыми на два блока по четыре (или менее в старшем блоке) разряда. Тогда произведение $ab = (a_1 10^4 + a_0) 10^{na} (b_1 10^4 + b_0) 10^{nb} = (a_1 b_1 10^8 + (a_1 b_0 + a_0 b_1) 10^4 + a_0 b_0) 10^{na+nb}$.

Частные произведения $a_i b_j$ в таком выражении содержат не более восьми цифр и, следовательно, могут быть точно вычислены на микрокалькуляторе с разрядностью $r \geqslant 8$. Например, для получения точного произведения чисел $a = 326{,}58795$ и $b = 0{,}0526786$ с помощью микрокалькулятора любого типа несложно вычислить $ab = (3265 \cdot 10^4 + 8795) 10^{-5} (52 \cdot 10^4 + 6786) 10^{-7} = (169780 \cdot 10^8 + (22156290 + 457340) 10^4 + 59682870) 10^{-12} = 17204195982870 \cdot 10^{-12} = 17{,}20419598287$.

По аналогичной схеме число, представление $a = (a_1 10^4 + a_0) 10^n$ которого содержит от пяти до восьми цифр, может быть точно возведено в квадрат $a^2 = (a_1^2 10^8 + 2a_1 a_0 10^4 + a_0^2) 10^{2n}$, представление которого содержит до 16 цифр.

С помощью микрокалькулятора удобно находить с высокой точностью частное от деления двух чисел или результат вычисления обратной величины. Для этого следует воспользоваться обычным способом деления двух многозначных чисел с записью на бумаге промежуточных результатов, определяя на каждом шаге не одну значащую цифру частного, а такое их количество, при котором результат умножения на делитель не округляется микрокалькулятором. Разбивая делитель и делимое на блоки, содержащие такое число разрядов, результат операций над которыми не округляется микрокалькулятором, можно с высокой точностью выполнять деление чисел с количеством разрядов, значительно превосходящим рабочую разрядность микрокалькулятора.

Вычисление результата арифметических операций с двойной точностью по рассмотренным схемам несложно автоматизировать с помощью программируемого микрокалькулятора, размещая отдельные блоки представления чисел в различные регистры памяти. По аналогичным схемам можно организовать вычисление результатов арифметических операций и с большей разрядностью.

5. ОБРАТНАЯ ЗАДАЧА АНАЛИЗА ПОГРЕШНОСТЕЙ

Точность численных результатов инженерного расчета даже при достаточно малых погрешностях округления и методических погрешностях может оказаться ниже требуемой, которая равна точности определения соответствующих физических величин или, в противном случае, оговаривается в техническом задании на проектирование. В этом случае возникает обратная задача анализа погрешностей или задача оптимизации погрешностей — определение предельно допустимых погрешностей математической модели и вычислений, обеспечивающих заданную точность результатов расчета. Эта же задача возникает, когда точность результата выше требуемой и можно увеличить предельные погрешности (допуски) параметров проектируемого объекта, соответственно снизив стоимость его изготовления.

Когда погрешности исходных данных и результатов операций достаточно малы и допустимо ограничение ряда (2.9) линейными членами, предельные абсолютные или относительные погрешности исходных данных выбирают по заданным погрешностям Δf или δf результата расчета, исходя из различных условий. Простейшее из них заключается в равенстве вкладов $|S_1|\,\delta x_1 = |S_2|\delta x_2 = \ldots = |S_n|\delta x_n$ в относительную погрешность результата расчета, откуда в соответствии с формулой (2.11) допустимые значения относительных погрешностей исходных данных и результатов операций

$$\delta x_i = \delta f/n|S_i|. \qquad (2.15а)$$

В тех случаях, когда точность параметров моделируемого объекта оценивают относительными погрешностями, при выборе условия $\delta x_1 = \delta x_2 = \ldots = \delta x_n$ из формулы (2.11) следует

$$\delta x_i = \delta f/\sum_{i=1}^{n} |S_i|. \qquad (2.15б)$$

Иногда точность изготовления компонентов удобнее оценивать по абсолютным погрешностям и при равных вкладах $|f_1'|\Delta x_1 = |f_2'|\Delta x_2 = \ldots = |f_n'|\Delta x_n$ в заданную абсолютную погрешность Δf результата расчета из формулы (2.10) следует

$$\Delta x_i = \Delta f/n|f_i'|. \qquad (2.15в)$$

Если все переменные x_i имеют одинаковую размерность, может оказаться удобным выбирать их предельные погрешности по условию $\Delta x_1 = \Delta x_2 = \ldots = \Delta x_n$. В этом случае из формулы (2.10) следует

$$\Delta x_i = \Delta f/\sum_{i=1}^{n} |f_i'|. \qquad (2.15г)$$

Практически достижимая точность представления чисел, отображающих физические величины, ограничена возможной точностью измерения последних или точностью изготовления моделируемых объектов. Во многих случаях требуемые по формулам (2.15) точности исходных данных оказываются практически недостижимыми, тогда как другие исходные данные могут быть заданы более точно, чем требуется по этим формулам. Поэтому обычно используют различные комбинации формул (2.15), разбивая исходные данные на группы с одинаковыми погрешностями или их одинаковыми вкладами в результирующую погрешность, а при использовании формулы (2.15г) — на группы исходных данных с одинаковой размерностью и точностью.

В качестве иллюстрации рассмотрим задачу определения допустимых погрешностей высоты h, радиуса $r = 10$ см и плотности материала $m = 7{,}96$ Г/см^3 при изготовлении цилиндра с массой $P = \pi r^2 h m = 100$ кг с допустимой погрешностью $\Delta P = 100$ г.

Определив требуемую высоту цилиндра $h = P / \pi r^2 m = 79{,}977 \approx 80$ см, вычислим чувствительности его массы к изменениям параметров: $S_r = (r/P)\partial P / \partial r = 2$; $S_h = (h / P)\partial P / \partial h = 1$; $S_m = (m / P)\partial P / \partial m = 1$. По относительной погрешности массы цилиндра $\delta P = \Delta P / P = 10^{-4}$ в соответствии с формулой (2.15а) получим $\delta r = 1{,}67 \cdot 10^{-5}$; $\delta h = \delta m = 3{,}33 \cdot 10^{-5}$.

При условии, равенства относительных погрешностей параметров по формуле (2.15б)получим $\delta r = \delta h = \delta m = 2 \cdot 10^{-5}$ — в этом случае требования к точности h несколько выше, но предельная погрешность r больше.

При условии равенства вкладов в абсолютную погрешность ΔP для вычисления допустимых погрешностей вычислим значения производных $f'_r = \partial P/\partial r = 2\pi rhm = 40011{,}324$ г/см; $f'_h = \partial P/\partial h = \pi r^2 m = 2500{,}7077$ г/см; $f'_m = \partial P/\partial m = \pi r^2 h = 25132{,}741$ см^3 и по формуле (2.15 в) найдем $\Delta r = \Delta P/3 f'_r = 8{,}33 \cdot 10^{-4}$ см; $\Delta h = \Delta P/3f'_h = 1{,}33 \cdot 10^{-3}$ см; $\Delta m = \Delta P/3f'_m = 1{,}33 \cdot 10^{-3}$ г/см^3.

Предположим, что изменение плотности m материала, из которого изготовляется цилиндр, не превышает $\Delta m = 10^{-3}$ г/см^3. Тогда в соответствии с формулой (2.10) зависящая от параметров r и h составляющая погрешности массы цилиндра $\Delta_1 P = \Delta P - f'_m \Delta m = 74{,}867$ г и согласно формуле (2.15 г) допустимые погрешности $\Delta r = \Delta h = \Delta_1 P/(f'_r + f'_h) = 1{,}76 \cdot 10^{-3}$ см.

При решении рассматриваемой задачи не затрагивался вопрос о возможности измерения параметров с требуемыми по расчету точностями или возможности изготовления цилиндра с требуемыми точностями, однако в инженерных задачах ответы на подобные вопросы имеют первостепенное значение.

При больших отклонениях параметров формулы (2. 15) неприменимы и приходится прибегать к более сложным методам оптимизации допусков. Эта задача становится особенно сложной при плохой обусловленности исходной модели проектируемого объекта в виде системы уравнений (2.6). Плохая обусловленность такой модели приводит к резкому увеличению чувствительности характеристик к изменениям параметров и может являться следствием недостаточной устойчивости свойств проектируемого объекта. Если последняя не предусмотрена техническим заданием, то необходимо так изменить схему и параметры проектируемого объекта, чтобы заданным допустимым отклонениям характеристик соответствовали максимальные допуски на параметры и, следовательно, минимальная стоимость изготовления объекта. Таким образом, обратная задача теории погрешностей при проектировании является практической задачей, в успешном решении которой и заключается инженерное искусство проектировщика.

Глава 3

ПРОГРАММИРОВАНИЕ МИКРОКАЛЬКУЛЯТОРОВ

1. ФОРМА ЗАПИСИ ПРОГРАММ

Составление программ, представляющих алгоритм решения задачи на входном языке микрокалькулятора, называют *программированием*. При решении простейших задач пользователь мысленно составляет программу, выполняя ее в обычном режиме или вводя в программную память нажатием клавиш. При решении более сложных задач ему приходится предварительно составлять программу на бумаге, используя известный алгоритм или разрабатывая его в соответствии с выбранным методом решения и особенностями входного языка микрокалькулятора. В этих случаях приходится многократно переписывать текст программы в поисках наилучшего варианта и поэтому существенное значение приобретает выбор рациональной формы за-

писи, обеспечивающей минимальные затраты времени на составление программы.

В пособиях по программированию микрокалькуляторов и в инструкциях по их применению каждый оператор часто обозначают в прямоугольнике, символизирующем нажимаемую клавишу, а в конце программы и, при необходимости, после некоторых операторов в скобках указывают высвечиваемые результаты вычислений. Для непрограммируемых микрокалькуляторов символы операторов набора чисел обычно записывают слитно, как и при обычной записи десятичных представлений чисел. В процессе составления программы обозначать операторы в прямоугольниках нецелесообразно и достаточно разделять их вертикальными линиями-разделителями или пробелами, например,

$$\text{C } 2{,}7 \times 2 \div 5 = (1{,}08)$$

или, в общем случае произвольных операндов a, b и c,

$$\text{C } a \times b \div c = .$$

Операнды, вызываемые из памяти, обозначают в программах символами операторов вызова, а исходное содержимое регистра X обычно не указывают, например

$$\times\ b \div \text{ИП} = ,$$

но дополняют программу указаниями на исходное содержимое регистров X и памяти.

Операторы, вводимые после нажатия префиксных клавиш, обозначают различными способами. Иногда программу составляют в виде последовательности символов, обозначенных на нажимаемых клавишах. Например, вычисления по формуле $y = \text{arc tg}x + \sqrt{\ln m}$ на микрокалькуляторе «Электроника Б3-36» (см. рис. 7, *в*) при предварительном занесении чисел m и x соответственно в память и регистр X отображают записью

$$\text{arc } 3 + \text{F} = \text{F } 4 \text{ F } 6 = .$$

Подобную запись нельзя рекомендовать, так как она определяет последовательность нажимаемых клавиш, а не алгоритм вычислений, который нельзя прочесть без обращения к клавиатуре микрокалькулятора.

Часто в программах записывают как символы операторов, так и символы нажимаемых для их ввода префиксных клавиш, например,

$$\text{arc tg} + \text{F ИП F ln F } \sqrt{} = .$$

Такая гибридная запись также нецелесообразна, так как обозначение символа оператора над или под клавишей указывает на необходимость предварительного нажатия префиксной клавиши. Аналогично при печатании заглавных букв на пишущей машинке приходится предварительно нажимать «префиксную» клавишу поднятия каретки, но символы этой клавиши в печатаемом тексте не указываются. Поэтому в программах для микрокалькуляторов символы префиксных клавиш

необходимо указывать лишь в тех случаях, когда они являются составными частями символов операторов, занимающих в программе один шаг и записываемых слитно, например, arccos, ГВП, Р6, ИПА, КБП5. Тогда запись программы будет однозначно определять алгоритм вычислений, например

$$\text{arctg} + \text{ИП ln } \sqrt{} = .$$

Неопределенность в выборе символов для обозначения операторов в программах автоматических вычислений возникает и в том случае, когда указатели переходов вводятся (как в микрокалькуляторах с входным языком ЯМК21) нажатием тех же клавиш, что и другие операторы (см. табл. 2). Поэтому в дальнейшем указатели переходов в программах на входном языке ЯМК21 будем обозначать слитной записью символов, обозначенных только на нажимаемых клавишах, а не над или под ними. Например, указатели переходов по адресам 00,32 и 64 будут обозначаться соответственно символами РО, Р×и ГВП, учитывая, что из высвечиваемых в режиме программирования кодов 01,33 и 65 этих указателей два последних совпадают с кодами операторов e^x и $\sqrt{}$.

Для сокращения затрат времени на запись программ целесообразно упрощать символы операторов со сложным начертанием, как указано например в сноске на стр. 33. При составлении программ часто приходится проверять изменения содержимого регистров операционного стека при выполнении операторов. Чтобы не прибегать к громоздкой табличной записи, подобной приведенной в табл. 1, достаточно в нужных местах чернового текста программы выписать в столбец над каждым оператором содержимое регистров операционного стека.

Существенное значение имеет выбор формы записи программ автоматических вычислений. В пособиях и инструкциях по применению программируемых микрокалькуляторов обычно используется табличная форма записи программ, содержащая столбец операторов (шагов программы) и вспомогательную информацию — адреса шагов, высвечиваемые в режиме программирования коды команд, символы нажимаемых клавиш, порядковые номера и описания операций, содержимое регистров операционного стека и числовой памяти. Примером такой записи может служить приведенная в табл. 4 программа вычисления факториала целых чисел по алгоритму, схема которого показана на рис. 3, *б*.

Подобные табличные записи удобны для учебных целей, но громоздкость ограничивает их применение для библиотек с большим количеством программ и практически полностью исключает их использование при составлении программ, когда требуется предельная компактность.

В табличной записи, подобной приведенной в табл. 4, необходимым является лишь столбец операторов, отображающий алгоритм решения задачи, и достаточно выбрать компактную форму записи последовательности этих операторов, удобную для определения адреса любого шага при переходах в разветвленных программах. Для входных языков с передачей управления адресами этому требованию отвечает запись программы по строкам с пробелами между символами команд

4. Программа вычисления факториала целого неотрицательного числа $a > 70$ на входном языке ЯМК21

Инструкция: включить микрокалькулятор, нажать клавиши Р и $\overleftarrow{\text{ШГ}}$, ввести программу в программную память, нажать клавиши Р и $\overrightarrow{\text{ШГ}}$, набрать нажатиями клавиш число a, нажать клавиши В/О и С/П, зарегистрировать высвечиваемый результат вычислений

№ п/п	Адрес шага	Код оператора	Клавиши	Оператор (команда)	Операция	Содержимое регистров	
						X	Y
1	00	59	Р БП	$x = 0$	Проверка условия $x = 0$	a	
2	01	05	F ↑	F↑	Переход по адресу 04	a	
3	02	14	1	1	Набор цифры 1	1	a
4	03	78	С/П	С/П	Стоп	1	a
5	04	71	Р 7	Р7	Засылка в регистр *7*	a	
6	05	81	Р 8	Р8	Засылка в регистр *8*	a	
7	10	82	F 8	F8	Вызов из регистра *8*	a_i	
8	11	14	1	1	Набор цифры 1	1	a_i
9	12	86	—	—	Вычитание	$a_i - 1$	a_i
10	13	81	Р 8	Р8	Засылка в регистр *8*	$a_i - 1$	a_i
11	14	59	Р БП	$x = 0$	Проверка условия $x = 0$	$a_i - 1$	a_i
12	15	23	Р ×	Р×	Переход по адресу 22	$a_i - 1$	a_i
13	20	72	F 7	F7	Вызов из регистра *7*	P_{i-1}	$a_i - 1$
14	21	78	С/П	С/П	Стоп	P_{i-1}	$a_i - 1$
15	22	06	↑	↑	Засылка в регистр Y	$a_i - 1$	$a_i - 1$
16	23	82	F 7	F7	Вызов из регистра *7*	P_{i-1}	$a_i - 1$
17	24	26	×	×	Умножение	P_i	$a_i - 1$
18	25	81	Р 7	Р7	Засылка в регистр *7*	P_i	$a_i - 1$
19	30	58	БП	БП	Безусловный переход	P_i	$a_i - 1$
20	31	11	Р 1	Р1	Переход по адресу 10	P_i	$a_i - 1$

с числом шагов в каждой строке, равным (или меньшим в последней строке) числу ячеек на странице программной памяти. В этом случае адрес ab любого шага программы легко определить по номерам a строки и b столбца, на пересечении которых находится этот шаг.

Составленные и отлаженные программы решения типовых задач для библиотеки пользователя целесообразно хранить в форме, достаточно компактной и удобной для многократного использования программ. В этом случае для облегчения проверки правильности ввода программы в программную память может оказаться желательным дополнение шагов программы их кодами, высвечиваемыми в режиме программирования. Однако для использования этой дополнительной информации приходится обращаться к таблицам соответствия, подобным табл. 2, что связано со значительными затратами времени. Между тем при небольшом навыке работы с микрокалькулятором пользователь запоминает коды наиболее употребительных операторов (в осо-

бенности с помощью описанных далее приемов) и необходимость в таких таблицах и, следовательно, указание кодов операторов в записи программ практически отпадает. Поэтому компактную запись программ, удобную при их составлении и используемую в этой книге, можно рекомендовать и для библиотеки.

При решении задачи с помощью составленной или библиотечной программы автоматических вычислений пользователь программируемого микрокалькулятора должен выполнить следующие действия:

1. Включить микрокалькулятор и ввести команду (РП или ПРГ) перехода в режим программирования. Если микрокалькулятор был включен и программная память не очищена, то перед переходом в режим программирования следует нажать клавишу В/О.
2. Ввести программу в память нажатием соответствующих клавиш.
3. Ввести команду (РР или АВТ) перехода в рабочий режим.
4. Занести в регистры числовой памяти и операционного стека исходные данные.
5. Ввести команду (В/О С/П, С/П или БП *ab*) перехода в режим автоматического выполнения программы (программируемый режим).
6. После выполнения программы зарегистрировать результаты вычислений.

Первые три пункта этого перечня совпадают при выполнении любой программы, но конкретное содержание остальных пунктов определяется программой и должно быть сформулировано в виде инструкции по выполнению программы. При компактной записи библиотечных программ инструкции также должны быть компактными, что сократит время решения задачи, и поэтому необходимо условиться о стандартизации таких инструкций.

В дальнейшем содержимое регистров операционного стека и памяти с произвольной выборкой будем обозначать в инструкциях стандартными символами PN, где N — номер или символ регистра числовой памяти (например, *5* или *Д*) или операционного стека (например, X, Y, Z, T или $X1$). Регистры кольцевого стека памяти в микрокалькуляторах с входным языком ЯМК21 будем обозначать символами CN, где N — номер регистра (см. рис. 11)*.

Размещение исходных данных перед выполнением программы в инструкциях будем описывать формулами (операторами присваивания) вида $a = PN$ или $a = CN$, где a — исходное содержимое регистра N. Размещение результатов вычислений будем описывать в инструкциях формулами вида $PN = a$ или $CN = a$, где a — численное значение результата. Следовательно, запись в инструкции $a = P2$, $b = P7$, $c = C6$, $d = PY$, $e = PX$ (под буквами подразумеваются числа, указываемые в конкретной инструкции) означает, что перед выполнением программы числа a, b, c, d и e должны быть занесены операторами набора чисел и засылки соответственно в регистры *2* и *7* с произвольной выборкой, регистр *6* кольцевого стека памяти, регистры Y и X

* Символы содержимого регистров (например, *P5*) в инструкциях и тексте будут обозначаться наклонным шрифтом в отличие от символов операторов и команд (например, оператора обращения к памяти Р5 или команды С/П), обозначаемых прямым шрифтом.

операционного стека. Аналогично запись в инструкции $PX = f$, $PY = 9$, $P3 = h$, $P6 = k$, $C1 = l$ будет означать, что результаты вычислений f, 9, h, k и l после выполнения программы хранятся соответственно в регистрах X и Y операционного стека, регистрах 3 и 6 с произвольной выборкой и регистре 1 кольцевого стека памяти.

В тех случаях, когда программа при решении задачи выполняется многократно (например, при вычислении функциональных зависимостей) и перед каждым последующим пуском нужно вводить лишь часть новых значений исходных данных, те из них, которые сохраняются после выполнения программы в тех же регистрах, в инструкциях будем заключать в скобки. Например, запись $(a = P1,\ b = P2)\ c = PX$ будет означать, что исходные данные a и b необходимо занести только перед первым выполнением программы, а число c необходимо вводить в регистр X перед каждым пуском.

В инструкциях следует стандартизовать и описание способа пуска программы. В дальнейшем запись в инструкции В/О С/П будет означать, что для пуска программы нужно нажать обе указанные клавиши, а запись (В/О) С/П будет означать, что клавиша В/О должна быть нажата только при первом пуске программы, а при следующих пусках следует нажимать только клавишу С/П. В тех случаях, когда при решении задачи программу пускают различными способами, в инструкции должны быть указаны все эти способы, включая и пуск программы с определенного адреса, отличающегося от адреса 00 первого оператора программы.

Программы библиотеки пользователя должны быть снабжены заголовками и, при хранении в библиотеке программ на различных входных языках, указаниями в заголовке на входной язык. В приводимых далее программах указания на входной язык ЯМК21 или ЯМК34 даны цифрами 21 или 34 соответственно через косую черту после номера программы. В тех случаях, когда программа неприменима для микрокалькуляторов первых выпусков в связи с отмеченными ранее особенностями, эти цифры дополнены символом С.

Программы библиотеки следует также дополнять контрольными примерами, обеспечивающими проверку правильности работы микрокалькулятора и отсутствия ошибок при вводе и пуске программы. В необходимых случаях в контрольных примерах следует указывать приближенное время их выполнения, а в инструкциях — точность результатов вычислений.

Компактность программ на входном языке ЯМК21 повышается при их записи в строки по 12 шагов. Для определения адреса ab шага в такой записи следует учитывать, что левые полустроки по 6 шагов соответствуют страницам программной памяти с четными номерами $a = 0, 2, 4, 6, 8$, а правые — с нечетными номерами $a = 1, 3, 5, 7, 9$. Примером может служить следующая компактная запись программы из табл. 4.

Программа 1/21. Вычисление факториала целого неотрицательного числа $a < 70$

$x = 0$ F↑ 1 С/П Р7 Р8 F8 1 — Р8 $x = 0$ Р$\times$
F7 С/П ↑ F7 $\times$ Р7 БП Р1

Инструкция: a = РХ В/О С/П РХ = a!; при $0 \leqslant a \leqslant 11$ значение a! точное, при $12 \leqslant a < 70$ значение a! содержит не менее шести верных цифр. *Контрольный пример*: время вычисления 10! = = 3628800 около 20 с.

Для определения адресов шагов более удобна компактная запись этой программы на входном языке ЯМК34 по строкам из 10 шагов.

Программа 2/34. Вычисление факториала целого неотрицательного числа $a < 70$

$x = 0$ 04 1 С/П П0 П1 ИП1 1 — П1
$x = 0$ 14 ИП0 С/П ИП0 × П0 БП 06

Инструкция: a = *РХ* В/О С/П *РХ* = a!; при $0 \leqslant a \leqslant 11$ значение a! точное, при $12 \leqslant a < 70$ значение a! содержит не менее шести верных цифр. *Контрольный пример*: время вычисления 10! = 3628800 около 30 с.

Время выполнения контрольных примеров даже на однотипных микрокалькуляторах может оказаться различным. В настоящей книге время выполнения контрольных примеров по программам на входных языках ЯМК21 и ЯМК34 указано соответственно для микрокалькуляторов с красным (светодиоды) и зеленым (газонаполненные трубки) свечением индикаторов, быстродействие которых различно.

Обычно достаточно знать лишь приближенное время выполнения контрольного примера, которое пропорционально числу шагов неразветвленных программ. Необходимость в контроле времени возникает лишь при решении задач методами последовательных приближений с большими затратами времени, когда по времени выполнения контрольного примера можно судить о правильности ввода программы или ее составления. При ошибках в процессе ввода такой программы или неудачном выборе критерия прекращения вычислений в цикле оператор С/П может не считываться и программа «зациклится».

В библиотеке пользователя наряду с программами решения конкретных задач могут храниться базовые программы, предназначенные для решения задач определенного класса (например, различных нелинейных уравнений). В таких базовых программах «сменные» фрагменты заменяют многоточиями.

В тех случаях, когда задача не может быть решена с помощью одной программы и приходится использовать несколько последовательно выполняемых программ, совокупность последних называют *пакетом*. Если программы, входящие в такой пакет, не имеют самостоятельного значения, то они сопровождаются одной инструкцией, но контрольные примеры должны быть составлены для каждой из программ пакета.

2. КРИТЕРИИ ОПТИМАЛЬНОСТИ ПРОГРАММ

Теоретически любую вычислительную задачу можно решить при помощи карандаша и бумаги, а вычислительные средства лишь сокращают время и, следовательно, стоимость вычислений. Основным кри-

терием экономической эффективности применения микрокалькулятора в служебных целях является стоимость решения задачи

$$C = c_{\text{р}}T + c_{\text{м}}T_{\text{м}},$$

где $c_{\text{р}}$ и $c_{\text{м}}$ — соответственно стоимость одного часа рабочего времени пользователя микрокалькулятора и машинного времени (времени включения) микрокалькулятора; T — общее время, затраченное пользователем на решение задачи; $T_{\text{м}}$ — время работы (точнее, время нахождения во включенном состоянии) микрокалькулятора.

Стоимость одного часа работы микрокалькулятора можно приближенно оценить по формуле $c_{\text{м}} = (C_0 + C_{\text{п}} + C_{\text{р}})/T_{\text{а}}$, где C_0 — начальная стоимость микрокалькулятора; $C_{\text{п}}$ и $C_{\text{р}}$ — соответственно стоимость питания и ремонта микрокалькулятора за период амортизации $T_{\text{а}}$ (например, за семь лет). По этой оценке стоимость одного часа работы даже для программируемых микрокалькуляторов не превышает 10 коп. и значительно ниже стоимости одного часа рабочего времени пользователя. Поэтому в тех случаях, когда во время выполнения программы вычисления микрокалькулятором в автоматическом режиме пользователь не занят решением других задач, можно принять $C \approx c_{\text{р}}T$ и, следовательно, экономический критерий эффективности применения микрокалькулятора совпадает с точностью до постоянного множителя с критерием минимальных затрат времени на решение задачи. В этих случаях общие затраты времени на решение задачи с использованием микрокалькулятора

$$T = T_{\text{с}} + T_{\text{к}} + T_{\text{в}} + T_{\text{д}}, \qquad (3.1)$$

где $T_{\text{с}}$ — затраты времени на составление и отладку программы для решения задачи; $T_{\text{к}}$ — затраты времени на ввод готовой программы нажатиями клавиш, включая затраты времени на исправление ошибок при вводе; $T_{\text{в}}$ — время выполнения микрокалькулятором операторов программы ($T_{\text{в}} < T_{\text{м}}$); $T_{\text{д}}$ — дополнительные затраты времени (в основном на ввод и регистрацию исходных данных).

Составляющие затрат времени в формуле (3.1) взаимосвязаны и задача программиста заключается в выборе оптимального варианта программы, обеспечивающего получение результата вычислений с требуемой точностью при минимуме общих затрат времени. Составляющая $T_{\text{с}}$ в формуле (3.1) минимальна при использовании готовой программы, взятой из библиотеки пользователя или стороннего источника, и в этом случае включает лишь затраты времени на поиск и изучение этой программы. Если программа составляется пользователем микрокалькулятора, то составляющая $T_{\text{с}} = T_{\text{со}}/k$, где $T_{\text{со}}$ — начальные затраты времени на составление и отладку программы, а k — число задач, решенных с помощью этой программы.

Если программа предназначена для однократного выполнения, то не имеет смысла тратить время на ее оптимизацию, и достаточно составить простейший вариант, обеспечивающий получение результата с требуемой точностью. В том случае, когда программа предназначена для многократного использования при решении достаточно сложных задач, большие начальные затраты времени $T_{\text{со}}$ на составление про-

граммы, предназначенной для хранения в библиотеке пользователя и обеспечивающей малые затраты времени $T_{\Sigma} = T_{к} + T_{в} + T_{д}$, окупаются тем быстрее, чем большее число раз выполнена программа.

Оптимальная программа обеспечивает минимальное значение T_{Σ}, зависящее от таких характеристик программы, как число операторов (длина программы), время $T_{в}$ однократного выполнения программы микрокалькулятором и уровень автоматизации решения задачи, определяющий дополнительные затраты времени $T_{д}$.

Затраты времени $T_{к}$ на ввод операторов программы зависят от количества нажимаемых клавиш, пропорционального длине программы, и навыков пользователя. При внимательном и четком нажатии клавиш уменьшается количество возможных ошибок (пропусков и неверных нажатий клавиш) и затраты времени на их исправления, которые могут оказаться преобладающими при необходимости повторного ввода программы.

Особенно опасны не замеченные при вводе программы ошибки, так как в этом случае результат вычислений окажется неверным, о чем может не подозревать пользователь. Для устранения таких ошибок правильность ввода программы автоматических вычислений проверяют по результатам выполнения контрольного примера. При вычислениях в обычном режиме длинные программы разбивают на части с регистрацией промежуточных результатов в вычислительном бланке. В этом случае возрастают дополнительные потери времени, но уменьшается вероятность ошибок (особенно при многократных вычислениях) и затраты времени на их исправление. Однако и этот прием не устраняет возможности возникновения незамеченных ошибок и поэтому при вычислениях на программируемых микрокалькуляторах даже те вычисления, которые могут быть выполнены в обычном режиме, но связаны с нажатием относительно большого числа клавиш, целесообразно выполнять в программируемом режиме с предварительной проверкой программы, занесенной в программную память.

Таким образом затраты времени $T_{к}$ на ввод операторов программы минимальны при минимальной длине программы в связи с уменьшением количества нажимаемых клавиш и вероятности ошибок.

Затраты времени $T_{в}$ на выполнение операторов программы микрокалькулятором пропорциональны количеству выполняемых операций, которое в разветвленных программах автоматических вычислений не соответствует количеству операторов (длине) программы. Минимизация длины программ, выполняемых в обычном режиме, и неразветвленных программ автоматических вычислений приводит в общем случае к уменьшению времени выполнения программы $T_{в}$ микрокалькулятором, но иногда замена медленно выполняемых операторов (например, X^{y}) несколькими быстро выполняемыми операторами может привести к уменьшению $T_{в}$ при увеличении длины программы. Сокращение длины программы автоматических вычислений при введении операторов условных и безусловных переходов обычно приводит к увеличению времени выполнения программы, так как в этом случае при том же количестве вычислительных операций возрастают потери времени на переходы. Повышение уровня автоматизации решения

задачи, уменьшая дополнительные затраты времени, приводит к удлинению программы и, следовательно, к увеличению составляющих $T_к$ и $T_в$. Следовательно, в общем случае оптимальность программы можно оценить только по суммарным затратам времени, а не только по длине программы или времени ее выполнения микрокалькулятором.

На основании анализа прикладных программ для массовых программируемых микрокалькуляторов можно принять, что для ввода одного оператора требуется в среднем 1,7 нажатий клавиш. Полагая, что на каждое нажатие клавиши в среднем затрачивается 0,5 с (при более быстром нажатии клавиш возрастают вероятность ошибок и влияние переходных процессов в пульте управления микрокалькулятора), получаем, что на ввод программы длиной L операторов затрачивается время $T_к \approx 0{,}85\ L$ с.

Операторы обращения к памяти вводят нажатиями двух клавиш и для ввода одного числа требуется от трех (при однозначном числе) до 16 нажатий клавиш. Примем среднее число нажатий клавиш для ввода в память одного числа восемь, что соответствует 4 с, и предположим, что для регистрации одного результата также требуется 4 с. Тогда суммарные затраты времени на решение задачи с помощью готовой программы (в секундах)

$$T_Σ \approx 0{,}85\,L + 4\,Д_и + 0{,}5N + k\,(T_{в1} + 4\,Д' + 4\,Д_р + 0{,}5\,N'),$$

где $Д_и$ и $Д'$— числа исходных данных, вводимых соответственно перед первым и последующими пусками программы; N и N'— числа клавиш, нажимаемых для первого и последующего пусков программы; $Д_р$ — число регистрируемых результатов; k — число пусков программы при решении задачи; $T_{в1}$— время однократного выполнения программы микрокалькулятором, которое оценивают по данным, аналогичным приведенным в работе [13].

В этой формуле не учтено время исправления случайных ошибок при вводе программы и исходных данных, но эта формула достаточно удобна для оценки оптимальности различных вариантов программы. Пути оптимизации программы в основном определяются соотношениями между обычно наибольшими составляющими $T_к \approx 0{,}85\ L$ и $T_в = kT_{в1}$. В случае, когда $T_в \gg T_к$ при большом числе k (например, при вычислении функциональных зависимостей в интервале изменения аргумента) или большом $T_{в1}$ (например, при решении задач методами последовательных приближений), следует в первую очередь стремиться к уменьшению времени однократного выполнения программы. Для этого сводят к минимуму число переходов в программе, что увеличивает ее длину, и минимизируют длины неразветвленных частей программы, выбирая, когда это возможно, наиболее быстро выполняемые операторы. Многократно выполняемые программы составляют так, чтобы при повторных пусках выполнялись лишь те вычисления, которые связаны с изменениями исходных данных, а общая часть расчета выполнялась только при первом пуске.

Если $T_к \gg T_в$ и $k = 1$, то общее время решения задачи

$$T_Σ \approx 0{,}85\,L + T_в + 4\,(Д_и + Д_р) + 0{,}5 \qquad (3.4)$$

и оптимальности программы добиваются, прежде всего, уменьшением ее длины, что сокращает и затраты времени на исправление ошибок при вводе программы. Минимизация программы необходима и в тех случаях, когда ее длина превышает допустимую емкость программной памяти, так как использование для решения задачи пакета программ приводит к резкому увеличению затрат времени, а некоторые задачи (например, решение уравнений методами последовательных приближений) можно решить с помощью только одной программы. Для минимизации длины программы применяются следующие основные способы:

1. Надлежащий выбор метода и алгоритма решения задачи. 2. Организация обращений к подпрограммам. 3. Организация вычислений в цикле. 4. Использование косвенной адресации. 5. Использование лексических и синтаксических особенностей входного языка.

Оптимальный выбор метода и алгоритма решения задачи с помощью программы минимальной длины зависит от условий конкретной задачи и для ряда типовых задач рассмотрен далее. Если в программе содержится n одинаковых фрагментов M по m операторов, то один такой фрагмент можно вынести в подпрограмму, оканчивающуюся оператором В/О, а на место каждого из исходных фрагментов записать двухшаговый оператор обращения к подпрограмме. Так как в этом случае mn операторов исходных фрагментов заменяются $m + 2n + 1$ операторами, то изменение длины программы при введении подпрограммы

$$\Delta L = mn - (m + 2n + 1) \tag{3.5}$$

и длина программы уменьшается при условии $\Delta L > 0$, являющимся критерием целесообразности организации подпрограмм. Время выполнения программы при введении обращений к подпрограмме возрастает на величину

$$\Delta T_{\text{в}} = n\,(t_{\text{п. п}} + t_{\text{в/о}}),$$

где $t_{\text{п. п}}$ и $t_{\text{в/о}}$ — время выполнения перехода к подпрограмме и возврата из нее соответственно.

Если в программе содержится последовательность m одинаковых фрагментов M из n операторов, то ее можно заменить одним фрагментом M, охваченным циклом с выполнением n итераций. В этом случае изменение длины программы определяется формулой

$$\Delta L = mn - (m + p) = m\,(n - 1) - p, \tag{3.6}$$

где p — число дополнительных операторов (шагов), вводимых для организации цикла, и целесообразность организации цикла определяется выполнением условия $\Delta L > 0$.

В программах на входном языке ЯМК21 требуется не менее $p = 6$ дополнительных операторов и один регистр памяти для организации цикла

$$\ldots\ M\ \text{FN}\ 1\ -\ \text{PN}\ x = 0\ A\ \ldots,$$

где N — номер регистра памяти, на котором организован счетчик итераций и в который перед каждым пуском программы заносят

число n итераций; A — указатель перехода к первому оператору фрагмента M. После выполнения n итераций содержимое регистра N становится равным нулю и условный оператор передает управление следующей части программы.

Уровень автоматизации решения задачи при многократном использовании программы повышают, организовав автоматическое восстановление содержимого счетчика итераций на регистре N_2 перезаписью содержимого регистра N_1, в который перед первым пуском программы заносят число n итераций:

$$\ldots \text{FN}_1\ \text{PN}_2\ M\ \text{FN}_2\ 1\ —\ \text{PN}_2\ x = 0\ A\ \ldots$$

В программах на входном языке ЯМК34 для организации циклов

$$\ldots M\ \text{LN}\ A\ \ldots \text{ или } \text{ИПN}_1\ \text{ИПN}_2\ M\ \text{LN}_2\ A\ \ldots$$

число итераций n заносят соответственно непосредственно в регистр N или, для перезаписи при повторных пусках программы, в регистр N_2. Время выполнения программы при организации цикла увеличивается на величину

$$\Delta T_в = nt' + t'',$$

где t'— время выполнения дополнительных операторов в цикле (без учета времени выполнения фрагмента M); t''— время выполнения операторов перезаписи содержимого счетчика итераций (если их нет, то $t'' = 0$).

При вычислениях в цикле с большим числом n итераций даже незначительное сокращение времени выполнения фрагмента M приводит к значительному уменьшению общего времени выполнения программы. Поэтому в таких случаях (особенно при решении задач методами последовательных приближений, когда условие выхода из цикла определяется значениями результата вычислений в цикле) следует вынести из фрагмента M все операторы, которые можно выполнить вне цикла. Длину программы часто удается сократить при использовании операторов косвенной адресации, но при их вводе время выполнения программы также возрастает.

Число операторов программы часто удается сократить, используя как «литературные», так и «жаргонные» особенности входного языка. Например, умножение содержимого регистра N на 10^m в «литературном» варианте реализуют фрагментом программы ИПN 1 ВП m ×, который можно заменить более коротким «жаргонным» фрагментом ИПN ВП m (или FN ВП m на входном языке ЯМК21).

Вычисление модуля $|x|$ текущей переменной, отображаемое «литературным» фрагментом $x < 0\ A\ /—/$ (где указатель перехода A передает управление части программы, следующей за оператором /—/), можно заменить (с потерей точности при округлении) «жаргонным» вариантом $x^2\ \sqrt{}$.

Следует максимально использовать лексические и синтаксические особенности входного языка для сокращения длины программы. Например, в программах на входном языке ЯМК21 фрагмент вида sin ↑ FN × целесообразно заменять более коротким фрагментом e^{ix}

FN $\times$, а при вычислении выражений вида $2a \pm b$ вместо «литературной» реализации FN_1 2 $\times$ $\uparrow$ FN_2 $\pm$ при $a = PN_1$, $b = PN_2$ использовать

$$FN_1 \uparrow FN_2 \pm +.$$

Аналогично при вычислениях по выражениям вида $a / (a \pm b)$ вместо FN_1 $\uparrow$ FN_2 $\pm$ $\uparrow$ FN_1 XY $\pm$ следует использовать

$$FN_1 \mid \uparrow \mid FN_2 \mid \pm \mid \div,$$

а при вычислениях по выражениям вида $(a - b)/(a + b)$ вместо FN_1 $\uparrow$ FN_2 + PN_3 FN_1 $\uparrow$ FN_2 — $\uparrow$ FN_3 $\div$ записывать в программу

$$FN_1 \uparrow FN_2 \div 1 - 2 + \div.$$

Подобные замены целесообразно использовать и при составлении программ на других входных языках, вычисляя, например, выражение вида $2a \pm b \pm c \pm d$ при $a = PN_1$, $b = PN_2$, $c = PN_3$, $d = PN_4$ на входном языке ЯМК34 вместо фрагмента $ИПН_1$ 2 $\times$ $ИПН_2$ $\pm$ $ИПН_3$ $\pm$ $ИПН_4$ $\pm$ при помощи более короткого фрагмента

$$ИПН_1\ ИПН_2\ ИПН_3\ ИПН_4 \pm \pm \pm +,$$

так как содержимое регистра N_1 после выполнения первых трех операций сохраняется в регистре Y операционного стека.

Количество записанных в программе операторов набора чисел часто удается сократить с помощью функциональных операторов, например, заменяя фрагменты 0 , 3 3 3 ... ; 1 , 4 1 4 2 ... ; 2 , 7 1 8 ... ; ИПН 0 , 0 6 2 5 $\times$ соответственно более короткими фрагментами 1 $\uparrow$ 3 $\div$; 2 $\sqrt{}$; 1 e^x, и ИПН 4 x^2 $\div$. Подобные приемы целесообразно использовать и при вводе исходных данных для сокращения количества нажимаемых клавиш.

Иногда пакет программ приходится использовать только потому, что емкости числовой памяти недостаточно для хранения исходных данных и результатов вычислений и каждая программа пакета выполняется при меньшем числе исходных данных. В таких случаях необходимость в пакете программ часто удается устранить, используя следующие основные приемы:

1. Надлежащий выбор алгоритма вычислений с нормированием, в частности, исходных данных.

2. Последовательный ввод исходных данных и вывод результатов с остановками программы.

3. Уменьшение числа регистров памяти, одновременно используемых в качестве буферных.

4. Занесение части исходных данных (в первую очередь, постоянных) в программу в виде операторов набора чисел.

5. Хранение части исходных данных и результатов вычислений в регистрах операционного стека.

При нормировании расчетных выражений (как было ранее показано на примере дифференциального уравнения) часто удается значительно сократить количество исходных данных. В частности, при нормировании алгебраического многочлена делением его коэффициентов

на один из них корни многочлена не изменяются, а значения нормированного и ненормированного многочленов отличаются постоянным множителем. Аналогично, при нормировании числителя или знаменателя дробно-рациональной функции (отношения многочленов) ее значение не изменяется, а при нормировании числителя и знаменателя их корни не изменяются, а значения нормированной и ненормированной функций отличаются постоянным множителем, хотя в этом случае число исходных данных уменьшается на два.

Требуемое число регистров памяти можно уменьшить, вводя в программу операторы С/П после выполнения операций над введенными исходными данными для замены их новыми. Аналогично можно вводить операторы С/П для регистрации результатов решения задачи по мере их вычисления.

Уменьшения числа регистров памяти, одновременно используемых в качестве буферных для хранения промежуточных результатов, часто удается достичь при использовании синтаксических особенностей входных языков. Например, для занесения нового значения вычисляемой функции в тот же регистр N, где хранилось прежнее значение функции, перед сравнением этих значений (обычным для методов последовательных приближений) достаточно использовать фрагмент |ИПN| XY |ПN| (или |↑| FN |XY| PN), после выполнения которого текущая переменная находится в регистрах X и N, а прежнее значение функции — в регистре Y, и может быть выполнена операция сравнения (обычно основанная на вычитании).

Число требуемых регистров памяти можно уменьшить, занося в соответствующие места программы операторы набора исходных данных. В этом случае в программу следует заносить немногозначные числа. При занесении в программу на входном языке ЯМК21 однозначных чисел ее длина даже уменьшается, так как при вызове такого числа из памяти операторы |↑|FN| занимают два шага программы.

Для хранения изменяющихся исходных данных и результатов вычислений целесообразно использовать регистры операционного стека, прежде всего, регистры X и Y. Это особенно удобно, когда исходные данные вводят парами (например, составляющие комплексных чисел), и сокращает затраты времени на ввод исходных данных, так как для занесения первого числа в регистр Y достаточно ввести оператор ↑, а второе число перед пуском программы сохраняется в регистре X. Аналогично при хранении пары результатов в регистрах X и Y для вызова содержимого регистра Y достаточно нажать одну клавишу XY, причем в этом случае прежнее содержимое регистра X сохранится в регистре Y.

В микрокалькуляторах с входным языком ЯМК34 для хранения исходных данных и результатов вычислений можно использовать все регистры операционного стека, включая регистр $X1$. При этом одно исходное число, находящееся предварительно в регистре T, может постоянно храниться в операционном стеке, если он не переполняется в процессе вычислений.

Составляющая $T_д$ дополнительных затрат времени обычно значительно меньше времени ввода и выполнения программы, но в некото-

рых случаях (например, при использовании вычислительного бланка) она оказывается значительной. Ее уменьшению способствуют четкие регистрация и считывание данных и продуманная организация вычислений, включая форму вычислительных бланков и размещение в регистрах памяти и операционного стека исходных данных и результатов вычислений.

3. МЕТОДИКА СОСТАВЛЕНИЯ ПРОГРАММ

Практическая необходимость в использовании вычислительных средств при решении инженерных задач возникает, когда известна математическая модель физического объекта или следующие из нее расчетные соотношения, связывающие исходные данные с искомыми результатами расчета.

Составление оптимальных программ решения громоздких задач на программируемых микрокалькуляторах с ограниченной емкостью запоминающих устройств не уступает по сложности программированию универсальных ЭВМ, при котором программисту обычно не приходится распределять в памяти исходные данные и результаты вычислений. Для квалифицированного составления программ различного назначения пользователь микрокалькулятора должен хорошо знать вычислительную математику, в совершенстве знать особенности входного языка микрокалькулятора используемого типа и обладать творческими способностями, в основном и определяющими искусство программирования. Поэтому любые методические правила программирования не могут охватить многообразия эвристических путей составления и оптимизации программ и ограничены лишь общими рекомендациями и описаниями типовых приемов оптимизации программы.

При решении простых задач программы составляют непосредственно по расчетным формулам, но для сложных задач часто приходится предварительно выбирать метод решения и составлять описание или схему соответствующего алгоритма. Разбивая такой алгоритм при необходимости на крупные блоки, оценивают возможности их реализации с помощью одной или нескольких программ.

Составление программы по описанию или схеме алгоритма или по соответствующему расчетному соотношению начинают с распределения регистров памяти и операционного стека для размещения исходных данных и результатов вычислений, учитывая по возможности и буферные регистры для размещения промежуточных результатов вычислений.

Исходные данные (в первую очередь, изменяемые перед каждым пуском программы) целесообразно размещать в регистрах X и Y операционного стека. Если эти переменные должны храниться также в памяти для вызова в процессе вычислений, то программу целесообразно начать операторами ПN или ПN_1 XY ПN_2— в этом случае после пуска программы содержимое регистра X будет заслано соответственно в регистр N или N_1 памяти, а содержимое регистра Y — в регистр N_2. При вычислениях на микрокалькуляторе с входным

языком ЯМК34 исходные данные (и результаты расчета) можно размещать и в остальных регистрах (*Z*, *T* и *X1*) операционного стека.

В регистрах числовой памяти исходные данные следует размещать так, чтобы клавиши набора номеров этих регистров были расположены близко к клавишам засылки или вызова, что обеспечит минимальное время ввода и вывода данных. При большом числе исходных данных целесообразно предусмотреть такое их размещение в памяти, которое легко запоминается пользователем без внимательного чтения инструкции к выполнению программы. Так, при упорядоченном множестве исходных данных (например, коэффициентов многочлена) a_0, a_1, a_2, a_3, ... их следует занести соответственно в регистры памяти с номерами *0*, *1*, *2*, *3*, ... или, при входном языке ЯМК21, в регистры *2*, *C1*, *C2*, *C3*, ... Элементы квадратных матриц удобно размещать в регистрах памяти, клавиши обращения к которым расположены соответственно размещению соответствующих элементов в матрице (например, элементы a_{11}, a_{12}, a_{21} и a_{22} квадратной матрицы следует размещать в регистрах с номерами *7*, *8*, *4* и *5* или *5*, *6*, *2* и *3* соответственно). Продуманное размещение исходных данных и результатов вычислений в памяти обеспечивает сокращение дополнительных затрат времени на решение задачи.

Эти затраты уменьшаются и при повышении уровня автоматизации решения задачи. В частности, в программах, предназначенных для многократного выполнения в процессе вычислений, оператор С/П в конце программы дополняют оператором безусловного перехода по адресу 00. В этом случае при повторных пусках программы достаточно нажать только клавишу С/П, так как выполняемый после этого оператор безусловного перехода передаст управление первому оператору программы, с которого начинается ее выполнение. В пустую ячейку программной памяти (например, при вводе программы после включения микрокалькулятора) адрес 00 на входном языке ЯМК34 после команды БП можно не вводить, так как содержимое пустой ячейки будет считано как адрес 00.

Если длина программы меньше предельной, то возможна автоматизация распределения набираемых исходных данных в памяти. Для этого достаточно начать программу фрагментом вида $ПN_1$ С/П $ПN_2$ С/П $ПN_3$ С/П ... и после набора очередного исходного данного нажать клавишу С/П (первый раз клавиши В/О и С/П). В программах на входном языке ЯМК34 такой фрагмент можно заменить операторами КПО С/П с занесением в регистр *0* числа, на единицу большего наибольшего номера регистров памяти, в которых размещаются исходные данные. Тогда после набора в регистре очередного исходного числа достаточно нажать клавиши В/О и С/П, после чего содержимое регистра *X* будет заслано в регистр, номер которого на единицу меньше предыдущего содержимого регистра 0. После размещения исходных данных в обоих случаях для выполнения основной части программы достаточно нажать только клавишу С/П.

Аналогично можно автоматизировать и вызов результатов вычислений из памяти, дополнив программу фрагментом вида $ИПN_1$ С/П $ИПN_2$ С/П ... $ИПN_k$ С/П. В этом случае после выполнения програм-

мы высвечивается содержимое регистра N_1, а для вызова остальных результатов, хранящихся в регистрах $N_2 \ldots N_k$, достаточно последовательно нажимать только клавишу С/П. Возможны и различные комбинации автоматических и ручных операций засылки и вызова данных.

В процессе программирования удобно использовать рассмотренную ранее компактную запись. При составлении разветвленных программ адреса переходов к еще несоставленной части программы неизвестны. В таких случаях следует пропустить шаг программы после оператора перехода (подчеркнув этот шаг или обведя его прямоугольником) и заполнить его адресом перехода после составления той части программы, которой передается управление. В черновике программы для наглядности удобно отображать переходы линиями со стрелками, например (для программы вычисления факториала)

$$\text{ПО}\ \ x=0\ \ 08\ \ 1\ \ \text{ИПО}\ \ \times\ \ \text{LO}\ \ 04\ \ \text{С/П} \qquad (3.7)$$

При составлении и оптимизации программ часто приходится исследовать изменение содержимого используемых регистров операционного стека (для этого можно выписывать «в столбик» содержимое операционных регистров над каждым оператором в черновике программы) и числовой памяти.

Составляя даже простые программы, следует выбирать эквивалентные представления расчетных соотношений, наиболее удобные для реализации на используемом входном языке и обеспечивающие оптимальность программ. В частности, следует выбирать формулы, содержащие минимальное число операций, что обычно обеспечивает минимальные погрешности вычислений и минимальную длину программы.

В качестве примера рассмотрим вычисление функции

$$f(x) = 2\ln(x-1) - \ln(x+1) = \ln\left((x-1)^2/(x+1)\right).$$

При выборе регистра *2* для хранения аргумента x и регистра *3* в качестве буферного для хранения промежуточных результатов по первой из приведенных формул на входном языке ЯМК21 несложно составить программу

$$\text{P2}\ \ 1\ +\ \ln\ \text{P3}\ \ \text{F2}\ \ 1\ -\ \ln\ \ 2\ \times\ \uparrow\ \text{F3}\ -\ \text{С/П}$$

Составление программы по второй из приведенных формул начнем с реализации вычисления промежуточного результата $y = x - 1$. Так как $x + 1 = y + 2$ и содержимое регистра X при наборе операнда засылается в регистр Y, составляем фрагмент Р2 1 — 2 +, после выполнения которого $PX = x + 1$ и $PY = x - 1$. Дополнив этот фрагмент операторами деления, умножения (для возведения y в квадрат) и логарифмирования, получим программу

$$\text{P2}\ \ 1\ -\ 2\ +\ \div\ \times\ \ln\ \text{С/П}.$$

Таким образом, выбор формулы с меньшим числом операций и рациональное использование правил работы операционного стека (соответствующих синтаксическим правилам входного языка ЯМК21) обеспечили сокращение первоначально составленной программы на 40 %.

Аналогично на входном языке ЯМК34 первая формула реализуется программой

$$\text{П2}\ \ 1\ -\ \ln\ \ 2\ \times\ \text{ИП2}\ \ 1\ +\ \ln\ -\ \text{С/П}$$

и вторая — более короткой программой

$$\text{П2}\ \ 1\ -\ x^2\ \ \text{ИП2}\ \ 1\ +\ \div\ \ln\ \text{С/П}.$$

В процессе составления программ часто приходится изменять как алгоритм, так и метод решения задачи, прибегая к разветвлениям вычислений для обеспечения требуемой точности или заданного диапазона исходных данных и результатов. В качестве примера рассмотрим вычисление факториала целых чисел.

Тщательный анализ выбранного ранее алгоритма (см. рис. 3) и особенностей входного языка ЯМК21 позволяет после оптимизации программы 1/21 составить более короткий вариант

$$x=0\ 0\ 1\ \uparrow\ \text{P2 F1 1}\ -\ \uparrow\ \text{F2}\ \times\ x=0\ \text{F}\uparrow\ \text{F2 С/П}$$

или, при использовании кольцевого стека памяти,

$$\rightarrow\ 1\ \leftarrow\ 1\ -\ \rightarrow\ \text{P2}\ \times\ x=0\ \text{P}\uparrow\ \text{F2 С/П}$$

с вычислением значения 10! всего за 15 с.

Оптимизируя программу 2/34, получим еще более короткий вариант (3.7), который для факториала целых положительных чисел можно дополнительно сократить, записав

$$\text{ПО 1 ИПО}\ \times\ \text{LO О2 С/П}$$

с вычислением 10! всего за 11 с.

Все приведенные программы непригодны для вычисления факториала чисел $a > 69$, так как в этом случае значение $a!$ попадает в область машинной бесконечности и возникает переполнение. Для его устранения необходимо представить факториал в форме, допускающей вычисление порядка $n > 99$.

Простейший способ решения этой задачи заключается в нормировании текущего произведения в цикле делением на число 10^m, если это произведение превышает 10^m с одновременным суммированием порядков m при каждом нормировании. Сумма S порядков m ограничена лишь граничным значением A_{max} представления чисел в микрокалькуляторе и по результатам вычисления произведения $M \cdot 10^n$ и суммы S несложно определить $a! = M \cdot 10^{n+S} \leqslant M \cdot 10^{n+A_{max}}$. Этот способ при нормировочном коэффициенте 10^{99} использован в следующей программе.

Программа 3/34. Вычисление факториала целых положительных чисел

П0	1	ИП0	×	П1	ИП4	—	$x \geqslant 0$ 18 ИП7
9	0	+	П7	ИП1	ИП4	÷	П1 ИП1 L0
02	С/П						

Инструкция: $(1 \cdot 10^{99} = P4)$ $0 = P7$, $a = PX$ В/О С/П $PX = a!$, если $a < 70$ и $PX = M \cdot 10^n$, $P7 = S$, $a! = M \cdot 10^{n+S}$, если $a > 69$.

Контрольный пример: время вычисления $10! = 3628800$ около 35 с, время вычисления $70! = 1{,}197857 \cdot 10^{100}$ около 3,5 мин, время вычисления $200! = 7{,}8865662 \cdot 10^{374}$ около 10 мин.

При вычислениях по этому алгоритму в каждом цикле выполняется вычитание, точное при $a \leqslant 10^{r-1}$, где r — разрядность микрокалькулятора, и умножение, предельная погрешность результата которого не превышает единицы последнего разряда мантиссы. При $a \leqslant 100$ выполняется не более 100 циклов и по мажоритарной (и, следовательно, завышенной) оценке операционной погрешности мантисса $a!$ содержит не менее шести верных цифр при $r = 8$.

Теоретически рассмотренным методом можно вычислять факториалы точно заданных чисел $a \leqslant 10^{r-1}$ и приближенно заданных $a \leqslant 10^{97}$. Однако на вычисление факториалов больших чисел затрачивается значительное время и для этой цели более удобными становятся асимптотические приближения, например, формула Стирлинга

$$a! \approx \sqrt{2\pi a}\,(a/e)^a\,(1 + 1/12a + \ldots) \qquad (3.8)$$

с методической погрешностью, уменьшающейся при увеличении числа a и числа учитываемых членов ряда в скобках.

При вычислениях по этой формуле на микрокалькуляторе с разрядностью $r = 8$ при $a > 69$ также возникает переполнение, а рассмотренный ранее способ нормирования текущего произведения непригоден. Поэтому приходится прибегать к другим способам представления факториала, например, вычислению его натурального логарифма

$$\ln a! \approx a(\ln a - 1) + \ln \sqrt{2\pi a} + \ln(1 + 1/12a + \ldots)$$

или десятичного логарифма

$$\lg a! \approx a(\lg a - \lg e) + \lg \sqrt{2\pi a} + \lg(1 + 1/12a + \ldots).$$

Для определения $a!$ по вычисленному значению $\ln a!$ последнее следует разбить на слагаемые, не превышающие 230 при $r = 8$, вычислить антилогарифм e^x для каждой составляющей, умножить мантиссы и сложить порядки результатов.

Для вычисления $a!$ по вычисленному значению $\lg a!$ последнее следует разбить на целую и дробную части и вычислить антилогарифм 10^x только дробной части, так как целая часть определяет основную составляющую порядка. Практически для этого следует вычислить антилогарифм дробной части и сложить порядок результата с целой частью $\lg a!$.

Наибольшие операционные погрешности возникают в рассматриваемом случае при выполнении операторов ln и e^x или lg и 10^x. Кроме того, при увеличении a увеличивается целая часть логарифма и уменьшается число значащих цифр в дробной части, что приводит к увеличению относительной погрешности. Поэтому при увеличении a методическая погрешность формулы (3.8) уменьшается, но одновременно возрастает операционная составляющая погрешности результата. В частности, при ограничении ряда в скобках формулы (3.8) первым членом вычисленное значение $a!$ содержит не более трех верных цифр. Ограничившись для повышения точности двумя членами этого ряда, составим следующую программу, пригодную для вычисления факториала нецелых (с достаточно большой целой частью) чисел.

Программа 4/34. Вычисление логарифма факториала $\lg a!$ положительных чисел

ПО ИПО 1 e^x ÷ lg × ИПО 2 × π × $\sqrt{}$ lg + ИПО 1 2 × 1/x 1 + lg+С/П

Инструкция: $a = PX$ В/О С/П $PX = \lg a!$ (время счета около 12 с).

Контрольный пример: $\lg 10! = 6{,}5597487$ ($10! = 36286807$); $\lg 70! = 100{,}0784$ ($70! = 1{,}1978433 \cdot 10^{100}$); $\lg 100! = 157{,}96999$ ($100! = 9{,}5523279 \cdot 10^{157}$); $\lg 200! = 374{,}89689$ ($200! = 7{,}8866038 \cdot 10^{374}$).

Из сравнения результатов выполнения этого и предыдущего контрольных примеров следует, что программа 4/34 для выбранных значений a обеспечивает вычисление $a!$ с четырьмя верными цифрами. При желании эту программу можно дополнить фрагментом, разделяющим дробную и целую части результата и вычисляющим значение 10^x от дробной части, например П9 КИП9→ ИП9 — 10^x с хранением порядка (целой части $\lg a!$) в регистре *9*. Программу 4/34, дополненную таким фрагментом, можно объединить с программой 3/34 для точного вычисления $a!$ при $a < 70$ и приближенного, но быстрого вычисления $a!$ при $a > 69$. В этом случае выбор способа вычислений удобно автоматизировать с помощью условного оператора, выбирающего нужную ветвь в зависимости от значения a, например,

ПО 6 9 — $x \geqslant 0$ *А* ПР1 С/П ПР2 С/П,

где при $a > 69$ управление передается ветви ПР2 с промежуточным вычислением $\lg a!$, а при $a < 69$ выполняется фрагмент ПР1, соответствующий программе 3/34.

Составляемые программы приходится разветвлять и при решении многих других задач, причем в случае совпадения фрагментов отдельных ветвей для сокращения длины программы их целесообразно вынести в подпрограммы или организовать объединение в общую ветвь.

В качестве примера составим на входном языке ЯМК21 программу вычисления функции

$$f(x) = \begin{cases} 0 \text{ при } x < 0; \\ ax^2/(b+x) \text{ при } 0 \leqslant x < 10, \\ ax^2/(b+x^2-90) \text{ при } x \geqslant 10, \end{cases}$$

Распределив исходные данные $a = P2$, $b = P3$, $x = P8$, реализуем вычисление функции в интервале $x < 0$ фрагментом F8 $x < 0$ ... Сх С/П, где многоточием обозначен неизвестный пока указатель перехода при $x \geqslant 0$. Для вычисления функции в интервале $0 \leqslant x < 10$ учтем, что после проверки $x < 0$ достаточно проверить выполнение только второго неравенства, например:

F8 1 0 — $x < 0$ ... F8 ↑ F3 + P4 F8
x^2 ↑ F2 × ↑ F4 ÷ С/П

Для вычисления функции в интервале $x \geqslant 10$ составим фрагмент

F8 x^2 ↑ F3 + 9 0 — P4 F8 x^2 ↑
F2 × ↑ F4 ÷ С/П

и, объединив все фрагменты, получим программу длиной 43 шага:

F8 $x < 0$ ↑ Сх С/П F8 1 0 — $x < 0$ F4 F8
↑ F3 + P4 F8 x^2 ↑ F2 × ↑ F4 ÷
С/П F8 x^2 ↑ F3 + 9 0 — P4 F8 x^2
↑ F2 × ↑ F4 ÷ С/П

Попытаемся уменьшить длину программы, объединив одинаковые части ветвей программы при положительных значениях аргумента. Учитывая сохранение значения x в регистре Y после выполнения условного оператора, преобразуем фрагмент вычисления функции при $x < 10$ к виду

F8 1 0 — $x < 0$ ... F3 + $1/x$ ↑ F8 x^2
× ↑ F2 × С/П

и представим фрагмент вычисления функции при $x \geqslant 10$ как

F8 x^2 9 0 — ↑ F3 + $1/x$ ↑ F8 x^2
× ↑ F2 × С/П

В этих фрагментах совпадает 11 операторов, которые можно вынести в подпрограмму, но в данном случае проще объединить их в общую ветвь с помощью операторов безусловного перехода. Можно обойтись и без этих операторов, учитывая, что после первой проверки условия $x > 0$ значение переменной сохраняется в регистре X, а для объединения одинаковых ветвей условие $x < 10$ удобнее заменить условием $x \geqslant 10$. Заменив начальный оператор на $P8$, что обеспечит автоматическую засылку в память набранного значения аргумента, получим состоящую всего из 27 шагов программу

P8 $x < 0$ ↑ Сх С/П 1 0 — $x \geqslant 0$ F× F8 x^2
9 0 — ↑ F3 + $1/x$ ↑ F8 x^2 × ↑
F2 × С/П

с инструкцией: $a = P2$, $b = P3$, $x = PX$ В/О С/П $PX = f(x)$.

Еще короче программа вычисления заданной функции на входном языке ЯМК34:

П8 $x < 0$ 05 Сх С/П x^2 ИП2 × ИП8 ИП8
1 0 — $x \geqslant 0$ 21 → x^2 9 0 —
↑ → ИП3 + ÷ С/П

При $a = 1$, $b = 3$ и $x = -2; 5; 12$ по этим программам получим соответственно $f(x) = 0$; 3,125; 2,5263157.

Для удобства пользования составленными программами их следует дополнить операторами БП РО и БП 00 соответственно — в этом случае клавишу В/О можно нажимать лишь для первого пуска программ.

Решение многих практических задач заключается в определении численных значений различных переменных, связанных одним соотношением, по известным значениям остальных переменных. При составлении программ решения подобных задач необходимо организовать проверку содержимого регистров, выделенных для хранения переменных с предварительным занесением нуля на место неизвестной переменной (если заданные значения переменных также могут быть нулевыми, то в этом случае на место неизвестной переменной следует занести число, лежащее вне интервалов задаваемых переменных, или использовать другой способ определения неизвестной переменной).

Методику программирования подобных задач рассмотрим на примере вычисления одного из параметров v, h, R и r, связанных формулой $v = \pi h\,(R^2 + r^2 + Rr)\,/\,3$ для объема усеченного конуса.

Составим формулы для вычисления $h = 3v/\pi\,(R^2 + r^2 + Rr)$, $R = \sqrt{3\,(v/\pi h - r^2/4)} - r/2$, $r = \sqrt{3\,(v/\pi h - R^2/4)} - R/2$ и примем $v = P2$, $h = P3$,

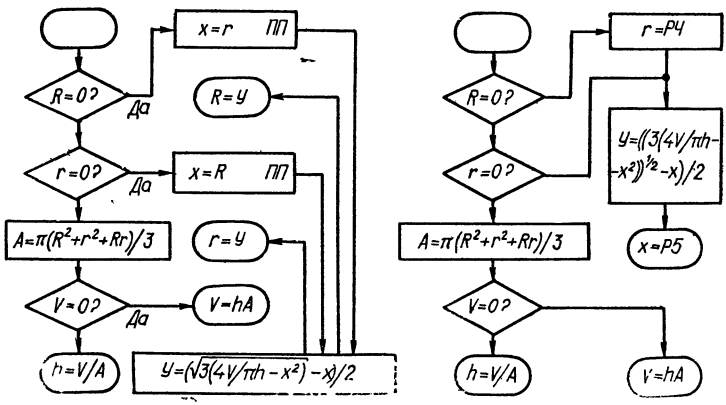


Рис. 21

$R = P4$, $r = P5$. Тогда v и h можно вычислить на входном языке ЯМК21 по программам Ф1 F3 × P2 С/П и Ф1 F2 XY ÷ P3 С/П с общим фрагментом Ф1:

F4 ↑ F5 + × ↑ F5 x^2 + ↑ π ×
3 ÷ ↑

Аналогично R и r можно вычислить по программам F5 Ф2 P4 С/П и F4 Ф2 P5 С/П с общим фрагментом Ф2:

2 ÷ P6 F2 ↑ F3 ÷ ↑ π ÷ ↑ F6
x^2 — 3 × $\sqrt{}$ ↑ F6 —

При непосредственном объединении рассмотренных программ с организацией автоматического выбора нужной ветви длина объединенной программы оказывается больше допустимой. Однако с учетом одинаковых фрагментов длину объединенной программы можно сократить, используя показанную на рис. 21,*а*

схему с вычислением в подпрограмме значений $x = (3\,(v/\pi h - (b/2)^2))^{1/2}$, где $b = R$ при $r = 0$ или $b = r$ при $R = 0$. Однако и в этом случае один оператор оказывается избыточным и поэтому изменим схему решения задачи так, чтобы известное значение R или r заносилось в регистр *4*, а неизвестное после его вычисления — в регистр *5* (рис. 21,*б*).В этом случае длина программы окажется меньше предельно допустимой на пять шагов. Этот запас целесообразно использовать для организации повторных пусков программы нажатием только клавиши С/П:

F4	$x \neq 0$	P	/—/	↑	F5	$x \neq 0$	F	/—/	+	×	↑	F5	x^2
+	↑	π		×	3	÷	↑		F3	$x \neq 0$	F4	×	P2
БП	P+	F2		XY	—	P3	БП		P+	F5	P4	F2	↑
F3	÷	4		×	↑	π	÷		↑	F4	x^2	—	3
×	$\sqrt{}$	↑		F4	—	2	÷		P5	С/П	БП	P0	

Пользование этой программой определяется следующей инструкцией: $v = P2$, $h = P3$, $R = P4$, $r = P5$ — вместо неизвестной величины занести нуль; (В/О) С/П $PX = x$, $P2 = v$, $P3 = h$, $P4 = R$ или r (заданное значение), $P5 = R$ или r (вычисленное значение); если заданы все исходные данные, то вычисляется h.

При составлении и особенно оптимизации программ приходится оценивать время выполнения микрокалькулятором отдельных операторов или фрагментов программы. Для этого следует засечь время t_n выполнения достаточно большого числа (n раз) этого фрагмента или оператора и определить время $t_1 = t_n/n$ однократного выполнения.

Оценку времени t_1 выполнения оператора или фрагмента М удобно получать с помощью многократных повторений М в цикле. Однако в этом случае необходимо также учесть время выполнения вспомогательных операций. Для этого следует ввести в программную память программу F2 1 — P2 $x = 0$ PO С/П на входном языке ЯМК21 или программу L2 OO С/П на входном языке ЯМК34, ввести в регистр *2* достаточно большое число n итераций, нажать клавиши В/О и С/П и засечь время $t_ц$ выполнения программы, соответствующее времени выполнения операций для организации n циклов.

Для проверяемого оператора или фрагмента М следует ввести программу М F2 1 — P2 $x = 0$ PO С/П или М L2 OO С/П соответственно, занести в регистр *2* то же число итераций n и, пустив программу нажатием клавиш В/О и С/П, засечь время $t_в$ ее выполнения. Тогда среднее время выполнения оператора или фрагмента М составит $t_1 = (t_в - t_ц)/n$.

Подобным образом проверяется и время выполнения вспомогательных операций, в частности, время выполнения переходов к подпрограммам и условных переходов. Пусть, например, требуется определить время переходов при выполнении подпрограмм (время выполнения операторов ПП *ab* и В/О). Выполнив программу ПП 1 F2 1 — P2 $x = 0$ PO С/П В/О на входном языке ЯМК21 (или программу ПП 05 L2 OO С/П В/О на входном языке ЯМК34) при $n = 100 = P2$, получим, например, $t_в = 111$ с. Если при $n = 100$ время $t_ц = 79$ с, то среднее время обращения к подпрограмме составит $t_1 = (111-79)/100 = 0{,}32$ с.

4. ПЕРЕВОД ПРОГРАММ

Затраты времени на составление сложной программы могут быть значительно сокращены при переводе на входной язык используемого микрокалькулятора программы решения задачи, составленной на другом входном языке. В частности, создание программного обеспечения микрокалькуляторов с входным языком ЯМК34 может быть упрощено при переводах программ, составленных на разработанном ранее входном языке ЯМК21 [11, 19]. В тех случаях, когда алгоритм решения задачи неизвестен, перевод программы с другого входного языка может оказаться простейшим способом ее составления.

Переводы программ аналогичны переводам текстов с одного естественного языка на другой — дословному (подстрочному) переводу, обычному переводу по предложениям и вольному переводу или пересказу содержания текста.

Дословный (пооператорный) перевод, когда каждый оператор программы-оригинала отображается оператором или фрагментом на входном языке программы-перевода, реализуется часто наиболее просто и всегда возможен, если длина программы-перевода не ограничена емкостью программной памяти, а число регистров памяти, используемых в программе-оригинале, не больше числа регистров памяти микрокалькулятора, на входной язык которого переводится программа.

Для выполнения дословного перевода необходимо установить формальные правила соответствия регистров N памяти в языке оригинала регистрам M памяти в языке перевода и составить правила соответствия (словарь) операторов оригинала операторам или фрагментам программы на языке перевода. Кроме того, должны быть установлены правила соответствия адресации переходов в оригинале и переводе программы.

Рассмотрим дословный перевод программ с входного языка ЯМК21 на язык ЯМК34, установив следующие правила соответствия регистров числовой памяти:

1. При отсутствии в программе-оригинале сбращений к кольцевому стеку памяти регистрам ЗУПВ с номерами $N = 2, \ldots, 8$ микрокалькулятора с входным языком ЯМК21 соответствуют регистры памяти с номерами $M = 2, \ldots, 8$ в программе-переводе.

2. При обращении в программе-оригинале к кольцевому стеку памяти регистрам ЗУПВ с номерами $N = 2, \ldots, 8$ соответствуют регистры $M = N + 4 = 6, 7, 8, 9, A, B, C$ в программе-переводе, а регистрам кольцевого стека $CN = C1, \ldots, C6$ в оригинале соответствуют в переводе регистры с номерами

$$M = \begin{cases} N + Д - 1, & \text{если } N + Д \leqslant 6; \\ N + Д - 7, & \text{если } N + Д > 6, \end{cases}$$

где «указатель смещения стека» (содержимое регистра $Д$) $Д = 0, 1, \ldots, 5$ в зависимости от смещения стека после выполнения программы. Например, при $Д = 0$ регистрам $CN = C1, \ldots, C6$ соответствуют регистры с номерами $M = 0, 1, \ldots, 5$, а при $Д = 3$ регистрам $C1, \ldots, C6$ — номера $M = 3, 4, 5, 0, 1, 2$.

Операторы набора чисел, синтаксические операторы ↑ и XY, а также одноместные операторы входного языка ЯМК21 семантически и синтаксически совпадают (при установке в микрокалькуляторе с входным языком ЯМК34 переключателя Р—Г в положение Р) с соответствующими операторами входного языка ЯМК34, за исключением оператора e^{jx}, который переводится фрагментом ↑ sin XY cos или sin BX cos.

Однако перевод двухместных операторов с совпадающими символами более сложен, так как должны быть сохранены синтаксические правила языка-оригинала, связанные с особенностями работы операционного стека микрокалькулятора. Например, после выполнения сложения a ↑ b + в микрокалькуляторе с входным языком ЯМК21 содержимое операционных регистров $PX = a + b$, $PY = a$ (см. рис. 12,*ж*), тогда как в микрокалькуляторе с входным языком ЯМК34 после сложения в регистр Y засылается прежнее содержимое регистра Z. Исключение составляет лишь оператор X^y, после выполнения которого содержимое регистров X и Y одинаково в микрокалькуляторах с обоими рассматриваемыми языками.

В микрокалькуляторах с входным языком ЯМК21 вызов операнда из памяти не изменяет содержимого регистра Y, тогда как в микрокалькуляторах с входным языком ЯМК34 вызов из памяти приводит к смещению «вверх» операционного стека. Поэтому оператор FN должен переводиться на входной язык ЯМК34 фрагментом XY ИПМ. Для перевода операторов → и ← обращения к кольцевому стеку памяти целесообразно использовать операторы косвенного обращения к памяти и «лишний» регистр *Д*, обеспечивающий последовательный опрос регистров $M = 0, 1, 2, \ldots, 5$ в соответствии с «переводимыми» изменениями содержимого стека памяти. Возможные варианты перевода регистров → и ← и остальных операторов входного языка ЯМК21 (кроме семантически и лексически совпадающих) на входной язык ЯМК34 приведены в табл. 5.

Длину дословного перевода можно несколько сократить, если учесть типовые «идиоматические» сочетания операторов входного языка ЯМК21, переводимые на входной язык ЯМК34 меньшим числом операторов, чем при дословном переводе каждого из операторов таких сочетаний. Например, сочетание операторов ↑ FN, дословно переводимое фрагментом ↑ XY ИПМ, можно перевести одним оператором ИПМ, а сочетание О ↑ (где О — двухместный оператор) на входной язык целесообразно переводить сочетанием операторов О ↑, так как при дословном переводе каждого из этих операторов длина перевода значительно возрастет. Перевод таких «идиом» также приведен в табл. 5.

Соответствие между адресами и указателями переходов (см. табл. 2) в программах на входных языках ЯМК21 и на ЯМК34 установить несложно, но при переводе изменяется длина программы и поэтому адреса изменяются неоднозначно. Поэтому целесообразно в процессе перевода оставлять после операторов перехода пропуски, заполняемые после перевода тех частей программы, куда передается управление соответствующими адресами в системе адресации входного языка ЯМК34.

5. Словарь перевода операторов и «идиом» с входного языка ЯМК21 на входной язык ЯМК34

ЯМК21	ЯМК34
+	XY + BX XY
—	— ↑ BX + XY
×	XY × BX XY
÷	÷ ↑ BX × XY
e^{jx}	sin BX cos
π	XY π
P1	↑
F1	XY ↑
FN	XY ИПМ (XY ИПN или XY ИП (N + 4)*)
PN	ПМ (ПN или П (N + 4)*)
→	ИПД 1 — $x<0$ [ ] 6 + ПД → КИПД XY КПД →**
←	КИПД XY КПД → ИПД 5 — $x\neq 0$ [ ] 6 + ПД → **
xП0 y_i	xП0 A_i***
БП y_i	БП A_i
ПП y_i	ПП A_i
↑ FN	ИПМ
↑ π	π
O ↑	O ↑ *4
FN_1 ↑ FN_2	$ИПМ_1$ $ИПМ_2$
O PN_1 FN_1 ↑	O $ПМ_1$ $ИПМ_2$ ↑

* При обращении в программе-оригинале к стеку памяти.
** Стеку памяти соответствуют регистры с номерами $M = 0, 1, \ldots, 5$, причем соответствие регистров стека CN регистрам M зависит от «указателя смещения стека», равного содержимому регистра $Д$.
*** Символом П обозначено проверяемое отношение ($\neq$, $=$, $\geqslant$ или $<$), y_i — указатель перехода, A_i — адрес перехода.
*4 O — двухместный арифметический оператор.

При многократном обращении к кольцевому стеку памяти в программе-оригинале целесообразно в программе-переводе фрагменты, реализующие повороты стека, вынести в подпрограммы. В качестве примера в табл. 6 приведен бланк дословного перевода программы 156 [11] для вычисления коэффициентов алгебраического многочлена степени $m \leqslant 5$ при изменении начала отсчета аргумента на величину x_0. Перевод программы длиной 59 операторов занимает 93 шага и

был бы еще длиннее без вынесения в подпрограммы фрагментов перевода операторов обращения к стеку памяти.

Перевод программы в табл. 6 необходимо дополнить следующим переводом инструкции (так как изменяются места хранения данных в памяти): $a_0 = P5$, $a_1 = P4$, $a_2 = P3$, $a_3 = P2$, $a_4 = P1$, $a_5 = P0$ (если $m < 5$, то в соответствующие регистры занести нули) $x_0 = P6$ В/О С/П $P3 = a'_0$, $P2 = a'_1$, $P1 = a'_2$, $P0 = a'_3$, $P5 = a'_4$, $P4 = a'_5$, $РД = 4$. Для проверки перевода можно использовать следующий контрольный пример: при $A(x) = 3x^5 + 5x^4 + 6x^3 + 4x^2 + 3x + 10$ и $x_0 = -3$ должны вычисляться коэффициенты многочлена $A(x) = 3x^5 - 40x^4 + 216x^3 - 590x^2 + 816x - 449$ примерно за 4 мин.

6. Бланк перегода программы 156 [11] с языка ЯМК21 на язык ЯМК34

ЯМК21		ЯМК34		ЯМК21		ЯМК34		ЯМК21		ЯМК34	
Адрес	Оператор	Адрес	Оператор	Адрес	Оператор	Адрес	Оператор	Адрес	Оператор	Адрес	Оператор
00	1	00	1	43	—	33	—	93	←	62	ПП
01	P5	01	П9	44	P4	34	П8			63	79
02	6	02	6	45	P3	35	П7	94	В/О	64	В/О
03	P4	03	П8	50	—	36	ПП			65	ИПД
04	ПП	04	ПП			37	79			66	1
05	F4 (42)	05	31	51	P8	38	ПС			67	—
10	ПП	06	ПП	52	P7	39	ПВ			68	$x < 0$
11	3 (34)	07	27	53	F3	40	ИП7			69	72
12	ПП	08	ПП	54	↑					70	6
13	P÷ (33)	09	25	55	F7	41	ИПВ			71	+
14	ПП	10	ПП	60	×	42	×			72	ПД
15	F3 (32)	11	23	61	↑					73	→
20	ПП	12	ПП	62	F5	43	ИП9			74	КИПД
21	P3 (31)	13	21	63	÷	44	—			75	XY
22	→	14	ПП	64	←	45	ПП			76	КПД
		15	65			46	79			77	→
23	→	16	ПП	65	P7	47	ПВ			78	В/О
		17	65	70	F8	48	ИПС			79	КИПД
24	→	18	ПП	71	↑					80	XY
		19	65	72	F2	49	ИП6			81	КПД
25	С/П	20	С/П	73	×	50	×			82	→
30	←	21	ПП	74	↑					83	ИПД
		22	79	75	F7	51	ИПВ			84	5
31	←	23	ПП	80	+	52	+			85	—
		24	79	81	P8	53	ПС			86	$x \neq 0$
32	←	25	ПП	82	F3	54	ИП7			87	90
		26	79	83	1	55	1			88	6
33	F5	27	ИП9	84	—	56	—			89	+
34	1	28	1	85	P3	57	П7			90	ПД
35	+	29	+	90	$x = 0$	58	$x = 0$			91	→
40	P5	30	П9	91	5 (54)	59	40			92	В/О
41	F4	31	ИП8	92	F8	60	XY				
42	1	32	1			61	ИПС				

Размещение результатов вычисления не в «своих» регистрах объясняется тем, что после выполнения программы «указатель смещения стека» $Д = 4$ и поэтому (согласно ранее принятым правилам) регистрам *С6*, *С5*, ... соответствуют регистры $M = 6 + Д - 7 = 3$; $5 + Д - 7 = 2$, ... При повторном пуске программы с $Д = 4$ она выполнится так же, как и при занесении коэффициентов в регистры *5*, *4*, ..., *0* и $Д = 0$.

Основной недостаток дословного перевода программ заключается в большой длине программы-перевода. Поэтому в тех случаях, когда требуется минимальная длина программы-перевода, прибегают к переводу оригинала по предложениям, разбивая оригинал на имеющие самостоятельные семантическое и синтаксическое значения фрагменты (предложения) и последовательно переводя такие предложения с учетом синтаксических и лексических особенностей обоих языков.

Длина дословного перевода программ с входного языка ЯМК21 особенно увеличивается при наличии в оригинале операторов обращения к кольцевому стеку памяти (см. табл. 6). Однако даже при отсутствии таких операторов дословный перевод программ с входного языка ЯМК21 оказывается, как правило, значительно длиннее перевода по предложениям.

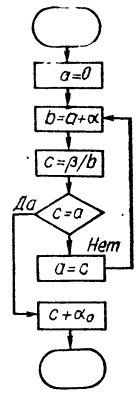


Рис. 22

Для сравнения рассмотрим перевод программы 28 [11] для вычисления бесконечной цепной дроби $x = \alpha_0 + \beta / (\alpha + \beta / (\alpha + + ...)...)$ при исходных данных $\beta = P2$, $\alpha = P3$, $\alpha_0 = P4$ с входного языка ЯМК21 на входной язык ЯМК34. Дословный перевод этой программы занимает 42 шага, вместо 22 в оригинале. Перевод по предложениям, соответствующий блокам алгоритма (рис. 22), по которому составлялась исходная программа, оказывается короче оригинала.

Программа 5/34. Вычисление бесконечной цепной дроби

```
Сх ПА ИПА  ИП3 +   ИП2 XY ÷ ИПА XY
ПА — x = 0 02  ИПА ИП4 +   С/П
```

Инструкция: $\beta = P2$, $\alpha = P3$, $\alpha_0 = P4$ В/О С/П $PX = x$.

При $\beta = 5$, $\alpha = 3$, $\alpha_0 = 4$ по этой программе получим $x = 5{,}1925824$ (время счета около 50 с).

Следует подчеркнуть, что входной язык ЯМК34 отличается достаточно большим словарным запасом и удобен для перевода программ с других языков или на другие языки, особенно в случаях, когда синтаксические правила (и соответствующие правила работы операционных стеков) языков оригинала и перевода близки или совпадают.

В качестве примера рассмотрим возможности взаимного перевода программ, составленных на входном языке ЯМК34 и одного из наиболее сложных и дорогостоящих микрокалькуляторов типа HP41C, допускающего ввод и вывод информации как в цифровом, так и в буквенном виде.

Входной язык HP41C характеризуется обратным синтаксисом одноместных и двухместных операций, а все операторы входного языка ЯМК34 имеют аналоги во входном языке HP41C, показанные в

7. Словарь перевода операторов входного языка ЯМК34 на входной язык HP41C

ЯМК34	HP41C		ЯМК34	HP41C	
	Клавиши	Индикатор		Клавиши	Индикатор
,	.		→	R↓	RDN
ВП	EEX		БП A	GTO M	GTO M
π	π	PI	ПП A	XEQ M	XEQ M
×	×	*	В/О	RTN	RTN
÷	÷	/	$x=0$ A	$x\neq 0$? GTO M	$x\neq 0$? GTO M
X^y	$X \gtrless YY^x$	$X <> YY^x$	$x\neq 0$ A	$x=0$? GTO M	$x=0$? GTO M
			$x\geqslant 0$ A	$x<0$? GTO M	$x<0$? GTO M
x^2	x^2	$X \nearrow 2$	$x<0$ A	$x=0$? GTO M	$x=0$? GTO M
$\sqrt{}$	$\sqrt{x}$	SQRT		$x>0$? GTO M	$x>0$? GTO M
e^x	e^x	$E \nearrow X$	КИПN	RCL IND ON	RCL IND ON
10^x	10^x	$10 \nearrow X$	КПN	STO IND ON	STO IND ON
ln	LN	LN	КБПN	GTO IND ON	GTO IND ON
lg	LOG	LOG	КППN	XEQ IND ON	XEQ IND ON
sin	sin	SIN	Kx=ON	$x\neq 0$? GTO IND ON	$x\neq 0$? GTO IND ON
cos	cos	COS	K$x\neq$ON	$x=0$? GTO IND ON	$x=0$? GTO IND ON
tg	tan	TAN			
arcsin	$\sin^{-1}$	ASIN	K$x\geqslant$ON	$x<0$? GTO IND ON	$x<0$? GTO IND ON
arccos	$\cos^{-1}$	ACOS			
ИПN	RCL ON	RCL ON	K$x<$ON	$x=0$? GTO IND ON	$x=0$? GTO IND ON
ПN	STO ON	STO ON			
$X \rightleftarrows Y$	$X \lessgtr Y$	$X <> Y$		ON $x>0$? GTO IND ON	$x>0$? GTO IND ON
↑	ENTER ↑				
С/П	R/S	R/S (END)			

табл. 7, где не приведены операторы, совпадающие по назначению и начертанию в обоих языках. Основное отличие этих языков связано с передачей управления метками и большим словарным запасом языка HP41C (имеющим, в частности, операторы установки, проверки и стирания флагов), а также значительно большей емкостью памяти. В этом языке метки вводятся в виде двузначных чисел или сочетаний, содержащих до шести букв и цифр, причем перед отмеченными операторами и меткой вводится символ LBL. При невыполнении условия, проверяемого условным оператором, управление передается следующему оператору, а при выполнении — отмеченному оператору с помощью оператора безусловного перехода GTO и метки, записываемых после условного оператора.

С учетом этих различий перевод несложных программ с входного языка ЯМК34 на язык HP41C проще всего выполнять дословно. Например, вычисление функции $\sin x/x$ при хранении аргумента x в регистре 0 и раскрытии неопределенности для $x = 0$ на входном языке ЯМК34 реализуется фрагментом

ИПО sin Вх $x = 0$ 07 1 ↑ ÷ ...

с передачей управления оператору ÷ и вычислением функции $\sin x/x$ при невыполнении проверяемого условия $x = 0$ и набором 1 в реги-

стре X при его выполнении. При дословном переводе этого фрагмента на входной язык HP41C оператор $x = 0$ следует заменить обратным по смыслу оператором $x \neq 0$ (чтобы при выполнении условия $x = 0$ следующим выполнялся оператор, записанный в программе-оригинале после оператора условного перехода), а перед оператором деления следует ввести метку, например, 01:

RLC 00 sin LASTX $x \neq 0$? GTO 01 1 ↑ LBL 01 ÷ ...

Такой перевод длиннее оригинала в связи с большим числом команд для выполнения условного оператора и передачей управления метками во входном языке HP41C.

В случае перевода по предложениям программы, составленной на входном языке HP41C, на входной язык ЯМК34 (при возможности перевода одной программой или пакетом программ) основные затруднения возникают при переводе операторов установки, снятия и проверки флагов, а также команд проверки выполнения условий вида $x\Pi y$?

Для перевода операторов флага следует выделить регистр N памяти микрокалькулятора с входным языком ЯМК34. Тогда оператор SF установки флага можно перевести, например, фрагментом 1 ПN →, оператор CF снятия флага — фрагментом 0 ПN → , а оператор FS? проверки флага — фрагментом ИПN $x = 0$ A → (оператор → следует ввести и перед оператором, которому передается управление при установленном флаге) или ИПN $x = 0$ A, если после проверки флага прежнее содержимое операционных регистров не используется.

Команду вида $x\Pi y$? можно переводить различным образом в зависимости от последующей программы. В простейшем случае, когда далее прежнее содержимое регистров X и Y не используется, эту команду можно перевести фрагментом — $x\overline{\Pi}0$ A, где $\overline{\Pi}$ —отношение ($\neq$, $=$, $\geqslant$, $<$), обратное по смыслу, проверяемому в оригинале. Если содержимое регистров операционного стека используется в последующих частях программы, то после указанного фрагмента необходимо организовать восстановление содержимого регистров операционного стека.

В тех случаях, когда синтаксические правила входных языков оригинала и перевода существенно отличаются (в частности, правилами выполнения функциональных операторов) и дословный перевод или перевод по предложениям выполнить затруднительно, прибегают к «пересказу» программы. Применительно к входным языкам микрокалькуляторов пересказ программы заключается в восстановлении по программе-оригиналу описания или схемы алгоритма решения задачи с последующим составлением по этому алгоритму (при необходимости с его коррекцией) программы-пересказа на другом входном языке. Затраты времени в этом случае могут оказаться бóльшими, чем при дословном переводе программы или переводе по предложениям, но меньше, чем при составлении программы, начиная с выбора исходной модели, метода и алгоритма решения задачи.

5. ПРОВЕРКА И ОТЛАДКА ПРОГРАММ

Составленные на бумаге сложные программы при их первом выполнении на микрокалькуляторе в редких случаях приводят к ожидаемым результатам. Обычно такие программы приходится по частям и целиком проверять и отлаживать при помощи микрокалькулятора. В этом случае необходимо прежде всего убедиться в правильности ввода операторов в программную память, для чего в режиме программирования и предусмотрена индикация кодов вводимых операторов и содержимого счетчика шагов.

Высвечиваемые при вводе программы коды сравнивают с вводимыми операторами по таблицам соответствия, приводимым в инструкциях по эксплуатации программируемых микрокалькуляторов. При составлении и вводе программ на входном языке ЯМК21 основные затруднения связаны с определением символов (нажимаемых клавиш) и десятичных кодов $ab' = a\,(b+1)$ указателей условных и безусловных переходов по адресам ab. Для этого можно использовать таблицы соответствия, подобные табл. 2, или трафареты, подобные описанным в работе [11]. Вместо этого можно воспользоваться табл. 8, в которой на пересечении строк с номерами a и столбцов с номерами b записаны указатели переходов по адресам ab. Высвечиваемые в режиме программирования коды ab' адресов совпадают с кодами других операторов, вводимых нажатием тех же клавиш. Исключение составляют операторы НОП и $X,^{y}$ а также операторы управления выполнением программы (см. табл. 2).

8. Соответствие символов указателей перехода адресам перехода и высвечиваемым кодам во входном языке ЯМК21

b'	1	2	3	4	5	6	1	2	3	4	5	6	b'
b / a	0	1	2	3	4	5	0	1	2	3	4	5	b / a
0	P0	F0	P↑	0	F↑	↑	P1	F1	PXY	1	FXY	XY	1
2	P2	F2	P×	2	F×	×	P3	F3	P÷	3	F÷	÷	3
4	P4	F4	P,	4	F,	,	P5	F5	P/—/	5	F/—/	/—/	5
6	P6	F6	РВП	6	FВП	ВП	P7	F7	PCx	7	FCx	Cx	7
8	P8	F8	P—	8	F—	—	P9	F9	P+	9	F+	+	9

Затраты времени на сверку высвечиваемых кодов с символами операторов и указателей переходов по вводимой программе существенно уменьшаются при запоминании пользователем этих кодов. Для микрокалькуляторов с входным языком ЯМК21 следует запомнить прежде всего указатели У перехода по адресам вида $a5 = 05, 15, \ldots, 95$, вводимые нажатием соответственно клавиш ↑ | XY | × | ÷ |, |/ — / | ВП

| Cx| — |+ (легче всего запомнить последовательность этих клавиш по их расположению на клавиатуре). Запомнив указатели У, легко определить указатели РУ и FУ переходов по адресам вида $a2$ и $a4$ соответственно, а переходам по адресам вида $a0$, $a1$ и $a3$ соответствуют указатели переходов Pa, Fa и a.

Во входном языке ЯМК34 указатели адреса и их коды совпадают с адресами перехода и достаточно запомнить коды операторов. Для этого следует прежде всего учесть, что в этом языке двухзначные коды $a'b'$ операторов определены в 16-ричной системе счисления с элементами из последовательности десятичных цифр и дополнительных символов, близких по начертанию к знакам | — | L |C | Г | E | соответственно. При этом различным группам операторов соответствуют различные кодовые сочетания:

1. Коды вида $0b'$ присвоены операторам набора чисел и оператору ↑, причем цифры b' соответствуют операторам набора тех же цифр, а остальные элементы системы счисления — операторам |, | /—/| ВП | Cx| ↑ |. Оператору Bx присвоен код 0 (второй символ не высвечивается).

2. Коды вида $1b'$ в основном присвоены функциональным операторам + (10), — (11), × (12); ÷ (13), XY (14), 10^x(15), e^x (16), lg (17), ln (18), arcsin (19), arccos (1—), arctg (1L), sin (1C), cos (1Г), tg (1E).

3. Коды вида $2b'$ соответствуют оператору π (код 20) и операторам вычисления показательных функций $\sqrt{}$ (21), x^2 (22), $1/x$ (23), X^y(24).

4. Коды вида $4b'$ соответствуют операторам ПN прямой засылки содержимого регистра X в регистр N памяти, причем регистрам $N = A$, B, C и $Д$ соответствуют коды $b' =$ | — | L | C | $Г$ |.

5. Коды вида $5b'$ соответствуют операторам и командам управления выполнением программы С/П (50), БП (51), В/О (52), ПП (53), НОП (54), $x \neq 0$ (57), L2 (58); $x \geqslant 0$ (59), L3 (5—), L1 ($5L$), $x < 0$ (5C), L0 (5Г), $x = 0$ (5E).

6. Коды вида $a'N$ соответствуют операторам ИПN ($6N$) прямого вызова содержимого регистра N памяти (регистрам $N = A, B, C, Д$ соответствуют коды | —| L | C | Г |) и операторам косвенной адресации Kx = 0N ($7N$), КБПN ($8N$), K$x \geqslant$ 0N ($9N$), КППN ($-N$), КПN (LN), K$x <$ 0N (CN), КИПN ($ГN$) и Kx = 0N (EN).

Практически достаточно запомнить следующее:

1. В кодах операторов набора чисел первый символ 0.

2. Коды арифметических операторов +, —, ×, ÷ соответственно равны 10, 11, 12 и 13.

3. Коды операторов управления программой начинаются с цифры 5, причем командам С/П, БП, В/О и ПП соответствуют коды 50, 51, 52 и 53.

4. Коды вида $4N$ и $6N$ соответствуют операторам прямых, а коды вида LN и $ГN$ — косвенных засылок в память и вызова из нее, причем регистрам $N = A$, B, C, $Д$ соответствуют коды | — | L | C | $Г$ |.

По этим правилам легко обнаружить большинство ошибок при вводе программы, не обращаясь к таблицам соответствия операторов их кодам. В частности, если при наборе функционального оператора первым в коде является нуль, то пропущена префиксная клавиша F,

если же при вводе операторов косвенной адресации первая цифра кода меньше 5, то пропущена префиксная клавиша К.

При вводе отлаженной «библиотечной» программы в компактной записи с четким и внимательным нажатием клавиш исправного микрокалькулятора нет необходимости в тщательной сверке высвечиваемых кодов. А так как основной ошибкой является пропуск оператора, то при вводе отлаженной программы на входном языке ЯМК21 в компактной записи по 12 шагов в строке достаточно контролировать лишь высвечиваемое в знакоместах порядка содержимое регистра-счетчика шагов после ввода крайних правых операторов строк (высвечиваются адреса 20, 40, 60, 80 и —0). В этом случае на ввод программы с максимальным числом шагов затрачивается не более полутора минут. Аналогично при вводе отлаженной программы на входном языке ЯМК34 в компактной записи по 10 шагов в строке достаточно контролировать адреса следующих шагов после ввода крайних правых шагов в строках программы (10, 20, ..., 90 и наибольший адрес 98).

Однако при проверке и отладке составляемой программы необходимо тщательно проверять по высвечиваемым кодам правильность ввода каждого оператора, чтобы исключить ошибки в программе, заносимой в программную память. В программируемых микрокалькуляторах с входными языками ЯМК21 и ЯМК34 предусмотрена возможность пошаговой отладки программы или ее фрагмента, введенных в программную память, в рабочем режиме нажатием клавиш В/О (если выполнение программы проверяется с начала) или БП *А* (если выполнение программы проверяется с адреса *А*) и многократным нажатием клавиши ПП для поочередного выполнения шагов программы и проверки результатов их выполнения по изменению высвечиваемого содержимого регистра *X*. Для контроля следует заранее определить и записать содержимое регистра *X* после выполнения каждого шага составленной программы.

При подготовке таких контрольных данных и отладке программ следует учитывать специфические особенности работы операционного стека и пошагового выполнения микрокалькулятором операторов программы. В частности, при нажатии клавиши для ввода команды безусловного перехода БП или ПП и команд проверки условий в операторах условного перехода высвечивается предыдущее содержимое регистра *X*, но при следующем нажатии клавиши ПП, соответствующем вводу указателя перехода, выполняется оператор программы, которому передается управление и на индикаторе высвечивается результат выполнения этого оператора. Аналогично после выполнения последнего оператора подпрограммы и очередного нажатия клавиши ПП при шаговой проверке выполняется оператор, записанный в программе после оператора обращения к подпрограмме.

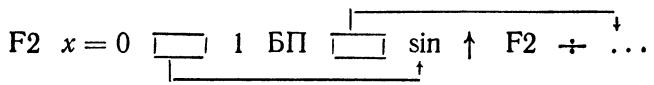
Если содержимое регистра *X* при очередном нажатии клавиши ПП отличается от контрольного значения, то следует перейти в режим программирования и проверить по содержимому счетчика шагов и высвечиваемым кодам правильность записи в программе соответствующего оператора. Особенно тщательно следует проверять содержимое счетчика шагов после выполнения переходов, так как при ошибках

во вводе программы или исходных данных передача управления также может оказаться ошибочной.

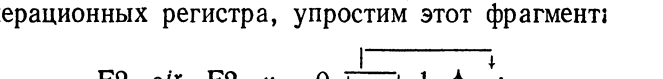
При правильности ввода проверяемого шага программы следует снова перейти в рабочий режим и тщательно повторить выполнение по шагам фрагмента программы, оканчивающегося проверяемым шагом, для обнаружения и устранения причины расхождения контрольных и высвечиваемых данных. При повторной проверке необходимо восстановить исходное состояние регистров операционного стека и используемых регистров памяти. В разветвленных программах должны быть проверены все ветви, связанные с безусловными и (при изменении исходных данных, обеспечивающих выполнение и невыполнение проверяемых условий) условными переходами.

Особое значение при отладке программы по шагам имеет выбор исходных данных, которые должны быть немногозначными для удобства проверки результатов операций в уме или обычном режиме и подобраны так, чтобы результаты выполнения соседних операторов отображались в регистре X различными числами, так как в противном случае окажется невозможным определить источник ошибки. Следует также избегать таких исходных данных, при которых оказываются одинаковыми результаты выполнения различных операций (например, $2 \times 2 = 2 + 2 = 2^2$ или $1 \times 1 = 1^k = 1$) или ввода различных коэффициентов (например, $1 = 1^k$ при любом значении k).

Если в программе реализовано вычисление функции, имеющей особые точки, то при отладке такой программы необходимо проверить правильность ее выполнения при исходных данных, соответствующих таким особым точкам. Допустим, например, что в программе реализовано вычисление функции $\sin x/x$ по значению аргумента, вычисленного по предыдущей части программы. Если такой аргумент может принимать нулевые значения, то это приведет к переполнению, хотя при других значениях аргумента функция будет вычислена правильно. Для получения правильных результатов в особой точке программу следует разветвить, обеспечив вычисление функции и при $x = 0$. Такой фрагмент на входном языке ЯМК21 (при $x = P2$) можно составить в следующем виде:



Учитывая, что операторы sin ↑ можно заменить оператором e^{jx}, а вместо безусловного перехода можно организовать засылку единицы в оба операционных регистра, упростим этот фрагмент:



Если подобные разветвления для особых точек ввести не удается в связи с ограниченной емкостью программной памяти, то в инструкции к программе следует указать, при каких значениях исходных данных программа не выполняется (или выполняется с неверным или неточным результатом).

При отладке программ, реализующих методы последовательных приближений, необходимо убедиться в надежности выхода из цикла по выбранному критерию и в необходимых случаях указать в инструкции к библиотечной программе условия, при которых возможно «зацикливание» программы.

Проверка и отладка на микрокалькуляторе сложных программ ускоряется при выполнении их отдельных фрагментов и сверке полученных промежуточных результатов с контрольными данными. Для этого в конце проверяемого фрагмента следует временно ввести оператор С/П, который после проверки фрагмента заменяется соответствующим шагом программы.

При составлении и отладке программы с помощью микрокалькулятора возникают ситуации, когда составляемый фрагмент может быть уточнен только после выполнения последующих. В этих случаях целесообразно записать подпрограммы в последние ячейки программной памяти и «разнести» остальные фрагменты программы, оставляя между ними незаполненные ячейки программной памяти для возможной записи нужных команд. Для пропуска пустых ячеек в режиме программирования микрокалькулятора с входным языком ЯМК21 достаточно соответствующее число раз нажать клавишу $\overrightarrow{ШГ}$. Для микрокалькулятора с входным языком ЯМК34 пропущенные шаги должны быть заполнены операторами НОП, так как содержимое 00 свободных ячеек управляющее устройство воспримет как оператор набора цифры 0 с кодом 00.

После отладки программу переписывают без пропусков и операторов НОП, изменив при необходимости адреса переходов, и окончательно проверяют ее выполнение в программируемом режиме. Библиотечные программы должны снабжаться контрольными примерами, обеспечивающими проверку выполнения всех разветвлений алгоритма.

Последним этапом отладки большинства программ является аттестация достижимой точности результата вычислений, особенно актуальная при реализации методов последовательных приближений.

В качестве примера рассмотрим оценку погрешности результата выполнения программы для вычисления интеграла вероятности

$$\Phi(x) = \sqrt{2/\pi} \int_0^x e^{-u^2/2}\, du \qquad (3.9)$$

по его разложению в ряд

$$\Phi(x) = \sqrt{2/\pi}\,(x - x^3/1!2 \cdot 3 + x^5/2!4 \cdot 5 - x^7/3!7 \cdot 8 + \ldots)$$

на микрокалькуляторе с входным языком ЯМК34 (аналогичная программа на входном языке ЯМК21 приведена в работе [13]).

Представим аппроксимирующий ряд в форме

$$\Phi(x) = \sqrt{2/\pi} \sum_{k=0}^{\infty} \varphi_k/(2k+1),$$

где $\varphi_0 = x$, а каждый следующий член ряда вычисляют по рекуррентной формуле $\varphi_k = -\varphi_{k-1}x^2/2k$, и распределим регистры памяти для хранения данных —

регистр *0* для накопления суммы S_k членов ряда, регистр *1* для хранения k-го члена ряда, регистр *2* для x^2 и регистр *4* для номера k суммируемого члена ряда. Реализуем рекуррентное соотношение фрагментом

ИП1 /—/ ИП2 × ИП4 2 × ÷ П1 ВХ 1 + ÷ ...,

после каждого выполнения которого значение k должно увеличиваться на единицу. Поэтому регистр *4* можно использовать в операторе КИП4, восстанавливая содержимое регистров X, Y и Z операционного стека оператором →. Организовав суммирование членов ряда в регистре *0*, для достижения максимальной точности результата указанный фрагмент следует охватить циклом с выходом из него, когда очередной член суммы станет меньше ее на восемь порядков (станет относительным машинным нулем). В этом случае разность очередных значений S_k — S_{k-1} = 0 и дальнейшие вычисления в цикле теряют смысл. Используя это условие выхода из цикла, составим рабочую часть программы:

КИП4	ИПО	ИП1	/—/	ИП2	×	ИП4	2	×	÷
П1	ВХ	1	+	÷	+	ПО	—	$x=0$	А

с адресом A перехода к начальному оператору фрагмента. Перед выполнением этого фрагмента должно быть $x = PO = P1$, $x^2 = P2$, $0 = P4$, а после его выполнения накопленную в регистре O сумму следует умножить на $\sqrt{2/\pi}$. С учетом этих дополнений получим окончательно следующую программу.

Программа 6/34. Вычисление интеграла вероятности $\Phi(x)$ по разложению в степенной ряд

ПО	П1	x^2	П2	Сх	П4	КИП4	ИПО	ИП1	/—/
ИП2	×	ИП4	2	×	÷	П1	Вх	1	+
÷	+	ПО	—	$x=0$	06	ИПО	2	π	÷
$\sqrt{}$	×	С/П	БП	00					

Инструкция: $x = PX$ (В/О) С/П $PX = \Phi(x)$.

Для проверки вычислим $\Phi(0, 1) = 0{,}079655672$ (за 20 с) и $\Phi(1) = 0{,}68268947$ (за 55 с). По содержанию регистра 1 определяем, что для $x = 0{,}1$ вычислялись три члена суммы, а для $x = 1$ — девять. Следовательно, на вычисление одного члена затрачивается приблизительно $(55 - 20) / 6 \approx 5{,}8$ с и на выполнение вспомогательных операций $20 - 5{,}8 \times 3 \approx 2{,}6$ с. В контрольном примере к программе следует привести несколько контрольных значений с указанием времени их вычисления.

Значение $\Phi(x) = 1$ при стремлении x к бесконечности и поэтому можно полагать, что суммирование прекращается при $x > 1$, когда

$$(x^{2k+1}/2^k)/k!\,(2k+1) \leqslant (4/9) \times 10^{-8},$$

так как с учетом округления $S_{k+1} - S_k = 0$, когда очередной член меньше суммы на величину $(4/9) \cdot 10^{-8}$. С помощью этого соотношения легко решить обратную задачу — по заданному k найти максимальное значение x, при котором достигается максимальная остаточная погрешность суммирования,

$$x \leqslant e^{k/2}\, e^{(k(\ln 2-1)+\ln(2k+1))/(2k+1)}/9 \cdot 10^7,$$

где факториал k вычислен по формуле (3.8).

Для приближенной оценки это соотношение можно упростить:

$$x \leqslant e^{k/2}/(10^4)^{1/k}.$$

Отсюда видно, что, например, для вычисления значения $\Phi(x)$ при $x \approx e^{5/2}/10^{0{,}16} \approx 8{,}42$ потребуется суммирование около 25 членов ряда и время выполнения программы окажется приблизительно равным $25 \times 5{,}8 + 2{,}6 \approx 148$ с.

Сравнивая приведенные выше вычисленные значения $\Phi(0{,}1)$ и $\Phi(1)$ с табличными значениями, нетрудно убедиться, что они содержат семь верных цифр.

Однако вывод о столь высокой точности при произвольном значении аргумента преждевременен. Действительно, вычислив Φ (5,327) = 0,99813267 по программе 6/34 и сравнив это значение с табличным Φ (5,326) = 0,9999999, убеждаемся, что неверной будет уже третья цифра после запятой. Причина роста погрешности в данном случае связана с характером изменения величины членов суммируемого знакопеременного ряда, которые при больших значениях аргумента вначале возрастают по модулю и, лишь достигнув некоторого максимального значения, начинают уменьшаться.

Оценим величину максимального по модулю члена ряда, приравняв два соседних члена a_{k-1} и a_k:

$$(x^{2k-1}/2^{k-1})/(k-1)!\,(2k-1) = (x^{2k+1}/2^k)/k!\,(2k+1)$$

или

$$2k\,(2k+1)/(2k-1) = x^2,$$

откуда

$$k = \mathrm{E}\,(((x^2-1)+\sqrt{x^4-6x^2+1})/4),$$

где E — символ целой части, или при $x \gg 1$

$$k \approx \mathrm{E}\,((x^2-1)/2).$$

Величина максимального по модулю члена

$$a_{k\,\max} \approx x\,(x^2/2)^{(x^2-1)/2}/(x^2-1)/2!\,(x^2-1) \approx$$
$$\approx (x/\sqrt{\pi\,(x^2-1)^3})\,\mathrm{e}^{(x^2-1)/2} \approx (1/x^2\sqrt{\pi})\,\mathrm{e}^{x^2/2},$$

что при $x = 5{,}327$ составит $a_{k\,\max} \approx 28868{,}974$.

Этот член, содержащий при разрядности $r = 8$ только три знака после запятой, не может быть вычислен точнее, чем с абсолютной погрешностью 10^{-3}. Следовательно, предельная погрешность результата вычислений соответствует его десятичному представлению, содержащему не более двух верных цифр после запятой.

Если, как в рассмотренном примере, предельная погрешность велика и превышает требуемую, то необходимо выбрать другой алгоритм вычислений, обеспечивающий заданную точность результата.

Глава 4

ИНЖЕНЕРНЫЕ РАСЧЕТЫ ПО АНАЛИТИЧЕСКИМ ВЫРАЖЕНИЯМ

1. ПОДГОТОВКА РАСЧЕТНЫХ ФОРМУЛ ДЛЯ ПРОГРАММИРОВАНИЯ

Задачу инженерного проектирования сложного объекта обычно разбивают на ряд более простых задач, заключающихся в определении технических заданий на отдельные узлы объекта с их последующим расчетом. Оптимальное, т. е. удовлетворяющее всем требованиям технического задания решение такой частной задачи в общем случае находят подбором (оптимизацией) схемы и компонентов проектируемого узла. Целенаправленность подбора контролируют, вычисляя на каждом шаге оптимизации характеристики узла и сравнивая их с заданными.

Обычно математическую модель узла составляют методами анализа в виде системы уравнений (2.6) с буквенными обозначениями коэффициентов и переменных, а затем по уравнениям этой системы находят формулы для расчета искомых характеристик в виде аналитических функций переменных и параметров. Инженер использует эти формулы для качественной оценки зависимости характеристик от переменных и параметров и (после замены их символов числами) для вычисления характеристик.

При вычислениях без применения ЭВМ приходится обращаться к таблицам элементарных или специальных функций, что увеличивает затраты времени, использовать различные номограммы (чаще всего логарифмическую линейку) и упрощать расчетные формулы, что снижает точность расчета. Применение микрокалькуляторов может существенно уменьшить затраты времени, практически устраняя необходимость обращения к таблицам функций, и повысить точность результатов вычислений. Для эффективной реализации этой возможности следует выбирать достаточно точные математические модели проектируемых узлов и готовить расчетные формулы для составления оптимальных программ вычислений.

Расчетная формула является одной из форм представления алгоритма вычислений. Для программной реализации такого алгоритма расчетное выражение должно содержать символы только тех математических операций, выполнение которых предусмотрено входным языком используемого микрокалькулятора. Поэтому при подготовке расчетной формулы для программирования символы специальных функций и функционалов (например, интегралов), операторы вычисления которых отсутствуют во входном языке, должны быть заменены выражениями, непосредственно реализуемыми на входном языке. При этом часто приходится использовать несколько программ, что увеличивает затраты времени. Для их уменьшения следует попытаться использовать упрощенные выражения, обеспечивающие требуемую точность вычисления аппроксимируемых функций или функционалов только в заданных интервалах изменения их аргументов.

Сложные расчетные формулы необходимо преобразовать, выделяя однотипные по структуре выражения для их программной реализации циклами, подпрограммами, общими ветвями программы или повторно выполняемыми (при изменении переменных) фрагментами и используя приемы оптимизации программ, рассмотренные в гл. 3. Критериями целесообразности таких преобразований являются критерии оптимальности программы.

Программную реализацию часто удается упростить, выделяя в расчетной формуле одинаковые выражения с одинаковыми результатами вычислений. В этом случае после замены таких однотипных выражений символами (идентификаторами) результатов их вычислений расчетную формулу дополняют формулами для вычисления этих результатов.

В качестве примера рассмотрим задачу вычисления ослабления мощности при передаче электрической энергии от источника с внутренним сопротивлением

$R_и$ в нагрузку с сопротивлением $R_н$ через *аттенюаторы* (ослабители), эквивалентные схемы которых показаны на рис. 23.

Ослабление мощности в децибелах при передаче электрической энергии оценивают по формуле

$$G = 20 \lg(1/K) - 20 \lg((R_н + R_и)/R_н),$$

но для вычислений по этому выражению необходимо предварительно выразить коэффициент передачи напряжения $K = U_н/U_и$ через известные параметры электрической цепи.

Составим математическую модель цепи с T-образным аттенюатором (рис. 23, *а*) в виде системы уравнений контурных токов

$$I_1 (R_и + R_1 + R_2) - I_2 R_2 = U_и;$$
$$-I_1 R_2 + I_2 (R_2 + R_3 + R_н) = 0.$$

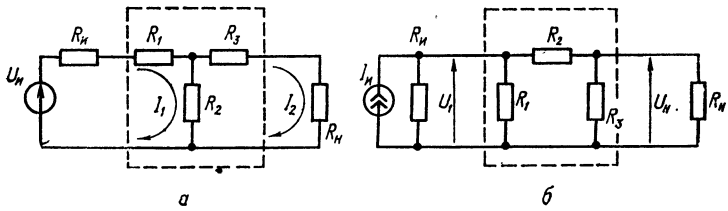


Рис. 23

Подставив в первое уравнение выражение для I_1 из второго, получим выражение

$$U_и = I_2 ((R_2 + R_3 + R_н)(R_и + R_3) + R_2 (R_3 + R_и))/R_2.$$

Так как падение напряжения на сопротивлении нагрузки $U_н = I_2 R_н$, то искомое значение

$$1/K = U_и/U_н = ((R_2 + R_3 + R_н)(R_и + R_3) + R_2 (R_3 + R_н))/(R_2 R_н)$$

и ослабление мощности в цепи с Т-образным аттенюатором

$$G_т = 20 \lg (1 + R_3/R_4 + (R_и + R_1)(1/R_н + 1/R_2 + R_3/(R_2 R_н))) - 20 \lg ((R_и + R_н)/R_н).$$

Для упрощения анализа цепи с П-образным аттенюатором заменим источник напряжения $U_и$ источником тока $I_и = U_и/R_и$ с внутренней проводимостью $1/R_и$ (рис. 23, *б*) и составим систему уравнений узловых напряжений

$$U_1 (1/R_и + 1/R_1 + 1/R_2) - U_н/R_2 = U_и/R_и;$$
$$-U_1/R_2 + U_н (1/R_н + 1/R_3 + 1/R_2) = 0.$$

Подставив выражение для U_1 из второго в первое уравнение, получим

$$U_и/R_и = U_н ((1/R_и + 1/R_1 + 1/R_2)(1/R_3 + 1/R_н) R_2 + 1/R_и + 1/R_1),$$

откуда ослабление мощности

$$G_п = 20 \lg (1 + R_2/R_3 + R_2/R_н + R_и (1/R_1 + R_2 (1/R_3 + 1/R_н)/R_1)) + R_и (1/R_3 + 1/R_н) - 20 \lg ((R_и + R_н)/R_н).$$

Непосредственная реализация полученных формул для G_T и $G_П$ одной программой на входных языках ЯМК34 или ЯМК21 невозможна. Это подтверждается приведенной в статье [16] подобной программой на входном языке микрокалькулятора HP41C длиной 103 шага, что превышает емкость программной памяти микрокалькуляторов с входными языками ЯМК21 и ЯМК34. Поэтому выделим одинаковые выражения с идентификаторами A, B и C и преобразуем исходные расчетные формулы в одинаковые по структуре выражения

$$G_T = 20 \lg (((1 + AR_1/R_2)\, BR_3 + AR_1)/C);$$
$$G_П = 20 \lg (((1 + AR_2/R_и)\, BR_и + AR_н)/C),$$

где $A = 1 + R_и/R_1$, $B = 1 + R_н/R_3$, $C = R_н + R_и$.

В связи с однотипностью структуры этих формул можно вычислить G_T и $G_П$, дважды выполняя один фрагмент при изменении значений переменных. В этом случае обе формулы реализуются на входном языке ЯМК34 программой длиной только в 40 шагов.

Программа 7/34. Расчет ослабления мощности в электрических цепях с Т- и П-образными аттенюаторами

ИП3	ИП1	П5	ИП2	÷	ИПО	ИП1	÷	1	+
П6	×	×	+	ИП4	ИП3	÷	1	+	×
ИП5	ИП6	×	+	ИПО	ИП4	+	÷	lg	2
0	×	С/П	ИП4	П5	ИПО	ИП2	ИПО	БП	04

Инструкция: ($R_и = P0$, $R_1 = P1$, $R_2 = P2$, $R_3 = P3$, $R_н = P4$) В/О С/П $PX = G_T$ С/П $PX = G_П$; если требуется вычислить только $G_П$, то после ввода исходных данных достаточно нажать клавиши БП З З С/П.

Контрольный пример: при $R_и = 100$ Ом, $R_1 = 75$ Ом, $R_2 = 38$ Ом, $R_3 = 19$ Ом, $R_н = 50$ Ом получим $G_T = 11{,}46924$ дБ, $G_П = 14{,}559761$ дБ.

Программа 8/21. Расчет ослабления мощности в электрических цепях с Т- и П-образными аттенюаторами

F8	↑	→	F7	+	P6	F3	÷	1	+
P8	F7	↑	→	F4	÷	1	+	P7	F7
↑	F5	×	→	F4	×	↑	F2	÷	1
+	↑	F3	×	↑	F8	×	↑	←	+
↑	F6	÷	ln	8	,	6	8	6	×
С/П	F2	P4	←	P2	P3	←	P5	БП	F3

Инструкция: $R_и = P7$, $R_1 = P4 = P5$, $R_2 = P2$, $R_3 = P3$, $R_н = P8$ В/О С/П $PX = G_T$ С/П $PX = G_П$.

При исходных данных из контрольного примера к программе 7/34 по этой программе получим $G_T = 11{,}46939$ дБ и $G_П = 14{,}55993$ дБ.

В программе на входном языке микрокалькулятора HP41C, приведенной в статье [16], предусмотрены автоматическое распределение исходных данных в регистрах памяти, вычисление G_T или $G_П$ при вводе кода 0 или 1 соответственно и ограничение разрядности результатов вычислений двумя значащими цифрами после запятой. Все эти особенности можно реализовать на входном языке ЯМК34 с помощью следующей программы, содержащей только 59 шагов,

ПД	1	0	П0	С/П	КП0	L0	04	ИПД	$x \neq 0$
53	ИП1	ПА	ИП9	ИП5	ИП9	÷	ИП9	ИП7	÷
1	+	ПВ	×	×	+	ИП1	ИП3	÷	1
+	×	ИПА	ИПВ	×	+	ИП9	ИП1	+	÷
lg	2	0	×	1	ВП	5	+	ВХ	−
С/П	БП	00	ИП3	ИП7	ПА	ИП5	БП	16	

с инструкцией: $0 = PX$ для вычисления G_T или $1 = PX$ для вычисления $G_П$; В/О С/П $R_и = PX$ С/П $R_1 = PX$ С/П $R_2 = PX$ С/П $R_3 = PX$ С/П $R_н = PX$ С/П $PX = G_T$ или $PX = G_П$.

Воспользовавшись преобразованными расчетными формулами, можно значительно сократить и длину программы на входном языке микрокалькулятора HP41C. Если для сохранения инструкции и идентификаторов программы сохранить неизменными ее первые 23 и последний шаги, то модификация программы будет содержать 62 шага вместо 103. Так как в этом входном языке управление передается по меткам, то при компактной записи программы допустимо различное число шагов в строках.

Программа 9/HP41C. Расчет ослабления мощности в электрических цепях с Т- и П-образными аттенюаторами

```
LBL ATTEN CLX  CF 01  STOP  X = 0?  SF 01  STOP  LBL A   STO
00 STOP LBL B  STO 01  STOP  LBL C  STO 02  STOP  LBL D  STO
03 STOP LBL E  STO 04  FS? 01  GTO 01  LBL 02  RCL 03  RCL 01
STO 05  RCL 02  ÷  RCL 00  RCL 01  ÷  1  +  STO 06  ×  ×  +
RCL 04  RCL 03  ÷  1  +  ×  RCL 05  RCL 06  ×  +  RCL 00
RCL 04  +  ÷  LOG  20  ×  STOP  LBL 01  RCL 00  RCL 02
RCL 00 GTO 01  END
```

Инструкция: RTN R/S $0 = PX$ для вычисления G_T или $1 = PX$ для вычисления $G_П$, $R_и = PX$ A $R_1 = PX$ B $R_2 = PX$ C $R_3 = PX$ D $R_н = PX$ E $PX = G_T$ или $PX = G_П$. При исходных данных из контрольного примера к программе 7/34 по двум последним программам получим одинаковые результаты $G_T = 11{,}47$ дБ; $G_П = 14{,}56$ дБ.

Результаты, полученные в рассмотренном примере, убедительно подтверждают целесообразность тщательной подготовки расчетных формул для их оптимальной программной реализации.

2. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ

Исследуя свойства проектируемого объекта, инженер представляет его вычисляемые характеристики в табличной или менее точной, но более наглядной графической форме. При построении графических моделей функциональных зависимостей вида $y = f(x)$ чаще всего используют прямоугольную систему координат с отсчетом значений аргумента x по оси абсцисс и функции y по оси ординат. Однако соответствия этих значений x и y длинам x_m и y_n отрезков координатных осей (шкалы) могут быть различными и в общем случае

$$x_m = M(x - x_0), \quad y = N(y - y_0), \tag{4.1}$$

где M и N — операторы, ставящие длины x и y в соответствие значениям x и y, а x_0 и y_0— значения аргумента и функции в начале координат.

В простейшем случае операторы $M = m$ и $N = n$ являются постоянными числами (масштабными множителями) и формулы (4.1) определяют равномерные линейные шкалы (рис. 24, *а*), которые при $x_0 = y_0 = 0$ называют пропорциональными (рис. 24,*б*). Иногда используют и более сложные шкалы, среди которых в инженерной практике наиболее употребительны полулогарифмические

$$x_m = m \log_a (x - x_0), \quad y_n = n(y - y_0)$$

и логарифмические

$$x_m = m \log_a (x - x_0),\ y_n = n \log_b (y - y_0),$$

где m и n — масштабные множители; a и b — основания логарифмов.

Построение графика функции $y = f(x)$ начинают с выбора масштабных множителей m и n (обеспечивающих отображение на чертеже определенного размера заданных интервалов изменения аргумента и возможных интервалов изменения функции) и, при необходимости, x_0 и y_0. Определив точки отсчета аргумента x_i и соответствующие значения x_{mi} на оси абсцисс, вычисляют значения функции $y_i = f(x_i)$ и отмечают на графике точки с координатами x_{mi} и y_{ni}. После вычисления всех значений функции в выбранном интервале изменения аргумента эти точки соединяют линией, которая и отображает искомую функциональную зависимость.

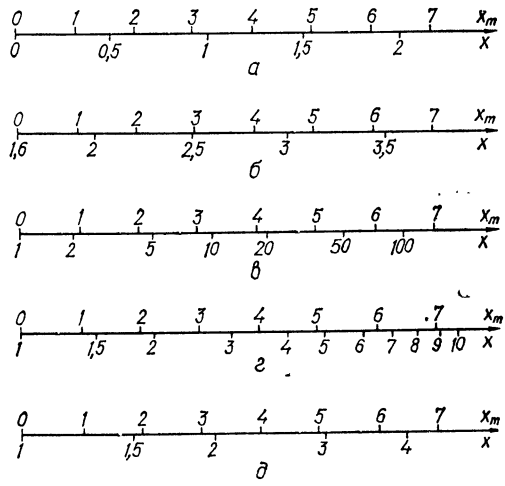


Рис. 24

При выборе линейной шкалы аргумента обычно выбирают равноудаленные точки отсчета $x_{mi} = x_{mi-1} + \Delta x_m = m x_i = m(x_{i-1} + \Delta x)$ с постоянным шагом Δx изменения аргумента. В тех случаях, когда необходимо «сжать» интервал аргумента со стороны больших его значений, обычно используют логарифмическую шкалу с отсчетами

$$x_{mi} = m \log_a x_i,\ x_i = a^{x_{mi}/m},$$

причем началу координат $x_m = 0$ соответствует значение $x = 1$, если интервал аргумента не смещен, или другое начальное значение x_0 при смещении интервала аргумента относительно $x_m = 0$.

При логарифмической шкале изменения аргумента в a раз соответствуют изменениям x_{mi} на величину m и при необходимости выбора k точек отсчета аргумента в каждом таком интервале вычисляют множитель $a^{1/k}$, соответствующий выбору отсчетов

$$x_i = (a^{1/k})^i x_0,\ x_{mi} = m \log_a x_0 + mi/k,$$

где x_0 — начальное значение выбранного интервала изменения аргумента.

Следовательно, изменяя масштабный множитель m, можно обеспечить одинаковое изменение значений x_{mi} при различных основаниях

а логарифма. В частности, при десятичной логарифмической шкале $x_m = m \lg x$ отсчеты

$$x_i = 10^{i/k} x_0, \quad x_{mi} = m \lg x_0 + mi/k$$

и интервал оси абсцисс шириной m соответствует изменению аргумента в 10 раз (декаде). Например, при $k = 3$ множитель $10^{1/3} = 2{,}154435$ и отсчетам x_i, $x_{i+1} = 2{,}154435\, x_i$, $x_{i+2} = 4{,}641589\, x_i$, $x_{i+3} = 10\, x_i$ (приближенно $x_{i+1} \approx 2x_i$, $x_{i+2} \approx 5x_i$) соответствуют отсчеты оси абсцисс x_{mi}, $x_{mi+1} = x_{mi} + m/3$, $x_{mi+2} = x_{mi} + 2m/3$, $x_{mi+3} = x_{mi} + m$ (рис. 24, *в*).

Аналогично при выборе натуральной логарифмической шкалы (рис. 24, *г*)

$$x_i = e^{i/k} x_0, \quad x_{mi} = m \ln x_0 + mi/k,$$

а при выборе логарифмической шкалы с основанием 2 (рис. 24, *д*)

$$x_i = 2^{i/k} x_0, \quad x_{mi} = m \log_2 x_0 + mi/k = m \lg x_0 / \lg 2 + mi/k = m \ln x_0 / \ln 2 + mi/k.$$

При одинаковом значении m одним и тем же значениям x_{mi} при логарифмических шкалах с разными основаниями будут соответствовать различные значения аргумента (рис. 24), но, изменив m, можно получить одинаковые логарифмические шкалы при разных основаниях *а* логарифма.

Шкалу отсчетов функции по оси ординат можно выбрать линейной или нелинейной независимо от шкалы аргумента, но значения y_i и y_{ni} связаны соотношениями, подобными приведенным для аргумента. При использовании микрокалькулятора задача построения графика зависимости вида $y = f(x)$ заключается в многократном выполнении программы вычисления этой функции при монотонном изменении аргумента в выбранном интервале. Оптимизация такой программы сводится к минимизации затрат времени на однократное выполнение программы и вспомогательные операции. Потери времени на вспомогательные операции уменьшаются при повышении уровня автоматизации таких операций и, в первую очередь, при автоматизации вычисления очередного значения аргумента x_i.

Если график строится при изменении аргумента с постоянным шагом, то для автоматизации вычисления x_i следует выбрать регистры памяти M и N для хранения соответственно шага Δx и текущего значения x_i. В этом случае вычисление очередного значения x_i целесообразно организовать после вычисления функции и оператора С/П, составляя на входном языке ЯМК34 программу

ПN ... С/П ИПN ИПМ + БП 00

или на входном языке ЯМК21 программу

РN ... С/П FN ↑ РМ + БП Р0,

где многоточием обозначены операторы для вычисления значения функции y по значению аргумента, хранящемуся в регистре N.

Перед первым пуском такой программы в регистры N и M соответственно заносят начальное значение x_0 и шаг Δx, после чего нажимают клавиши В/О и С/П и отмечают на графике индицируемое значение $y_0 = f(x_0)$. Перед последующими пусками нажимают только клавишу С/П, что приводит к вычислению нового значения $x_i = x_{i+1} + \Delta x$ и функции для значения x_i. Если необходимо вычислить значение функции для произвольного значения x_q аргумента, то его заносят в регистр X и нажимают клавиши В/О и С/П. При последующих пусках программы нажатием клавиши С/П будут вычисляться значения $x_{q+j} = x_q + j\,\Delta x$ и для восстановления исходной системы отсчетов x_i следует ввести в регистр X значение x_i из этой системы и нажать клавиши В/О и С/П, после чего при нажатии клавиши С/П будет вычисляться $x_{i+j} = x_i + j\,\Delta x$.

При построении графика с линейной шкалой аргумента значения $x_{mi} = mx_i$ могут быть вычислены заранее и отмечены на оси абсцисс. Однако в тех случаях, когда для уточнения исследуемой функциональной зависимости приходится вычислять y_q в произвольных точках x_q, целесообразно автоматизировать и вычисление x_{mq} в соответствии, например, с единицами измерения длины абсциссы (например, клеточками бумаги с прямоугольной сеткой) на графике. Для этого достаточно выбрать регистр памяти L для хранения масштабного множителя m и составить программу

ПN ... ИПN ИПL × С/П ИПN ИПМ + БП 00.

Тогда после каждого выполнения программы на индикаторе будет высвечиваться значение x_{mi} и для индикации соответствующего ему значения функции y_i достаточно нажать клавишу XY.

Подобный прием еще больше нужен для автоматизации вычисления y_i, поскольку эти значения заранее неизвестны и приходится каждый раз определять на графике абсциссу y_{ni}, соответствующую y_i. Для этого следует дополнительно занести масштабный множитель n (если он отличается от m) в регистр Q памяти и использовать программу

ПN ... ↑ ИП0 × ИПN ИПL × С/П ИПN ИПМ + БП 00.

В этом случае после каждого выполнения программы значения x_i, x_{mi}, y_{ni} и y_i будут храниться соответственно в регистре N памяти и регистрах X, Y и Z операционного стека. Для вывода значений y_{ni} и y_i достаточно дважды ввести оператор → поворота операционного стека.

При логарифмической шкале аргумента для вычисления микрокалькулятором очередных значений x_i следует ввести в выбранный регистр M памяти множитель $a^{1/k}$ и использовать программу со структурой

ПN ... С/П ИПN ИПМ × БП 00.

Пользование такой программой аналогично рассмотренному выше, но для вычисления значения функции в произвольной точке x_q необходимо ввести это значение в регистр X и нажать клавиши В/О

и С/П (значение $x_{mq} = m \log_a x_q$ придется вычислить в обычном режиме). В этом случае при последующих пусках программы нажатиями только клавиши С/П будут вычисляться значения функции при значениях аргумента $x_j = a^{j/k} x$.

Если желательно при логарифмической шкале аргумента задаваться его значениями с равномерным шагом, то при занесении масштабного множителя m в некоторый регистр Q памяти можно использовать программу со следующей структурой:

ПN ... ИПN log ИПQ × С/П ИПN ИПМ + БП 00.

После выполнения такой программы будет высвечиваться значение $x_{mi} = m \log_a x_i$ при $x_i = x_{i-1} + \Delta x$ и для индикации соответствующего значения функции достаточно нажать клавишу XY.

Может оказаться более удобным задаваться значениями x_{mi} с равномерным шагом Δx_m. В этом случае, выбрав регистры памяти M, L и Q для хранения соответственно значений Δx_m, x_{mi} и m и занося в регистры N и X значение x_i перед первым пуском программы, целесообразно составить ее со следующей структурой:

ИПQ × a^x ПN ... С/П ИПL ИПМ + ПL БП 00,

где, в зависимости от основания логарифмов, выбирают вместо a^x оператор 10^x или e^x. Если основание логарифма отличается от 10 и e, то оператор a^x следует заменить фрагментом a X^y, где a — оператор набора основания логарифма.

Приведенные фрагменты несложно перевести и на другие входные языки, в частности, на входной язык ЯМК21. В последнем случае трудности могут возникнуть лишь в связи с отсутствием оператора lg при построении графиков с десятичными логарифмическими шкалами. Однако в этом случае оператор lg можно перевести фрагментом ln 0 , 4 3 4 3 × или, при занесении числа 0,4343 в некоторый регистр N памяти, фрагментом ln ↑ FN ×.

Рассмотренные выше приемы можно использовать и для вычисления в требуемом масштабе значений $y_{mi} = N(y - y_0)$. В частности, при вычислении отношения мощностей с размерностью неперов y_i(Неп) = $\ln y$ или децибелов y_i(дБ) = $10 \lg y$ на микрокалькульторе с входным языком ЯМК21 последние удобно вычислять по формуле y_i(дБ) = $4{,}343 \ln y_i$.

При построении графиков может оказаться целесообразной и автоматизация пересчета единиц измерений. Например, если график зависимости температуры от времени желательно представить относительно различных единиц измерения температуры, то при вычислении температуры в градусах Цельсия для пересчета в кельвины достаточно добавлять 270, а для пересчета в градусы Реомюра, практически не используемые в настоящее время, придется кроме изменения начала отсчета вводить нормирующий множитель.

Вычисление на микрокалькуляторе результата с большим числом значащих цифр при построении графиков приводит к дополнительным потерям времени для округления результата до числа цифр, соответствующих практически возможной точности нанесения точек на гра-

фик, или попыток исполнителя нанести эти точки с практически недостижимой точностью. Поэтому в некоторых микрокалькуляторах предусмотрен оператор FIX, обеспечивающий вывод результата с заданным числом значащих цифр. В программах на входных языках, не содержащих подобного оператора, он может быть реализован фрагментом, обеспечивающим округление результата до требуемого числа значащих цифр.

Построение таких фрагментов зависит от выбранного способа сокращения количества значащих цифр результата вычислений a, а также от формы представления чисел, разрядности r мантиссы и способа округления чисел в используемом микрокалькуляторе.

Округление a до a_1 с числом k значащих цифр можно выполнить, используя смещение «избыточных» младших разрядов мантиссы в область относительного машинного нуля с последующим восстановлением ее порядка согласно следующему алгоритму:

1. Вычислить $b = a\ 10^s$, где $s = r - k$.
2. Вычислить $c = b + a$.
3. Вычислить $a_1 = c - b$.

Этот алгоритм реализуется на входном языке ЯМК21 фрагментом

↑ ВП S XY + YY —

и на входном языке ЯМК34 — фрагментом

↑ ВП S + RX — .

Результат выполнения рассматриваемого алгоритма зависит от округления чисел микрокалькулятором. При округлении отбрасыванием сохраняемые цифры числа a не округляются и, например, при $s = 8 - 2 = 6$ получим $4490 \approx 4400$; $4445 \approx 4400$; $0{,}0999 \approx \approx 0{,}099$; $2{,}2222 \cdot 10^9 \approx 2{,}2 \cdot 10^9$. При симметричном или близком к нему округлении (например, характерном для первых выпусков микрокалькуляторов с входными языками ЯМК21 и ЯМК34) при $s = 6$ получим $4490 \approx 4500$; $4445 \approx 4500$; $0{,}0999 \approx 0{,}1$; $2{,}2222 \cdot 10^9 \approx \approx 2{,}2 \cdot 10^9$. Округление до одной значащей цифры чисел с мантиссой, большей 9,999999 (восьмая цифра в таких числах больше нуля), по рассматриваемому алгоритму на микрокалькуляторах с входным языком ЯМК21, вычисляющих $10 - 10^{-20} = 9{,}9999999$, приводит к ошибочному результату, равному нулю.

В микрокалькуляторах парных выпусков с входным языком ЯМК21 при последовательном симметричном округлении, рассмотренном в гл. 2, результат округления некоторых чисел по рассмотренному алгоритму зависит от цифры в младшем разряде, например, $44{,}4444444 \approx 44$, но $44{,}4444445 \approx 45$. Для проверки способа округления чисел в микрокалькуляторе достаточно выполнить фрагмент

5 , 5 ↑ 1 ВП 7 XY + XY — .

Если вычисляется число 6, то микрокалькулятор последовательно округляет по дополнению, если же вычисляется 5, то микрокалькулятор выполняет округление отбрасыванием. В первом случае можно реализовать симметричное округление, вычитая из округляемого

числа $(1/18)10^k$, во втором — увеличив его предварительно на величину $0{,}5 \cdot 10^k$.

Часто желательно округлять результаты вычислений до определенного числа значащих цифр после запятой. Округление числа a до a_2 с $k_д$ цифрами после запятой на микрокалькуляторе с r разрядами мантиссы можно выполнить согласно следующему алгоритму:

1. Вычислить $b = a + 10^q$, где $q = r - 1 - k_д$.
2. Вычислить $a_2 = b - 10^q$.

Этот алгоритм реализуется на входном языке ЯМК21 фрагментом

↑ 1 ВП q XY + XY —

и на входном языке ЯМК34 — фрагментом

↑ 1 ВП q + ВХ — ,

где оператор ↑ вводят, если округляемое число a набирают в регистр X, а не вызывают из памяти или предварительно вычисляют.

Рассмотренный алгоритм обеспечивает требуемое округление чисел, если порядок p мантиссы не превышает q, так как при r разрядах мантиссы ее дробная часть содержит не более $r - 1 - p$ цифр. Результат вычислений по этому алгоритму также зависит от способа округления чисел в микрокалькуляторе. Для микрокалькуляторов с округлением отбрасыванием сохраняемые цифры не округляются, для микрокалькуляторов первых выпусков с последовательным симметричным округлением результата суммирования при $k_д = 1$ ($q = 8 - 1 - 1 = 6$) получим $4{,}4444 \approx 4{,}4$; $4{,}4445 \approx 4{,}5$; $0{,}99 \approx 1$; $0{,}099 \approx 0{,}1$; $1{,}2345 \approx 1{,}2$ $12{,}345 \approx 12{,}4$; $123{,}45 \approx 123{,}5$; $1{,}2345 \times \times 10^{-1} \approx 0{,}1$; $1{,}2345 \cdot 10^{-2} \approx 0$.

Частным случаем округления до k значащих цифр при $k_д = 0$ является выделение целой части числа. Однако в этом случае обычно требуется отбрасывание дробной части числа без округления целой части. На микрокалькуляторе любого выпуска с входным языком ЯМК34 (для ЯМК21 см. программу 193/21 в приложении) это требование удовлетворяется при использовании фрагмента

ПN КИПN ИПN ,

если содержимое регистра N не менее единицы,
где $N > 6$, так как в противном случае при выполнении оператора КИП содержимое регистра N будет изменяться.

С помощью косвенной адресации удобно выделять дробную часть числа, а также разделять целую и дробную части в процессе выполнения программы, содержащей например фрагмент

ПМ ПN КИПN ИПМ ИПN — ИПN

при $N > 6$. После выполнения такого фрагмента целая часть исходного содержимого регистра X хранится в регистре N и высвечивается на индикаторе, а исходное число и его дробная часть хранятся соответственно в регистрах M и Y. Например, после выполнения фрагмента П7 П8 КИП7 ИП8 ИП7 — ИП7 для числа $x = 123{,}456$ получим $PX = P7 = 123$, $PY = 0{,}456$; $P8 = 123{,}456$.

Следует помнить, что числа с нулевой целой частью при выполнении оператора КИПN не модифицируются и, например, для числа $x = 0{,}456$ после выполнения рассмотренного фрагмента получим $PX = P7 = P8 = 0{,}456$; $PY = 0$.

Рассмотрим примеры практического применения описанных алгоритмов для построения графиков.

График абсолютной погрешности $\Delta\varphi°$ аппроксимационной формулы

$$\varphi° = \operatorname{arctg} x \approx 90x/\sqrt{1{,}2 + x^2 + \sqrt{1{,}62\,(1 + x^2)}} \qquad (4.2)$$

нельзя построить с помощью программы на входном языке ЯМК21 в связи с отсутствием оператора arctg в словарном запасе этого языка. Поэтому в качестве аргумента выберем не величину x, а значение $\varphi° = \operatorname{arctg} x$, что и удобнее, так

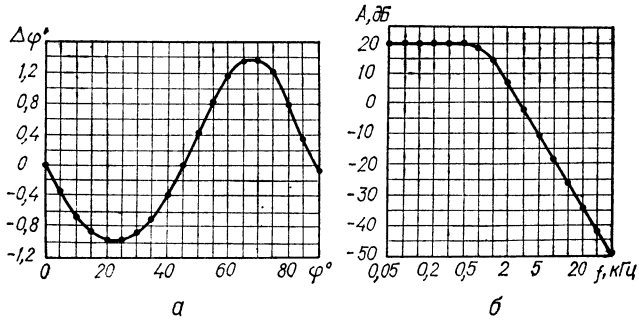


Рис. 25

как интервал x бесконечен, тогда как для оценки погрешности $\varphi°$ достаточно ограничиться интервалом [0, 90°]. В этом случае

$$\Delta\varphi\,(\varphi°) = 90 \operatorname{tg}\varphi/\sqrt{1{,}2 + \operatorname{tg}^2\varphi + \sqrt{1{,}62\,(1 + \operatorname{tg}^2\varphi)}} - \varphi.$$

Разместив исходные данные $\varphi° = PX$, $\varepsilon\varphi = P4$, $n = P6$, $\pi/180 = P7$; $1{,}62 = P8$ и выбрав регистр *3* для хранения промежуточного результата $x = \operatorname{tg}\varphi$, составим следующую программу для вычисления погрешности $\Delta\varphi°$ с округлением результата до двух цифр после запятой в точках абсциссы $\varphi°_i$ с шагом $\varepsilon\rho$:

Р2	↑	F7	×	$e^{|x|}$	÷	Р3	x^2	1	+
↑	F8	×	$\sqrt{}$	+	5	$1/x$	+	$\sqrt{}$	↑
F3	XY	÷	9	0	×	↑	F2	—	↑
F6	×	1	ВП	5	XY	+	XY	—	С/П
F2	↑	F4	+	БП	Р0				

Начальная часть этой программы (адреса шагов с 00 до 41) предназначена для вычисления $\varphi°_i$, операторы ↑ F2 обеспечивают вычисление искомой погрешности $\Delta\varphi°$, следующий фрагмент, оканчивающийся оператором С/П, предназначен для масштаба результата и его округления до двух цифр после запятой, а заключительный фрагмент обеспечивает вычисление следующего значения $\varphi°_i$.

Приняв предварительно $n = 1$, убеждаемся после контрольных проверок, что значения $\Delta\varphi°$ не превышают $0{,}02° \approx 1{,}2'$. Выражая $\Delta\varphi$ в угловых минутах и приняв для оси ординат $\Delta\varphi$ масштаб, при котором одной минуте соответствуют 5 единиц длины оси ординат, вычисляем $n = 60 \cdot 5 = 300$. Введя исходные данные $\varepsilon\varphi = 5 = P4$, $n = 300 = P6$, $\pi/180 = P7$; $1{,}62 = P8$, $\varphi_0 = 0 = PX$ и пуская программу нажатием клавиш В/О и С/П, отмечаем на графике значение $\Delta\varphi\,(0) = 0$. После этого достаточно нажимать клавишу С/П и отмечать на графике точки, соответствующие значениям $\Delta\varphi$ в выбранном масштабе (рис. 25, *а*).

Построение графиков в логарифмическом масштабе рассмотрим на примере вычисления частотной характеристики передачи мощности в децибелах $A(\omega)$ дБ $= 20 \lg |K(j\omega)|$ через электрическую цепь с коэффициентом передачи напряжения $K(j\omega) = K_0/((1-\alpha\omega^2)^2 + \beta^2\omega^2)^{1/2}$, где α и β — постоянные.

При использовании входного языка ЯМК34 в соответствии с ранее приведенными формулами $\omega = 10^{x m/m}$ и при максимальной длине x_{max} оси абсцисс на графике и заданном диапазоне частот $[\omega_н, \omega_в]$ масштабный множитель аргумента $m = (1/x_{max}) \lg(\omega_в/\omega_н)$. Масштабный множитель n результата вычисленнй целесообразно выбирать после контрольных выполнений программы или по аналитической оценке предельных значений вычисляемой функции.

Для удобства построения графика вычисляемые координаты точек оси ординат целесообразно округлять до двух цифр после запятой. Кроме того, целесообразно автоматизировать вычисление значений аргумента по логарифмическому закону при постоянном шаге на оси аргумента.

Программа 10/34. Вычисление частотной характеристики передачи электрической мощности

П8	ИП2	÷	10^x	ИП1	×	П9	x^2	ИПА ×
1	—	x^2	ИП9	ИПВ	×	x^2	+	ИП0 XY
÷	lg	2	0	×	ИП3	×	1	ВП 5
+	Вх	—	С/П	ИП8	ИПД	+	БП	00

Инструкция: ($K_0^2 = P0$, $\alpha = PA$, $\beta = PB$, $\omega_н = P1$, $m = P2$, $n = P3$, $\Delta x_m = PД$) $x_{m0} = PX$ В/О С/П $PX = A(\omega_н)$ дБ С/П $PX = A(\omega_1)$ дБ ... С/П $PX = A(\omega_2)$ дБ, $P8 = x_{mi}$, $P9 = \omega_i$.

По исходным данным, например, $K_0 = 10$, $\alpha = 0{,}03$; $\beta = 0{,}2$, приняв предварительно $\omega_н = n = m = 1$ и выполнив несколько раз программу при различных значениях x_m, находим, что зависимость $A(\omega)$ целесообразно строить для интервала $-0{,}5 \leqslant x_m \leqslant 2{,}5$, которому соответствует диапазон частот от 0,316 рад/мс (или $f \approx 0{,}05$ кГц) до 316 рад/мс ($f \approx 50$ кГц), причем в этом диапазоне A изменяется от 20 до —50 дБ.

Приняв пределы изменения длины абсциссы от 0 до 15 единиц длины с шагом $\Delta x_m = 1$ (что соответствует шести точкам на декаду изменения частоты) и диапазон частот от 0,05 до 50 кГц, определим $\omega_н = 2\pi\, 0{,}05 = 0{,}31415926$; $1/m = (1/15) \lg 1000 = 0{,}2$. Приняв масштаб A на оси ординат в 10 дБ на единицу длины, получим $n = 0{,}2$. Результаты выполнения программы 10/34 по этим исходным данным (0 = PX В/О С/П $PX = 4$ С/П $PX = 4{,}01$ С/П $PX = 4{,}02$...) наносим на график, показанный на рис. 25, *б*.

Приведенную программу можно дополнить фрагментом перевода круговой частоты ω в линейную f с округлением результата до требуемого числа значащих цифр, но в этом случае увеличатся затраты времени на выполнение программы. Поэтому достаточно определить несколько значений ω в удобных для отсчета точках оси абсцисс и вычислить для этих точек $f = \omega/2\pi$.

Для вычисления передачи мощности в децибелах на входном языке ЯМК21 следует использовать соотношение $\lg x = 0{,}4342944 \ln x$.

3. ОПЕРАЦИИ НАД КОМПЛЕКСНЫМИ ЧИСЛАМИ И ВЕКТОРАМИ

В различных инженерных приложениях для отображения двухмерных векторов используются комплексные числа. Комплексное число A можно представить в алгебраической, тригонометрической и показательной формах

$$A = \mathrm{Re}\, A + \mathrm{j}\, \mathrm{Im}\, A = |A| e^{\mathrm{j}\varphi_A} = e^{\ln|A| + \mathrm{j}\varphi_A},$$

где $\mathrm{Re}\,A$ и $\mathrm{Im}\,A$ — вещественная и мнимая составляющие; $|A|$ и φ_A — модуль и аргумент (фазовый угол) комплексного числа, отображающего вектор на комплексной плоскости (рис. 26, *а*).

Преобразование комплексного числа из тригонометрической формы в алгебраическую (соответствующее преобразованию вектора из полярной системы координат в прямоугольную) описывается формулами

$$\mathrm{Re\,A} = |A| \cos \varphi_A, \quad \mathrm{Im}\,A = |\mathrm{A}| \sin \varphi_A.$$

При обратном преобразовании модуль вычисляют по формуле $|A| = \sqrt{(\mathrm{Re}\,A)^2 + (\mathrm{Im}\,A)^2}$, а для определения аргумента комплексного числа в интервале $]-\pi, \pi]$ или $]-180, 180°]$ можно воспользоваться соотношениями

$$\varphi_A = \begin{cases} \varphi = \mathrm{arctg}(\mathrm{Im}\,A/\mathrm{Re}\,A) & \text{при } \mathrm{Re}\,A \geqslant 0; \\ \varphi + \pi & \text{при } \mathrm{Re}\,A < 0,\ \mathrm{Im}\,A \geqslant 0; \\ -(\varphi + \pi) & \text{при } \mathrm{Re}\,A < 0,\ \mathrm{Im}\,A < 0. \end{cases}$$

Для программной реализации этих соотношений требуется три условных оператора, но ее можно упростить, воспользовавшись соотношениями

$$\varphi_A = \begin{cases} \varphi_c = \arccos(\mathrm{Re}\,A/|A|) & \text{при } \mathrm{Im}\,A \geqslant 0; \\ -\varphi_c & \text{при } \mathrm{Im}\,A < 0, \end{cases}$$

для реализации которых требуется только один условный оператор. Например, при размещении исходных данных и результатов вычислений $\mathrm{Re}\,A = PA$, $\mathrm{Im}\,A = PB$, $PC = |A|$, $PД = \varphi_A$ преобразование комплексного числа из алгебраической формы в тригонометрическую на входном языке ЯМК34 реализуется фрагментом

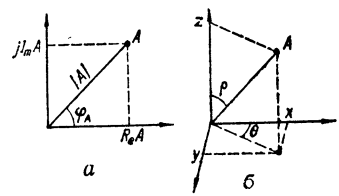


Рис. 26

```
ИПА ↑  x²  ИПВ x²  +  √  ПС  ÷
ПД  ИПВ x<0 A  ИПД /—/ ПД ...,
```

где A — адрес первого оператора следующей части программы, обозначенной многоточием.

Размерность фазового угла, вычисленного по этому фрагменту, зависит от положения переключателя Р—Г. Если необходимо определять фазовый угол в интервале $]\,0, 2\pi]$ или $]\,0, 360°]$, то достаточно использовать соотношения

$$\varphi_A = \begin{cases} \varphi_c = \arccos(\mathrm{Re}\,A/|A|) & \text{при } \mathrm{Im}\,A \geqslant 0; \\ 2\pi - \varphi_c & \text{при } \mathrm{Im}\,A < 0, \end{cases}$$

также реализуемое с помощью только одного условного оператора.

При расчетах на микрокалькуляторе с входным языком ЯМК21, не содержащем операторов вычисления обратных тригонометрических

функций, для их программной реализации можно использовать различные алгоритмы, [11], но они либо не обеспечивают достаточную точность результата, либо реализуются со значительными затратами времени. Поэтому при необходимости получения точного результата с небольшими затратами времени при вычислении функции arctg x следует использовать аппроксимирующую формулу

$$\varphi^0 = \operatorname{arctg} x = 90x \Big/ \sqrt{\alpha + x^2 + \sqrt{\beta(\gamma + x^2 + \sqrt{\rho(1 + x^2)})}} \quad (4.3)$$

при $\alpha = 0{,}8114228$; $\beta = 1{,}6211707$; $\gamma = 1{,}056546$; $\rho = 0{,}403225$.

Методическая погрешность этой формулы менее 0,03″ при любом значении аргумента, но влияние операционных погрешностей при вычислениях на микрокалькуляторе с разрядностью мантиссы $r = 8$ приводит к увеличению предельной погрешности аппроксимации до 0,05″. Для вычисления $\varphi = \operatorname{arc\,tg} x$ в радианах достаточно в формуле (4.3) заменить множитель 90 множителем $\pi/2$ — в этом случае погрешность аппроксимации не превышает $2{,}5 \cdot 10^{-7}$ рад.

Для вычисления с максимальной точностью функций $\arcsin z$ и $\arccos z$ достаточно в формуле (4.3) использовать соответственно подстановки $x = z/\sqrt{1 - z^2}$ или $x = \sqrt{1 - z^2}/z$. Полная погрешность результата при вычислениях на микрокалькуляторе с входным языком ЯМК21 в этом случае не превышает 0,07″.

Составление программы преобразования комплексного числа из алгебраической формы в тригонометрическую (или преобразования системы координат из прямоугольной в полярную) на входном языке ЯМК21 с максимальной точностью требует предварительного преобразования расчетных соотношений. Один из возможных вариантов алгоритма (реализованного программой 142/21 в Приложении) представим следующим описанием:

1. Принять $\alpha' = 1 - \alpha = 0{,}1885772$; $\beta' = \beta(\gamma - 1) = 0{,}091670712$; $\gamma' = 1/(\gamma - 1) = 17{,}684716$; $\rho' = \rho/(\gamma - 1)^2 = 126{,}10829$.
2. Вычислить $|A|^2 = \mathrm{Re}^2 A + \mathrm{Im}^2 A$.
3. Вычислить $q = (\mathrm{Im}\, A/|A|)^2$.
4. Вычислить $S = \sqrt{|A|^2(1 - \alpha' q + \sqrt{\beta' q(\gamma' + q + \sqrt{\rho' q})})}$.
5. Вычислить $\xi = (S - \mathrm{Re}\, A)/S$.
6. Если $\mathrm{Im}\, A \geqslant 0$; то $\varphi = 90\ \xi$, иначе $\varphi = -90\ \xi$.

Точность аппроксимации по формуле (4.3) для большинства практических приложений (за исключением, например, геодезии или навигации) избыточна и вычисления можно упростить, приняв в этой формуле $\rho = 0$, $\alpha = 1{,}2117$; $\beta = 1{,}622281$; $\gamma = 0{,}971533$, что соответствует предельной погрешности аппроксимации 2,2″. Если допустимо вычисление функции с меньшей точностью, то затраты времени на вычисления можно сократить, воспользовавшись аппроксимационной формулой (4.2) с предельной погрешностью, меньшей 0,03° (см. рис. 24, *а*). Вычисления по этой формуле реализованы в программе 141/21 приложения.

Арифметические операции над алгебраическими представлениями комплексных чисел A и B выполняют по следующим формулам:

$$C = A \pm B = \mathrm{Re}\,A \pm \mathrm{Re}\,B + \mathrm{j}\,(\mathrm{Im}\,A \pm \mathrm{Im}\,B);$$

$$C = A \times B = (\mathrm{Re}\,A\,\mathrm{Re}\,B - \mathrm{Im}\,A\,\mathrm{Im}\,B) + \mathrm{j}\,(\mathrm{Re}\,B\,\mathrm{Im}\,A + \mathrm{Re}\,A\,\mathrm{Im}\,B);$$

$$C = A/B = \frac{\mathrm{Re}\,A\,\mathrm{Re}\,B + \mathrm{Im}\,A\,\mathrm{Im}\,B}{\mathrm{Re}^2\,B + \mathrm{Im}^2\,B} + \mathrm{j}\,\frac{\mathrm{Re}\,B\,\mathrm{Im}\,A - \mathrm{Re}\,A\,\mathrm{Im}\,B}{\mathrm{Re}^2\,B + \mathrm{Im}^2\,B}.$$

Те же операции над тригонометрическими представлениями комплексных чисел выполняют по формулам

$$|A \pm B| = \sqrt{|A|^2 + |B|^2 \pm 2|A||B|\cos(\varphi_A - \varphi_B)};$$

$$\varphi_{A\pm B} = \mathrm{arctg}\,((|A|\sin\varphi_A \pm |B|\sin\varphi_B)/(|A|\cos\varphi_A \pm |B|\cos\varphi_B)) =$$

$$= \arcsin\,((|A|\sin\varphi_A \pm |B|\sin\varphi_B)/|A \pm B| = \arccos\,((|A|\cos\varphi_A \pm$$

$$\pm |B|\cos\varphi_B)/|A \pm B|),$$

$$|A \times B| = |A| \times |B|; \quad \varphi_{A\times B} = \varphi_A + \varphi_B,$$

$$|A/B| = |A|/|B|; \quad \varphi_{A/B} = \varphi_A - \varphi_B.$$

При решении задач с комплексными числами требуется программа, обеспечивающая выполнение требуемых операций в произвольной последовательности. Такую программу можно составить из программ выполнения отдельных операций, разместив их в программной памяти таким образом, чтобы способ вызова каждой программы легко запоминался и не требовалось тратить время на обращение к инструкции.

Операторы входного языка микрокалькулятора переводятся (транслируются) на внутренний язык с помощью микропрограмм, хранимых в ПЗУ и вызываемых при вводе соответствующих операторов. Для программы, составленной из нескольких отдельных программ, команды, по которым вызываются нужные программы, можно рассматривать как операторы языка высшего уровня (языка управления заданиями), транслируемые программами на входной язык, каждый оператор которого, в свою очередь, транслируется микропрограммами на внутренний язык.

Словарный запас такого языка высшего уровня содержит, кроме команд вызова нужных программ, используемые операторы входного языка (например, операторы набора операндов и обращения к памяти). Синтаксические правила языка высшего уровня определяются инструкцией по использованию составной программы и правилами ввода используемых операторов входного языка.

В простейшем случае операторы, транслируемые отдельными программами, можно сформулировать из команд В/О С/П для вызова первой программы и команд БП A_i С/П для вызова остальных программ, начальные операторы которых занесены в программной памяти по адресам A_i. Однако такие операторы высшего уровня трудно запоминаются, и более удобно использовать операторы высшего уровня, содержащие операторы входного языка, соответствующие требуемой операции. Примером может служить составная программа 138/34 в приложении, в которой операторы высшего уровня для вычисления

гиперболических и обратных гиперболических функций содержат соответствующие операторы вычисления тригонометрических и обратных тригонометрических функций.

Для арифметических операций над комплексными числами удобно использовать операторы высшего уровня, содержащие арифметические операторы входного языка. Для микрокалькуляторов с входным языком ЯМК21 эта задача решается при таком размещении программ, когда указатели адресов их начальных операторов совпадают с нужными арифметическими операторами. В этом случае для вызова требуемой программы достаточно ввести оператор высшего уровня БП О С/П , где О — нужный арифметический оператор входного языка (см программу 143/21 в приложении).

На входном языке ЯМК34 этот прием непосредственно неприменим и приходится использовать другие способы составления удобных операторов высшего уровня. Один из таких способов заключается в подборе чисел m и n, результат арифметической операции над которыми равен адресу начального оператора нужной программы. В этом случае вызов нужной программы обеспечивается оператором косвенного-безусловного перехода. Достаточную гибкость в распределении программной памяти между отдельными программами обеспечивает вычисление адреса по выражению в обратной записи $m \uparrow n$ О m + (A), где О — арифметический оператор, реализуемый над комплексными числами программой с адресом A. Например, приняв $m = 16$, $n = 3$, получим

$$A\,(+) = 16 + 3 + 16 = 35; \quad A\,(-) = 16 - 3 + 16 = 29;$$
$$A\,(\times) = 16 \times 3 + 16 = 64; \quad A\,(\div) = 16 \div 3 + 16 = 21.$$

Оценим длину фрагментов, реализующих арифметические операции над комплексными числами. Сложение и вычитание реализуется фрагментом с восемью операторами

ИПА ИПС ± ИПВ ИПД ИПД ± БП A_k,

где A_k— адрес перехода к оператору остановки, при занесении операндов $\mathrm{Re}A = PA$, $\mathrm{Im}A = PB$, $\mathrm{Re}B = PC$, $\mathrm{Im}B = PД$. Умножение реализуется фрагментом с 16 операторами

ИПА ИПС × ИПВ ИПД × — ИПА ИПД × ИПВ ИПС ×
× + БП А

Деление целесообразно реализовать как умножение комплексного операнда A на операнд, обратный комплексному числу B. Обращение операнда B реализуется фрагментом из 15 операторов

ИПА ↑ x^2 ИПВ x^2 + ÷ ПА ИПВ /—/ Вх ÷ ПВ БП $A(\times)$

с последующим переходом к программе умножения по адресу A (×).

Возведение в квадрат комплексного числа B реализуется программой умножения после фрагмента перезаписи ИПС ПА ИПД ПВ БП А (×). Преобразование тригонометрического представления операнда B в алгебраическую форму выполняется фрагментом из 12 операторов

ИПД cos ИПС × ПС Вх ИПД sin × ПД БП $A_ф$

с адресом $A_ф$ начального фрагмента формирования адресов (в этом случае оператор высшего уровня содержит оператор С/П).

Для составления полной программы целесообразно предварительно оценить возможность размещения отдельных программ с помощью табл. 9. Первые 16 шагов используем для начального фрагмента (при предварительном занесении операндов A и B в операционный стек)

ПД → ПС → ПВ → ПА → 6 П9
3 С/П ИП9 + П9 КБП9 ...

Остальные фрагменты размещаются в имеющихся «зазорах» с разбиением на части и использованием операторов НОП для заполнения «пустых» шагов.

Более удобную программу можно составить при использовании «нештатных» операторов входного языка ЯМК34, в котором при нажатии клавиш с символами +, —, ×, ÷ и XY после операторов обращения к памяти выполняется обращение соответственно к регистрам памяти с номерами 0, 1, 2, 3 и 4. Занося предварительно адреса нужных программ в эти регистры, получим легко запоминающиеся операторы высшего уровня (идентификаторы), соответствующие следующей составной программе.

9. Данные для распределения в памяти фрагментов программы

Адрес	Оператор	Операторы ЯМК34	Разность адресов	Длина реализующего фрагмента
16	$1/x$	$1/x$ С/П	3	4+16
19	Т → А	С/П	2	12
21	÷	÷ С/П	4	4
25	x^2	x^2 С/П	4	6
29	—	— С/П	6	8
35	+	+ С/П	29	8
64	×	× С/П	32	16
		Итого	80	74

Программа 11/34. Выполнение операций над комплексными числами

ПД	КБПД	ИП5	ИП6	cos	×	ИП5	ИП6	sin	×
П6	XY	П5	С/П	БП	00	ИП8	x^2	ИП9	x^2
+	$\sqrt{-}$	П7	ИП8	ИП7	÷	arccos	ПС	ИП9	$x<0$
34	ИПС	/—/	ПС	ИПС	ИП7	БП	13	ИП9	ИП6
+	П9	ИП8	ИП5	+	П8	БП	13	ИП9	ИП6
—	П9	ИП8	ИП5	—	П8	БП	13	ИП8	ИП5
×	ИП9	ИП6	×	—	ИП8	ИП6	×	ИП9	ИП5
×	+	П9	XY	П8	БП	13	ИП5	x^2	ИП6
x^2	+	ПВ	ИП6	/—/	П6	ИПВ	÷	П6	ИП5
ИПВ	÷	П5	БП	58					

Инструкция: (38 = *P0*, 48 = *P1*, 58 = *P2*, 77 = *P3*, 16 = *P4*, 2 = *PA*), ReA = *P8*, ImA = *P9*, ReB = *P5*, ImB = *P6*; для сложения ввести ИП + С/П, (при первом пуске программы вместо С/П ввести В/О С/П), для вычитания ввести ИП—С/П, для умножения ввести ИП × С/П, для деления ввести ИП ÷ С/П; результат PX = *P8* =

$= \mathrm{Re}C$, $РУ = P9 = \mathrm{Im}\ C$; для преобразования из тригонометрической формы в алгебраическую $|A| = P5$, $\varphi_A = P6$ ввести ИП А С/П, результат $РХ = P5 = \mathrm{Re}A$, $РУ = P6 = \mathrm{Im}A$; для преобразования в тригонометрическую форму $\mathrm{Re}A = P8$, $\mathrm{Im}A = P9$ ввести ИП XY С/П, результат $РХ = P7 = |A|$, $РУ = РС = \varphi_A$ (в интервале от $-\pi$ до π в градусах или радианах в зависимости от положения переключателя Р—Г).

Эта программа удобна для выполнения цепных операций, так как результат арифметической операции заносится на место первого операнда и достаточно ввести следующий операнд в регистры *5* и *6*. Для временного хранения операнда (например, при выполнении операций вида $A \times B + C \times E$) можно использовать регистры *7* и *C* (если не выполняется преобразование в тригонометрическую форму).

Контрольный пример: $(-2 + j2)(2 - j3) : (3 + j4) + (6 + j6) - 5e^{j45°} = 4{,}3444659 + j3{,}3444659 = 5{,}4510438\ e^{j37{,}846346°}$.

Могут быть использованы и другие способы формирования операторов высшего уровня (идентификаторов программ) для составных программ, предназначенных для выполнения различных операций, не предусмотренных входным языком микрокалькулятора, в произвольной последовательности.

В качестве примера рассмотрим программу для выполнения основных операций над векторами в трехмерном эвклидовом пространстве. Преобразование вектора из сферической системы координат в прямоугольную (см. рис. 26,*б*) описывается формулами

$$x = \rho \sin\theta \cos\varphi,\ y = \rho \sin\theta \sin\varphi,\ z = \rho \cos\theta\ .$$

При обратном преобразовании модуль вектора определяется по его проекциям на координатные оси согласно формуле

$$\rho = \sqrt{x^2 + y^2 + z^2},$$

а угловые координаты в интервалах $]\ -\pi\ ,\ \pi\ ]$ — по соотношениям

$$\varphi = \begin{cases} \varphi_0 = \mathrm{arctg}\,(y/x) & \text{при } x \geqslant 0; \\ \varphi_0 + \pi & \text{при } x < 0,\ y \geqslant 0; \\ -(\varphi_0 + \pi) & \text{при } x < 0,\ y < 0 \end{cases}$$

и

$$\theta = \begin{cases} \theta_0 = \mathrm{arctg}\,(\sqrt{x^2 + y^2}/z) & \text{при } z \geqslant 0; \\ \theta_0 + \pi & \text{при } z < 0. \end{cases}$$

Для упрощения программной реализации эти соотношения, как и при преобразовании двухмерных векторов, отображаемых комплексными числами, удобно заменить формулами

$$\varphi = \begin{cases} \varphi_c = \arccos\,(x/\sqrt{x^2 + y^2} & \text{при } y \geqslant 0; \\ -\varphi_c & \text{при } y < 0; \end{cases}$$

$$\theta = \arccos\,(z/\rho).$$

При размещении исходных данных и результатов вычислений $x = P7$, $y = P8$, $z = P9$, $\rho = P4$, $\varphi = P5$, $\theta = P6$ вычисления по

приведенным формулам для преобразования вектора из сферической системы координат в прямоугольную можно реализовать подпрограммой на языке ЯМК34

```
ИП5 cos ИП4 ИП6 sin  ×  ×  П7 Вх  ИП5
sin  ×  П8  ИП4 ИП6 cos ×  П9 В/О
```

и для обратного преобразования — подпрограммой

```
ИП8 П4    ИП7 ПП  A(↑) П5   ИП8 x<0 A(ИП9) ИП5
/—/ П5    ИП9 ↑   x²    ИП4 x²   +   √‾     П4
÷   arccos П6   В/О
```

с адресами $A(\uparrow)$ и A(ИП9) перехода соответственно к операторам ↑ и ИП9 в той же подпрограмме.

Для сложения векторов, заданных в прямоугольной системе координат (с хранением исходных слагаемых в регистрах *7*, *8*, *9* и *1*, *2*, *3* и занесением результата в регистры *1*, *2*, *3*), используем простую подпрограмму

```
ИП7 ИП1 + П1 ИП8 ИП2 + П2 ИП9 ИП3 + П3 В/О
```

и для векторного произведения, вычисляемого по формулам $x = y_1 \times \times z_2 - z_1 y_2$; $y = z_1 x_2 - x_1 z_2$; $z = x_1 y_2 - y_1 x_2$, при том же размещении исходных данных и результатов — подпрограмму

```
ИП8 ИП3 ×   ИП9 ИП2 ПО ×  —   ИП9 ИП1
×   ИП7 ИП3 ×   —   П2 XY ИП7 ИП0 ×
ИП8 ИП1 ×   —   П3  XY П1 В/О
```

Выполнение требуемой операции организуем по занесению в регистр X двузначного десятичного кода pq перед пуском программы, выбрав для преобразования координат $p = 0$, для сложения $p = 3$, для умножения $p = 5$, для прямоугольной системы координат $q = 7$ и для сферической системы координат $q = 5$. Например, для сложения очередного вектора, заданного в сферической системе координат, с результатом предыдущей операции следует нажать клавиши 3 5С/П.

Подобная система формирования операторов входного языка высшего уровня заставляет несколько изменить ранее составленные подпрограммы и разбить подпрограмму преобразования сферических координат в прямоугольные на две части. С учетом этих замечаний составляемую программу представим в следующем виде.

Программа 12/34. Выполнение последовательности операций над трехмерными векторами в прямоугольной и сферической системах координат

ПС	ИП8	КППС	С/П	КБПД	БП	82	П4	ИП7	ПП
19	П5	ИП8	$x<0$	18	ИП5	/—/	П5	ИП9	↑
x^2	ИП4	x^2	+	$\sqrt{}$	П4	÷	arccos	П6	В/О
ИП5	sin	×	П8	В/О	ПП	82	ИП2	+	П2
ИП7	ИП1	+	ИП3	ИП9	+	П3	П9	XY	П1
П7	ИП2	П8	БП	07	ПП	82	ИП3	×	ИП9
ИП2	П0	×	—	ИП9	ИП1	×	ИП7	ИП3	×
—	П2	XY	ИП7	ИП0	×	ИП8	ИП1	×	—
БП	46	ИП4	ИП6	cos	×	П9	ИП5	cos	ИП4
ИП6	sin	×	×	П7	Вх	БП	30		

Инструкция: 1) установить переключатель Р—Г в положение, соответствующее размерности угловых координат; 2) очистить регистры *1*, *2*, *3*, *Д*; 3) ввести сферические координаты $\rho = P4$, $\varphi = P5$, $\theta = P6$ или прямоугольные координаты $x = P7$, $y = P8$, $z = P9$ первого операнда; 4) ввести В/О 35 С/П или В/О 37 С/П при задании первого операнда в сферических или прямоугольных координатах соответственно; 5) ввести следующий операнд в сферических $\rho = P4$, $\varphi = P5$, $\theta = P6$ или прямоугольных $x = P7$, $y = P8$, $z = P9$ координатах; при необходимости преобразования в сферические или прямоугольные ввести соответственно 05 С/П (время счета около 15 с) или 07 С/П (время счета около 12 с); 6) для сложения операндов в сферической или прямоугольной системе ввести соответственно 35 С/П (время счета около 38 с) или 37 С/П (время счета около 20 с), для векторного умножения операндов в сферической или прямоугольной системе ввести соответственно 55 С/П (время счета около 43 с) или 57 С/П (время счета около 28 с); 7) считать результат в сферических $P4 = \rho$, $P5 = \varphi$, $P6 = \theta$ или прямоугольных $P7 = x$, $P8 = y$, $P9 = z$ координатах; для продолжения вычислений перейти к п. 5.

В качестве примера вычислим вектор $D = A \times [B + C]$ при координатах $a_x = 3$, $a_y = 4$, $a_z = 5$, $b_x = 8$, $b_y = -1$, $b_z = 0$, $\rho_c = 4$, $\varphi_c = -30°$, $\theta_c = 60°$. В соответствии с инструкцией:

1) устанавливаем переключатель Р—Г в положение Г;
2) очищаем регистры $0 = P1 = P2 = P3 = PД$;
3) вводим $b_x = 8 = P7$, $b_y = -1 = P8$, $0 = P9$;
4) выполняем 37 В/О С/П;
5) вводим $\rho_c = 4 = P4$, $\varphi_c = -30 = P5$, $\theta_c = 60 = P6$;
6) выполняем 35 С/П;
7) вводим $a_x = 3 = P7$, $a_y = 4 = P8$, $a_z = 5 = P9$;
8) выполняем 57 С/П;
9) регистрируем $P7 = d_x = 21{,}660255$; $P8 = d_y = 49$; $P9 = d_z = -52{,}196153$ или $P4 = \rho_d = 74{,}797091$; $P5 = \varphi_d = 66{,}152365°$; $P6 = \theta_d = 134{,}25369°$.

Таким образом словарный запас входного языка высокого уровня, реализованного программой 12/34, содержит операторы набора чисел, функциональные операторы 05 С/П, 07 С/П, 35 С/П, 37 С/П, 55 С/П, 57 С/П, а также операторы обращения к регистрам памяти *1*, ..., *9* и *Д*. При использовании программ 11/34 и 12/34 функциональные операторы высокого уровня переводятся (транслируются) на входной

язык ЯМК34, функциональные операторы которого, в свою очередь, транслируются на внутренний язык микропрограммами, хранящимися в ПЗУ микрокалькулятора. Подобная методика составления входных языков высокого уровня может быть использована и при решении других задач с ограниченным числом сложных операций.

4. ОПЕРАЦИИ НАД СТЕПЕННЫМИ МНОГОЧЛЕНАМИ

В инженерных расчетах часто встречающейся функцией является степенной многочлен

$$A(p) = \sum_{i=0}^{n} a_i p^i = a_n p^n + a_{n-1} p^{n-1} + \cdots + a_1 p + a_0 \qquad (4.4)$$

с комплексным в общем случае аргументом $p = x + jy$.

При операциях над степенными многочленами широко используют схему Горнера, соответствующую итерационной формуле

$$b_{n-1-j} = b_{n-j}\xi + a_{n-j},\ j = 0,\ 1,\ \ldots,\ n, \qquad (4.5)$$

где начальное значение $b_n = 0$, результат вычислений b_{n-1-j} на каждой итерации с номером $j = 0,\ 1,\ \ldots,\ n-1$ равен коэффициенту частного $B(p) = A(p)/(p-\xi)$, а последнее значение $b_{-1} = A(\xi)$ является остатком от деления, равным значению исходного многочлена $A(p)$ при $p = \xi$.

Для вычислений только значений многочлена $A(\xi)$ при заданных значениях аргумента $p = \xi$ степенной многочлен (4.4) в соответствии со схемой Горнера представляют расчетным выражением

$$A(\xi) = (\ldots(a_n\xi + a_{n-1})\xi + a_{n-2})\xi + \cdots + a_1)\xi + a_0, \qquad (4.6)$$

обеспечивающим получение результата при минимальном числе операций и, следовательно, минимальной операционной погрешности.

Выбор расчетного соотношения для программной реализации вычисления степенного многочлена зависит от условий оптимальности. В программах для многократного вычисления многочлена при изменении вещественного аргумента $p = x$ для сокращения времени счета целесообразно минимизировать число переходов и использовать непосредственно формулу (4.6). При программировании на входном языке ЯМК34 в этом случае для хранения аргумента следует использовать операционный стек — тогда все 14 регистров памяти могут быть использованы для хранения коэффициентов многочлена.

Программа 13/34. Вычисление многочленов степени $n \leqslant 13$ вещественного аргумента $p = x$ при минимальном времени счета

↑	↑	ИПД	×	ИПС	+	×	ИПВ	+	×
ИПА	+	×	ИП9	+	×	ИП8	+	×	ИП7
+	×	ИП6	+	×	ИП5	+	×	ИП4	+
×	ИП3	+	×	ИП2	+	×	ИП1	+	×
ИП0	+	С/П	БП	00					

Инструкция: (a_0= *P0*, a_1= *P1*, a_2= *P2*, ..., a_9= *P9*, a_{10}= *PA*, a_{11}= *PB*, a_{12}= *PC*, a_{13}= *РД*), x = *PX* В/О С/П *PX* = $A(x)$.

Контрольный пример: при $x = 2$ время вычисления многочлена $A(x) = 13x^{13} + 12x^{12} + 11x^{11} + 10x^{10} + 9x^9 + 8x^8 + 7x^7 + 6x^6 + 5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 0{,}5 = 196610{,}5$ около 14 с.

При степени $n < 13$ вместо отсутствующих коэффициентов в соответствующие регистры памяти следует занести нули, но в этом случае для сокращения времени счета целесообразно изменить начало программы, записывая его, например, при $n = 10$ как

↑ ↑ ИПА × ИП9 + × ИП8 + × ...

Если по условиям задачи требуется вычислять многочлен для одного или нескольких значений аргумента, то для сокращения общего времени решения задачи целесообразно минимизировать длину программы, используя для этого все особенности входного языка, например, косвенную адресацию во входном языке ЯМК34.

Программа 14/34. Вычисление многочленов степени $n \leqslant 12$ вещественного аргумента $p = x$ с минимальным временем ввода программы

↑	↑	↑	0			XY	↬	КИПД	+	×	ИПД
1	—	ПД	$x=0$	05	↬	ИП0		+	С/П	БП	

Инструкция: (a_0= *P0*, a_1= *P1*, a_2= *P2*, ..., a_{10}= *PA*, a_{11}= *PB*, a_{12}= *PC*) n = *РД*, x = *PX* (В/О) С/П *PX* = $A(x)$.

Контрольный пример: при $x = 2$ время вычисления многочлена $A(x) = 12x^{12} + 11x^{11} + 10x^{10}\,9x^9 + 8x^8 + 7x^7 + 6x^6 + 5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 0{,}5 = 90114{,}5$ около 40 с. Для сравнения укажем, что это значение вычисляется по программе 13/34, модифицированной для $n = 12$, всего за 13 с.

Для вычисления многочленов степени $n = 14$ достаточно их нормировать делением на коэффициент a_{14} и использовать программу 13/34, изменив ее начало следующим образом:

↑ ↑ ↑ 1 × ИПД + × ...

Можно повысить максимальную степень вычисляемого многочлена до $n = 16$, размещая коэффициенты при старших степенях в операционном стеке [13], но их придется вводить перед каждым пуском программы, что увеличит время решения задачи. Вычисления многочленов степени $n > 13$ можно реализовать непосредственно по итерационной формуле (2.14) с вводом очередного коэффициента перед каждым пуском программы 124/34 в приложении, но в этих случаях часто оказывается более удобным представлять многочлены высокой степени разложением

$$A(x) = (\cdots + a_{m+2}x + a_{m+1})\,x^{m+1} + (a_m x^m + \cdots + a_1 x + a_0) \quad (4.7)$$

с вычислением каждого выражения в скобках по приведенным выше программам.

Для вычисления степенных многочленов комплексного аргумента формулу (2.14) следует представить выражением

$$A_k = x \operatorname{Re} A_{k-1} - y \operatorname{Im} A_{k-1} + a_{n-k} + j\,(y \operatorname{Re} A_{k-1} + x \operatorname{Im} A_{k-1}),$$

где $k = 1, 2, \ldots, n$, $A_0 = a_n$ и $A_n = A(p) = \operatorname{Re} A(p) + \mathrm{j} \operatorname{Im} A(p)$.

Вычисления по этой формуле реализуются достаточно длинным фрагментом, который приходится выносить в подпрограмму. Так как для размещения аргумента требуется два регистра, то даже при размещении вещественной составляющей накапливаемого результата в операционном стеке программная реализация на входном языке ЯМК34 не обеспечивает при однократном выполнении программы вычисления многочленов степени больше 10.

Программа 15/34. Вычисление многочленов $A(p)$ комплексного аргумента $p = x + \mathrm{j}x$ степени $n \leqslant 10$

ПС	XY	ПВ	ИПА	×	ИП9	+	ИПС	ИПА	×
ПД	XY	ИП8	ПП	42	ИП7	ПП	42	ИП6	ПП
42	ИП5	ПП	42	ИП4	ПП	42	ИП3	ПП	42
ИП2	ПП	42	ИП1	ПП	42	ИП0	ПП	42	С/П
ВП	00	ИПД	ИПС	×	—	XY	ИПВ	×	+
XY	ИПС	×	ИПД	ИПВ	×	+	ПД	XY	В/О

Инструкция: ($a_0 = P0$, $a_1 = P1$, ..., $a_9 = P9$, $a_{10} = PA$) $x = PY$, $y = PX$ (В/О) С/П $PX = \operatorname{Re} A(p)$, $PY = PД = \operatorname{Im} A(p)$.

Контрольный пример: при $p = 1 + \mathrm{j}1$ время вычисления многочлена $A(p) = 10p^{10} + 9p^9 + 8p^8 + 7p^7 + 6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 1 = 288 + \mathrm{j}351$ составляет около 67 с.

Можно повысить максимальную степень вычисляемых многочленов, составив программу с вводом коэффициентов при старших степенях в операционный стек [13], до $n = 13$, но в этом случае увеличиваются затраты времени на ввод исходных данных при вычислениях многочлена в интервале изменения аргумента. В общем случае при вычислении многочленов комплексного аргумента степени $n > 10$ следует воспользоваться программой 125/34 приложения с последовательным вводом коэффициентов.

При $n < 10$ для сокращения затрат времени при многократном использовании программы 15/34 ее следует сократить, изъяв избыточные обращения к регистрам памяти и соответственно изменив адрес подпрограммы. По этим программам можно вычислять многочлены мнимого аргумента, приняв $x = 0$, но для сокращения затрат времени при вычислении многочленов мнимого аргумента целесообразно использовать приведенные в Приложении программы. В этих программах используется итерационная формула (2.14), представленная выражением

$$A_k = -y \operatorname{Im} A_{k-1} + a_{n-k} + \mathrm{j}y \operatorname{Re} A_{k-1}, \quad k = 1, 2, \ldots, n$$

при $A_0 = a_n$ и $A_n = \operatorname{Re} A(p) + \mathrm{j} \operatorname{Im} A(p)$.

Программу 127/34 при $n < 12$ для сокращения времени счета можно модифицировать, изменив начало программы и адрес обращения

к подпрограмме. Например, при $n = 10$ программу следует начать фрагментом

ПД ИПА × ИП9 XY ИП8 ПП 35 ...

В ряде задач приходится вычислять коэффициенты степенного многочлена при изменении аргумента исходного многочлена $A(p)$ на постоянную вещественную слагаемую x_0. В этих случаях обычно используют обобщенную схему Горнера, отображаемую итерационной формулой

$$b_{n-j}^{(k)} = b_{n-j}^{(k-1)} x_0 + b_{n-j-1}^{(k-1)}, \quad j = 0, 1, \ldots, n-k+1, \quad k = 1, 2, \ldots, n, \tag{4.8}$$

где на первом цикле $b_{n-j}^{(0)} = a_{n-j}$, а по окончании каждого k-го цикла вычисляются коэффициенты $b_{k-1}^{(k)} = b_{n-k+1}$ преобразованного многочлена.

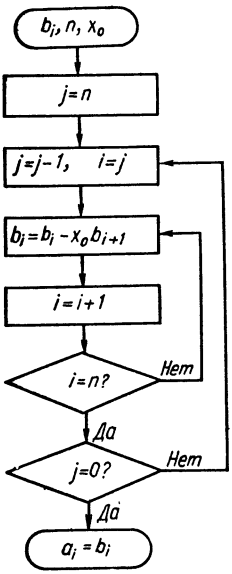


Рис. 27

При реализации этой формулы на входном языке ЯМК34 удобно использовать операторы косвенного обращения к памяти, обеспечивающие выполнение требуемого числа итераций на каждом цикле и размещение коэффициентов преобразованного многочлена $B(p)$ вместо коэффициентов исходного многочлена $A(p)$ при тех же степенях аргумента. Подобное преобразование соответствует обеспечению равенства

$$\sum_{i=0}^{n} a_i p^i = \sum_{i=0}^{n} b_i (p - x_0)^i$$

и может быть реализовано программным представлением алгоритма, схема которого показана на рис. 27.

Программа 16/34. Вычисление коэффициентов многочлена $A(p)$ степени $n \leqslant 10$ по коэффициентам многочлена $B(p - x_0)$ при смещении начала отсчета аргумента на постоянную вещественную составляющую x_0

ПД	ПС	ПВ	→	КИПВ	ИПВ	1	—	ПВ →
×	КИПВ	+	КПВ	→	ИПД	ИПВ	2	+ ПВ
—	$x<0$	03	→	ИПС	1	—	$x=0$	01 С/П

Инструкция: $b_0 = P0$, $b_1 = P1$, ..., $b_9 = P9$, $b_{10} = PA$, $x_0 = PY$, $n = PX$ В/О С/П $PX = 0$, $P0 = a_0$, $P1 = a_1$, ..., $PA = a_{10}$.

Контрольный пример: для $B(p - x_0) = 7p^7 + 6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 10$ при $x_0 = 0{,}5$ получим $A(p) = 7p^7 + 30{,}5p^6 + 53{,}75p^5 + 69{,}625p^4 + 53{,}8125p^3 + 28{,}96875p^2 + 10{,}703126p + 11{,}929688$ (время счета около 3 мин).

Решение многих инженерных задач связано с выполнением арифме-

тических операций над многочленами с численными значениями коэффициентов и символьным представлением аргумента. Такие операции аналогичны арифметическим операциям над десятичными представлениями чисел. Эта аналогия не случайна, так как десятичное представление числа в позиционной системе $a_n a_{n-1} \ldots a_1 a_0$ с однозначными числами a_i в десятичных разрядах соответствует степенному многочлену

$$A(10) = a_n 10^n + a_{n-1} 10^{n-1} + \cdots + a_2 10^2 + a_1 10 + a_0.$$

Сложение и вычитание степенных многочленов сводится к суммированию коэффициентов при одинаковых степенях аргумента (в общем случае без ограничений на величины сумм, характерных для позиционных представлений чисел), и автоматизация целесообразна лишь для умножения и деления.

Умножение многочленов сводится к вычислению частных произведений первого многочлена $A(p)$ на коэффициенты b_m, b_{m-1}, ..., b_0 второго многочлена, смещению очередного частного произведения на один член влево и суммированию коэффициентов частных произведений при одинаковых степенях аргумента. Следовательно, достаточно хранить в памяти микрокалькулятора $m + 1$ коэффициентов и выделить $m + 1$ регистров для хранения текущих сумм с последовательным вводом коэффициентов b_i и выводом коэффициентов c_{i+n} произведения при каждом выполнении программы. Полному использованию емкости числовой памяти микрокалькулятора с входным языком ЯМК34 в этом случае соответствует следующая программа.

Программа 17/34. Вычисление коэффициентов c_i произведения $C(p)$ многочлена $A(p)$ степени $n \leqslant 6$ на многочлен $B(p)$ произвольной степени

ПД	Сх	П7	П8	П9	ПА	ПВ	ПС	ИПД	БП
12	С/П	ПД	ИП6	×	ИПС	+	ИПД	ИП5	×
ИПВ	+	ПС	→	ИПД	ИП4	×	ИПА	+	ПВ
→	ИПД	ИП3	×	ИП9	+	ПА	→	ИПД	ИП2
×	ИП8	+	П9	→	ИПД	ИП1	×	ИП7	+
П8	→	ИПД	ИПО	×	П7	ХУ	ИПД	БП	11

Инструкция: ($a_0 = P0$, $a_1 = P1$, ..., $a_6 = P6$) $b_m = PX$ В/О С/П $PX = b_m$, $PY = c_{m+6}$, $b_{m-1} = PX$ С/П $PX = b_{m-1}$, $PY = c_{m+5}$, $b_{m-2} = PX$ С/П $PX = b_{m-2}$, $PY = c_{m+4}$, ..., $b_0 = PX$ С/П $PX = b_0$, $PY = c_6$, $PC = c_5$, $PB = c_4$, $PA = c_3$, $P9 = c_2$, $P8 = c_1$, $P7 = c_0$.

Контрольный пример: время вычисления одного коэффициента произведения $(6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 1)(5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 1) = 30p^{11} + 49p^{10} + 58p^9 + 58p^8 + 50p^7 + 41p^6 + 30p^5 + 18p^4 + 10p^3 + 5p^2 + 2p + 1$ около 15 с.

При $n < 6$ программу целесообразно упростить, как показано в приложении (программа 128/34) для умножения многочлена произвольной степени на многочлен второй степени. Эти программы применимы и для умножения по частям многочленов более высоких степеней, предварительно представленных разложениями (4.7). При вычислении

произведения идентичных многочленов (квадрата многочлена) их предельные степени можно увеличить.

Программа 18/34. Вычисление коэффициентов квадрата многочлена $A(p)$ степени $n \leqslant 10$

ИПД	ПС	Сх	ИПД	ПЗ	ХУ	КИПВ	КИПС	×	ИПВ
ИПС	—	$x \neq 0$	18	ХУ	2	×	ХУ	→	+
ИПС	1	+	ПС	ИПВ	1	—	ПВ	—	1
—	$x \geqslant 0$	05	ХУ	С/П	ИПВ	ИПС	+	1	—
ПВ	ИПД	—	$x < 0$	01	Сх	ПС	БП	06	

Инструкция: ($a_0 = P0$, $a_1 = P1$, . . . , $a_9 = P9$, $a_{10} = PA$, $n = РД$ В/О С/П $PX = c_{2n}$ С/П $PX = c_{2n-1}$ . . . С/П $PX = c_1$ С/П $PX = c_0$)

Контрольный пример: $(3p^3 + 2p^2 + p + 0{,}5)^2 = 9p^6 + 12p^5 + 10p^4 + 7p^3 + 3p^2 + p + 0{,}25$.

В некоторых задачах возникает необходимость в вычислении произведения многочленов $C(p) = A(p)\,A(-p)$, содержащего только четные степени аргумента. Это позволяет несколько повысить максимальную степень $A(p)$.

Программа 19/34. Вычисление коэффициентов произведения $C(p) = A(p)\,A(-p)$ многочленов степени $n \leqslant 11$

Сх	ПС	ПД	КИПС	x^2	ИПД	$x \neq 0$	32	1	—
ПД	ХУ	ИПС	1	+	ПС	1	2	—	$x \neq 0$
31	ХУ	КИПС	КИПД	×	2	×	ХУ	—	БП
05	ХУ	—	ИПС	ИПД	+	2	÷	1	+
ХУ	С/П	ХУ	БП	01					

Инструкция: ($a_0 = P0$, $a_1 = P1$, . . . , $a_{11} = PB$, вместо отсутствующих коэффициентов занести нули) В/О С/П $PX = c_0$ С/П $PX = c_2$ С/П $PX = c_4$ . . . С/П $PX = c_{2n}$.

Контрольный пример: для $A(p) = 3p^3 + 2p^2 + p + 0{,}5$ получим $C(p) = A(p)\,A(-p) = -9p^6 - 2p^4 + p^2 + 0{,}25$.

На входном языке ЯМК21 можно запрограммировать вычисление подобных произведений для многочленов степени $n \leqslant 8$ (или $n \leqslant 5$ по программе 151/21 приложения).

Деление многочленов аналогично делению многозначных чисел, но в этом случае даже при последовательном выводе коэффициентов частного в памяти микрокалькулятора приходится хранить коэффициенты делителя и преобразуемые после каждого деления коэффициенты делимого. Реализация алгоритма деления по формуле (4.5) наглядно иллюстрируется следующей программой для часто встречающейся задачи деления многочлена на двучлен первого порядка.

Программа 20/34. Деление многочлена $A(p)$ степени $n \leqslant 12$ на двучлен $p + x_0$

↑	↑	↑	→	КИПД	×	ИПД	1	—	ПД
→	КИПД	+	КПД	→	ИПД	$x = 0$	03	ИПО	С/П

Инструкция: $a_0 = P0$, $a_1 = P1$, ..., $a_9 = P9$, $a_{10} = PA$, $a_{11} = PB$, $a_{12} = PC$, $n = PD$, $-x_0 = PX$ В/О С/П $PX = r$ (остаток от деления), $PC = b_{11}$, $PB = b_{10}$, $PA = b_9$, ..., $P2 = b_1$, $P1 = b_0$.

Контрольный пример: $(3p^3 + 15p^2 + 24p + 30)(p + 3) = 3p^2 + 6p + 6$ при остатке $r = 12$ (вводится $-3 = PX$).

Подобную программу удобно использовать в качестве подпрограммы или фрагмента деления многочленов в более сложных программах. Для деления многочленов произвольной степени на двучлен $p - x_0$ можно воспользоваться программами 129/34 и 152/21 приложения.

Используя операторы косвенного обращения к памяти входного языка ЯМК34, несложно составить программы деления многочленов на многочлены более высокой степени, например, второй.

Программа 21/34. Деление многочлена $A(p)$ степени $n \leqslant 9$ на трехчлен $p^2 + b_1 p + b_0$

ИПД	1	—	ПС	КИПД	КИПС	ИПС	1	—	ПС
$x \geqslant 0$	26	→	XY	ИПА	×	—	XY	КИПС	XY
КПС	ИПВ	×	—	БП	06	→	XY	С/П	

Инструкция: ($b_1 = PA$, $b_0 = PB$, $n = PД$) $a_0 = P0$, $a_1 = P1$, $a_2 = P2$, ..., $a_3 = P9$ В/О С/П $PX = r_1$ [(коэффициент остатка при первой степени), $PY = r_0$ (коэффициент остатка при нулевой степени), $P0 = c_0$, $P1 = c_1$, $P2 = c_2$, ...

Контрольный пример: $(3p^4 + 15p^3 + 24p^2 + 30p + 10)/(p^2 + 2p + 5) = 3p^2 + 9p - 9$ при остатке $3p + 55$.

При составлении подобных программ на входном языке ЯМК21 приходится использовать кольцевой стек памяти, но и в этом случае максимальные степени делимого и делителя оказываются меньше. Программирование упрощается при хранении в памяти только коэффициентов делителя и последовательном вводе коэффициентов делимого, степень которого в этом случае не ограничена.

В ряде инженерных задач приходится выполнять операции над дробно-рациональными функциями (отношениями степенных многочленов). Составление программ в этом случае [11, 13] в основном сводится к реализации рассмотренных операций над многочленами и комплексными числами. Поэтому ограничимся лишь разложением дробно-рациональных функций в простые дроби. С этим разложением связаны методы синтеза некоторых устройств и удобная для практики оценка устойчивости степенных многочленов и физических объектов, свойства которых отображены степенными многочленами [13]. Степенной многочлен называют *устойчивым*, если вещественные составляющие всех его корней отрицательны. Оценку устойчивости можно выполнить различными способами, не вычисляя корней [11], но практически удобный и относительно просто реализуемый метод основан на критерии **Гурвица**.

При использовании этого критерия анализируемый многочлен представляют суммой $A(p) = M(p) + N(p)$ четной $M(p) = \cdots +$

$+ a_4p^4 + a_2p^2 + a_0$ и нечетной $N(p) = \cdots + a_5p^5 + a_3p^3 + a_1p$ частей и раскладывают в цепную дробь отношениие $M(p)/N(p)$ при четной степени n многочлена или отношение $N(p)/M(p)$ при нечетной степени, например,

$$\frac{N(p)}{M(p)} = c_1p + \cfrac{1}{c_2p + \cfrac{1}{c_3p + \cdots + \cfrac{1}{c_np}}}.$$

Многочлен устойчив, если все n коэффициентов c_i положительны.

Программа 22/34. Разложение в цепную дробь с коэффициентами c_i отношения четной и нечетной частей многочлена $A(p)$ степени $n \leqslant 10$ или отношения двух многочленов с суммой всех членов $m \leqslant 11$

ПД	ПС	КИПС	ИПС	1	—	ПС	→	КИПС	÷
↑	ИПС	1	+	ПС	→	КПС	→	↑	ИПС
2	--	ПС	$x \geqslant 0$	45	→	КИПС	ИПС	1	—
ПС	$x < 0$	38	→	XY	0	БП	41	→	XY
КИПС	×	—	БП	11	ИПД	1	—	$x = 0$	00
С/П									

Инструкция: для оценки устойчивости многочлена $A(p)$ ввести $a_0 = P0$, $a_1 = P1$, $a_2 = P2$, ..., $n = PX$ В/О С/П $PX = 0$, $P1 = c_1$, $P2 = c_2$, $P3 = c_3$, ... ; для разложения отношения многочленов, степени которых отличаются на единицу или менее, коэффициенты многочлена с большей или равной степенью занести в регистры *0*, *2*, ... с четными номерами в порядке возрастания степеней аргумента, коэффициенты второго многочлена занести соответственно в регистры с нечетными номерами *1*, *3*, ...

Контрольный пример: для многочлена $A(p) = 6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 0{,}5$ получим коэффициенты цепной дроби $c_1 = 3{,}5$; $c_2 = 0{,}28571428$; $c_3 = -14$; $c_4 = -0{,}057142857$; $c_5 = 12{,}5$; $c_6 = 1{,}2$; следовательно, многочлен неустойчив (время счета 130 с).

Для оценки устойчивости многочленов степени $n > 10$ можно принять $p = j\omega$ и воспользоваться программами для вычисления многочленов мнимого аргумента, построив на плоскости с координатами $\mathrm{Re}A(j\omega)$ и $\mathrm{Im}A(j\omega)$ геометрическое место точек (годограф) при изменении ω от нуля до такого значения, при котором годограф явно уходит в бесконечность. Если годограф не охватывает начало координат (при движении по годографу в сторону увеличения ω охваченная им область остается справа), то анализируемый многочлен устойчив.

5. ИНТЕРПОЛИРОВАНИЕ ТАБЛИЧНЫХ МОДЕЛЕЙ

Функциональные зависимости вида $y = y(x)$ часто задаются (например, при использовании таблиц функций или результатов экспериментальных измерений) табличными моделями в виде дискретных по-

следовательностей пар значений аргумента x_i и функции $y_i = y(x_i)$, причем обычно предполагается, что моделируемая функция достаточно гладкая между табличными значениями. Вычисление функции для значений аргумента x, лежащих между табличными значениями x_i (узлами интерполяции или узлами табличной модели), называют *интерполированием*, а вычисление функции для значений аргумента, лежащих за пределами табличного интервала аргумента,— *экстраполированием*.

Простейший способ интерполирования при произвольно расположенных узлах $x_i < x_{i+1}$ заключается в приближении табличной модели на каждом интервале $[x_i, x_{i+1}]$ линейными функциями $P_i(x) = a_{0i} + a_{1i}x$, совпадающими с граничными значениями y_i и y_{i+1}. Эта же функция экстраполирует табличную модель за пределами рассматриваемого интервала. Линейное интерполирование (и экстраполирование) обычно выполняют согласно формуле

$$P_i(x) = y_i + (x - x_i)(y_{i+1} - y_i)/(x_{i+1} - x_i),$$

где x_i и x_{i+1} — ближайшие к x меньший и больший по величине узлы интерполяции; y_i и y_{i+1} — соответствующие им значения функции.

Эта формула легко программируется, но линейное интерполирование недостаточно точно, так как производные кусочно-линейной функции разрывны в узлах интерполяции, хотя реальные процессы не могут изменяться мгновенно и их обычно моделируют аналитическими (имеющими все производные) функциями.

Аналитическое представление табличных моделей чаще всего выбирают в виде функциональных многочленов

$$P(x) = \sum_{i=0}^{n} a_i f(i, x), \tag{4.9}$$

где a_i — коэффициенты; $f(i, x)$ — аналитическая базовая функция.

Для построения такого интерполирующего выражения по табличной модели с $n + 1$ узлами применяют метод неопределенных коэффициентов, заключающийся в определении коэффициентов a_i по системе уравнений

$$y_i = \sum_{k=0}^{n} a_k f(k, x_i), \quad i = 0, 1, 2, \ldots, n. \tag{4.10}$$

В качестве базовых выбирают различные элементарные (например, тригонометрические при использовании в качестве интерполирующего ряда Фурье) или специальные функции. Часто в качестве базовых используют степенные функции, и интерполяционная формула (4.9) в этом случае представляет собой удобный для вычислений степенной многочлен. Непосредственное определение коэффициентов такого многочлена по системе уравнений (4.10) достаточно громоздко и поэтому обычно используют различные эквивалентные представления степенных многочленов, не требующие поиска решения системы уравнений.

В общем случае неравноотстоящих узлов табличной модели ее обычно интерполируют по формуле Лагранжа

$$P(x)=\sum_{i=0}^{n}\frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}y_i, \tag{4.11}$$

где x_i и y_i — узлы и отсчеты функции в узлах табличной модели; x — заданное значение аргумента в точке интерполирования.

Несмотря на кажущуюся громоздкость этой формулы, ее программную реализацию можно упростить, образовав из множителей числителя и знаменателя квадратную матрицу

$$\begin{bmatrix} x-x_0 & x_0-x_1 & x_0-x_2 & \cdots & x_0-x_n \\ x_1-x_0 & x-x_1 & x_1-x_2 & \cdots & x_1-x_n \\ x_2-x_0 & x_2-x_1 & x-x_2 & \cdots & x_1-x_n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_n-x_0 & x_n-x_1 & x_n-x_2 & \cdots & x-x_n \end{bmatrix}.$$

Обозначив произведения элементов i-й строки и элементов главной диагонали этой матрицы соответственно символами $H_i(x)$ и $П_{n+1}(x)$, представим формулу (4.11) выражением

$$y(x)\approx П_{n+1}(x)\sum_{i=0}^{n}y_i/H_i(x), \tag{4.12}$$

позволяющим свести задачу интерполирования в основном к вычислению множителей $H_i(x)$ и $П_{n+1}(x)$.

При программировании микрокалькуляторов с входным языком ЯМК34 для вычисления этих множителей целесообразно использовать косвенный вызов из памяти значений x_i согласно их положению в строках и столбцах составленной матрицы. Так как требуется два адресных регистра (для изменения номеров i по строкам и столбцам матрицы) и два регистра для хранения значений x и y_i, то, при последовательном вводе отсчетов y_i и хранении текущих результатов в операционном стеке, в числовой памяти удается разместить 10 значений x_i, что достаточно для большинства практических задач интерполирования.

Программа 23/34. Интерполирование табличных моделей с $n+1\leqslant 10$ неравноотстоящими узлами по формуле Лагранжа

ПВ	Сх	↑	↑	→	ПС	С/П	XY	→	ИПД
1	—	ПА	→	КИПС	КИПА	—	$x\neq 0$	25	÷
ИПВ	КИПА	—	×	↑	→	ИПА	$x=0$	10	→
+	ИПС	1	+	↑	ИПД	—	$x=0$	04	→
БП	04								

Инструкция: ($x_0 = P0$, $x_1 = P1$, $x_2 = P2$, ..., $n+1 = РД$) $x =$ $= PX$ В/О С/П $PX = 0$, $y_0 = PX$ С/П $PX = 1$, $y_1 = PX$ С/П ... ... $PX = n$, $y_n = PX$ С/П $PX = y(x)$.

Отказавшись от индикации номера вводимого отсчета функции y_i, можно сократить эту программу на 9 шагов, но она станет менее удобной для пользователя. Так как при выполнении программы текущие данные хранятся в регистрах операционного стека, то ошибочно набранное значение y_i допустимо лишь стирать вводом оператора Сх.

Для проверки интерполирующих программ и точности интерполирования удобно использовать таблицы функций с известным числом верных цифр. Так, выбрав из четырехзначных таблиц гамма-функции [6] шесть отсчетов $\Gamma_0(1) = 1$; $\Gamma_1(1{,}3) = 0{,}8975$; $\Gamma_2(1{,}4) = 0{,}8873$; $\Gamma_3(1{,}6) = 0{,}8935$; $\Gamma_4(1{,}8) = 0{,}9314$; $\Gamma_5(2) = 1$, по программе 23/34 для $x = 1{,}723$ получим $\Gamma(x) = 0{,}91317$ (время счета после каждого ввода y около 40 с) при табличном значении $\Gamma(1{,}723) = 0{,}9132$.

В качестве интерполирующей функции часто используют степенной многочлен Ньютона

$$P(x) = b_0 + \sum_{i=1}^{n} b_i \prod_{j=0}^{i} (x - x_j), \qquad (4.13)$$

где $b_0 = y_0$, а остальные коэффициенты b_j выражаются через разделенные разности [4,6].

В процессе вычислений коэффициентов такого многочлена в памяти микрокалькулятора приходится хранить $2n$ значений x_i и b_i и поэтому на входном языке ЯМК34 даже при последовательном вводе отсчетов y_i с прекращением вычислений и использовании косвенной адресации удается программировать определение коэффициентов не более чем по шести узлам интерполяции.

Для практических целей более удобны интерполирующие функции в виде обычных степенных многочленов (4.4), коэффициенты a_i которых можно вычислить по коэффициентам многочлена (4.13) согласно следующему алгоритму:

1. Принять $j = 0$, $b_n = 0$.
2. Принять $j = j + 1$.
3. Вычислить

$$b_i^{(j)} = \begin{cases} b_i^{(j-1)} - b_{i+1}^{(j-1)} & \text{при } j + i = n, \\ b_j^{(j-1)} & \text{при } j + i < n. \end{cases}$$

4. Если $j = n + 1$, то закончить вычисления, приняв $a_i = b_i^{(n+1)}$, иначе перейти к п. 2.

Совместив программные реализации алгоритмов вычисления коэффициентов b_i и a_i с фрагментами для вычисления значений степенного многочлена по заданному аргументу и смещения начала его отсчета, получим следующую удобную для практических целей универсальную программу.

Программа 24/34. Вычисление коэффициентов b_i интерполирующего многочлена Ньютона, коэффициентов a_i и a'_i интерполирующего степенного многочлена при исходном и смещенном на ξ началах

отсчета аргумента и результата интерполирования $y(x)$ по табличной модели с $n+1 \leqslant 6$ неравноотстоящими узлами

ПД	XY	П7	7	П1	1	4	П0	С/П П8
→	ИП8	КИП0	—	XY	ИП0	5	—	П0 →
КИП0	—	XY	÷	ИП0	6	+	П0	6 —
ИП1	—	$x=0$	10	→	КП1	ИП8	КП0	БП 05
ИП1	1	+	П8	7	+	П0	6	— П1
9	—	$x\neq 0$	79	КИП0	↑	→	↑	КИП1 ХУ
КИП1	×	—	ИП1	2	+	П1	→	КП1 →
ИП1	ИП8	—	$x=0$	56	ИП0	2	БП	45 С/П
↑	ИП9	П1	Сх	ИП8	П0	→	×	КИП0 +
ИП0	2	+	L1	Ɛ5	→	БП	79	

Инструкция: $y_0 = PY$, $x_0 = PX$ В/О С/П $y_1 = PY$, $x_1 = PX$ С/П $\ldots$ $y_n = PY$, $x_n = PX$ С/П $P7 = b_0$, $P6 = b_1$, $P5 = b_2$, $\ldots$, $P2 = b_5$, $РД = x_0$, $РС = x_1$, $РВ = x_2, \ldots$, $P8 = x_5$; для вычисления коэффициентов степенного многочлена БП 4 0 С/П $P7 = a_0$, $P6 = a_1$, $P5 = a_2, \ldots$, $P2 = a_5$, $P8 = 8 - n$; для вычисления значений $y(x)$ выполнить $(n+1 = P9)$, $x = PX$ С/П $PX = y(x)$; для вычисления коэффициентов a_i' степенного многочлена при смещении начала отсчета аргумента в точку $x_0' = x_0 + \xi$ выполнить $\xi = РД = РС = \cdots = P(14-n)$, $7 - n = P1$ БП 40 С/П $P7 = a_0'$, $P6 = a_1'$, $P5 = a_2', \ldots$, $P2 = a_5'$.

Для табличной модели гамма-функции из предыдущего контрольного примера по этой программе получим $b_0 = 1$; $b_1 = -0{,}34166666$; $b_2 = 0{,}5991666$; $b_3 = -0{,}259722$; $b_4 = 0{,}206944$; $b_5 = -0{,}102182$; $a_0 = 3{,}7314954$; $a_1 = -6{,}729417$; $a_2 = 6{,}565789$; $a_3 = -3{,}3981216$; $a_4 = 0{,}9324362$; $a_5 = -0{,}102182$; для $x = 1{,}723$ получим $\Gamma(x) = 0{,}9131714$; для проверки влияния операционных погрешностей вычислим значения функции в узлах интерполяции $\Gamma(1) = 1$; $\Gamma(1{,}3) = 0{,}8975$; $\Gamma(1{,}4) = 0{,}8873001$; $\Gamma(1{,}8) = 0{,}9314001$; $\Gamma(2) = 0{,}9999998$; для $\xi = 1$ и $7 - n = 2$ вычислим $a_0' = 21{,}459441$; $a_1' = -34{,}296015$; $a_2' = 23{,}376591$; $a_3' = -8{,}1496864$; $a_4 = 1{,}4433462$, $a_5' = -0{,}102182$ и для $x' = x + 1 = 2{,}723$ получим $\Gamma(x) = 0{,}91317$.

При равноотстоящих узлах табличной модели с постоянным шагом $h = x_{i+1} - x_i$ целесообразно использовать нормированные значения аргумента $q = (x - x_0)/h$. Если начало отсчета выбрать в начальном узле x_0, то получим целочисленные значения узлов $q_i = (x_i - x_0)/h = i = 0, 1, 2, \ldots, n$. В этом случае формулу (4.12) можно представить приближенным равенством

$$y(x) \approx (\Pi_{n+1}(q)/n!) \sum_{i=0}^{n} (-1)^{n-i} y_i C_n^i/(q-i),$$

где $\Pi_{n+1}(q) = q(q-1)\ldots(q-n)$; $C_n^i = n!/i!(n-i)!$, а q — нормированное значение аргумента в точке интерполирования.

Результаты вычислений по этой формуле отличаются малыми операционными погрешностями, так как большинство операций выполняется над целыми числами без округления. Кроме того, отпадает необходимость в хранении нормированных узлов i, так как их можно формировать программно с помощью счетчиков, что обеспечивает интерполирование табличных моделей с практически неограниченным числом узлов. В следующей программе при последовательном вводе значений y_i вычисление факториала $i!$ (равного $n!$ после ввода всех $n+1$ отсчетов y_i) и числа сочетаний из n по i обеспечивается с помощью операторов косвенного вызова из памяти при адресных регистрах *2* и *5*.

Программа 25/34. Интерполирование по формуле Лагранжа табличных моделей с произвольным числом $n+1$ равноотстоящих узлов

ИП7	ПД	ИП8	П0	π	$\times$	cos	П9	$\times$	$\div$
ПС	1	П2	П5	ИП5	С/П	ИП7	ИП5	—	$\uparrow$
ИПД	$\times$	ПД	$\rightarrow$	$\div$	ИП0	/—/	ИП9	$\times$	П9
$\times$	ИП5	ИП2	$\times$	П2	$\div$	ИПС	+	ПС	КИП5
КИП0	ИП0	$x=0$	14	ИПС	ИПД	$\times$	ИП2	$\div$	С/П

Инструкция: $q=(x-x_0)/h=P7$, $n=P8$, $y_0=PX$ В/О С/П $PX=1$, $y_1=PX$ С/П $PX=2$, $y_2=PX$ С/П ... $y_n=PX$ С/П $PX=y(x)$. Переключатель Р — Г в положении Р.

Для табличной модели гамма-функции $\Gamma_0=1$; $\Gamma_1=0{,}9182$; $\Gamma_2=0{,}8873$; $\Gamma_3=0{,}8935$; $\Gamma_4=0{,}9314$; $\Gamma_5=1$ с шагом $h=0{,}2$ и $x_0=1$ по этой программе при округлении до четырех значащих цифр получим значения $\Gamma(1{,}115)=0{,}9456$; $\Gamma(1{,}326)=0{,}8939$; $\Gamma(1{,}488)=0{,}8859$; $\Gamma(1{,}723)=0{,}9132$; $\Gamma(1{,}907)=0{,}9643$, точно совпадающие, кроме последнего значения, с табличными данными [6].

При интерполировании табличных данных с помощью программы 25/34 для каждого значения аргумента в точке интерполирования приходится повторять ввод отсчетов функции. Поэтому в тех случаях, когда приходится многократно интерполировать табличную модель с небольшим числом равноотстоящих узлов, целесообразно составлять программы, с помощью которых вначале вычисляются коэффициенты интерполирующего многочлена, по которому вычисляется интерполируемое значение $y(x)$ при вводе только значений аргумента.

Вычислив для табличной модели с равноотстоящими узлами коэффициенты многочлена Ньютона (4.13), можно найти коэффициенты a_i эквивалентного степенного многочлена согласно следующему алгоритму:

1. Принять $j=0$, $\alpha_i^{(0)}=\alpha_i$.
2. Принять $j=j+1$.
3. Вычислить

$$\alpha_i^{(j)}=\begin{cases}\alpha_i^{(j-1)} & \text{при } i<j,\\ \alpha_i^{(j-1)}-(n-j)\,\alpha_{i+1}^{(j-1)} & \text{при } i\geqslant j.\end{cases}$$

4. Если $j = n - 1$, то принять $a_i = \alpha_i^{(n-1)}$, иначе перейти к п. 2.

Вычисление коэффициентов Ньютона и эквивалентного степенного многочлена на входном языке ЯМК21 приходится реализовать различными программами [11], но на входном языке ЯМК34 эти операции и вычисление интерполирующих значений степенного многочлена удается совместить в одной следующей универсальной программе.

Программа 26/34. Вычисление коэффициентов α_i интерполирующего многочлена Ньютона, коэффициентов a_i и значений $A(z)$ степенного многочлена при $z_0 = x - x_0$ и числе $n + 1 \leqslant 10$ равноотстоящих узлов

ПА	Cx	ПД	КИПД	ПВ	ИПД	1	+	ПС	ИПД
1	+	ПД	ИПВ	КИПД	ПВ	XY	—	ИПС	÷
КПД	ИПД	ИПА	—	$x = 0$	09	ИПС	ПД	ИПА	—
$x = 0$	03	ИПА	1	—	ПС	ПВ	ИПВ	1	+
ПД	КИПВ	КИПД	ИПС	×	—	КПВ	ИПВ	1	+
ПВ	ИПА	—	$x = 0$	37	ИПС	1	—	$x=0$	35
ПД	С/П	ПВ	КИПД	ИПС	÷	КПД	С/П	ИПС	ИПВ
×	ПС	ИПД	1	+	ПД	БП	63	↑	1
↑	Cx	↑	→	×	КИПД	+	ИПД	1	—
ПД	$x < 0$	83	→	С/П	БП	78			

Инструкция: $y_0 = P0$, $y_1 = P1$, ..., $y_n = Pn$, $n = PX$ В/0 С/П $PX = 0$ ($P0 = \alpha_0$, $P1 = \alpha_1$, $Pn = \alpha_n$) $h = PX$ С/П $PX = P0 = a_0$ С/П $PX = P1 = a_1$ ... С/П $PX = Pn = a_n$; для вычисления значений $y(x)$ выполнять $n = РД$, $z = x - x_0 = PX$ БП 78 С/П $PX = y(x)$.

Для табличной модели гамма-функции $\Gamma_0 = 1$; $\Gamma_1 = 0{,}9182$; $\Gamma_2 = 0{,}8873$; $\Gamma_3 = 0{,}8935$; $\Gamma_4 = 0{,}9314$; $\Gamma_5 = 1$ при $h = 0{,}2$ по этой программе получим $\alpha_0 = 1$; $\alpha_1 = -0{,}11475$; $\alpha_2 = 0{,}037866667$; $\alpha_3 = -0{,}0055666667$; $\alpha_4 = 0{,}00068333334$; $\alpha_5 = -0{,}000033333337$; $a_0 = 1$; $a_1 = -0{,}57375$; $a_2 = 0{,}94666667$; $a_3 = -0{,}69583333$; $a_4 = 0{,}42708333$; $a_5 = -0{,}10416666$ и при $n = 5$, $z = x - 1$ найдем $A(0{,}115) = \Gamma(1{,}115) = 0{,}9456$; $\Gamma(1{,}326) = 0{,}8939$; $\Gamma(1{,}488) = 0{,}8859$; $\Gamma(1{,}723) = 0{,}9132$; $\Gamma(1{,}907) = 0{,}9643$ с той же точностью, что и с помощью предыдущей программы.

В последние годы для интерполирования используют функционалы, называемые *сплайнами* [1] и достаточно хорошо отображающие физические зависимости. На каждом интервале $[x_i, x_{i+1}]$ между узлами табличной модели сплайн $S(x)$ обычно представляют кубическим многочленом, коэффициенты которого выбирают так, чтобы во всех узлах обеспечивалась непрерывность сплайна, а в неокончных узлах — и непрерывность его первой и второй производных. При этих условиях число уравнений $S(x_0) = y_0$, $S(x_n) = y_n$, $S_{i+1}(x_{i+1}) = S_{i+2}(x_{i+2}) = y_{i+1}$, $S'_{i+1}(x_{i+1}) = S'_{i+2}(x_{i+2})$, $S''_{i+1}(x_{i+1}) = S''_{i+2}(x_{i+1})$ равно $4n - 2$, тогда как для определения всех коэффициентов кубических многочленов требуется $4n$ уравнений.

При выборе дополнительных условий $S''(x_0) = S''(x_n) = 0$ сплайн называют естественным, но возможен выбор физически более обоснованных граничных условий. Так, при выборе значений $S''(x_0)$ и $S''(x_n)$, равных вторым производным степенных многочленов, построенным по четырем крайним узлам табличной модели с постоянным шагом, на каждом интервале между узлами

$$S(x) = y_{i+1}q + y_i\bar{q} + (q - q^3)\sigma_{i+1} + (\bar{q} - \bar{q}^3)\sigma_i,$$

где $q = (x - x_i)/h$; $\bar{q} = 1 - q$, а коэффициенты σ_i определяются решением системы уравнений с рассмотренными дополнительными условиями.

На входном языке ЯМК34 удается реализовать интерполирование кубическим сплайном табличных моделей с шестью равноотстоящими узлами при помощи следующих двух программ, первая из которых предназначена для определения коэффициентов кубических многочленов, а вторая обеспечивает собственно интерполирование.

Программа 27/34. Вычисление коэффициентов σ_i для интерполирования кубическим сплайном табличных моделей с шестью равноотстоящими узлами

П1	1	П6	П7	5	П0	С/П	КП6	L0	06
5	П0	ПП	Б6	П8	4	ПП	78	—	1
ИП7	1/x	—	÷	ПД	ИПС	+	ИП7	П0	÷
ПС	4	↑	7	+	П7	→	КИП7	+	4
/—/	ИП0	—	1/x	П0	÷	КП7	ИП7	8	—
x = 0	33	С/П	XY	КИП0	КИП0	2	×	—	КИП0
+	+	ИП0	9	+	П0	→	КП0	4	/—
ИП7	1/x	—	П7	÷	ИП0	4	—	П0	8
—	x = 0	53	XY	КИП0	XY	КИП0	КИП0	КИП0	—
3	×	—	КИП0	—	6	÷	В/О		

Инструкция: $y_0 = PX$ В/О С/П $y_1 = PX$ С/П ... $y_5 = PX$ С/П $PX = 0$ ($P1 = y_0$, $P2 = y_1$, $P3 = y_2$, $P4 = y_3$, $P5 = y_4$, $P6 = y_5$, $P8 = \sigma_0$, $P9 = \sigma_1$, $PA = \sigma_2$, $PB = \sigma_3$, $PC = \sigma_4$, $PД = \sigma_5$).

Программа 28/34. Вычисление сплайна по значениям аргумента

ИП7	—	...	...	...	÷	1	0	+ П0
КИП0	XY	ИП0	—	1	—	XY	Bx	x^2 1
—	×	×	ХУ	1	—	x^2	Bx	× Bx
—	КИП0	×	—	ИП0	5	—	П0	→ XY
КИП0	×	XY	—	XY	1	—	КИП0	× —
С/П	БП	00						

Инструкция: после выполнения программы 27/34, не изменяя содержимого регистров памяти, ввести программу 28/34, заменив многоточия операторами набора значения шага h табличной модели и выполнить ($x_0 = P7$) $x = PX$ (В/О) С/П $PX = S(x) \approx y(x)$.

Для табличной модели гамма-функции из предыдущего контрольного примера по этим программам получим $\sigma_0 = -1{,}0852891 \cdot 10^{-2}$; $\sigma_1 = -8{,}5513452 \cdot 10^{-3}$; $\sigma_2 = -5{,}8420549 \cdot 10^{-3}$; $\sigma_3 = -5{,}1801583 \times \times 10^{-3}$; $\sigma_4 = -5{,}1373014 \cdot 10^{-3}$; $\sigma_5 = -4{,}9706348 \cdot 10^{-3}$ и значения $\Gamma(1{,}115) = 0{,}9456$; $\Gamma(1{,}326) = 0{,}8938$; $\Gamma(1{,}488) = 0{,}8859$; $\Gamma(1{,}723) = = 0{,}9131$; $\Gamma(1{,}907) = 0{,}9643$.

Интерполирование сплайнами, отображаемыми кубическими параболами в каждом интервале между узлами табличной модели, отличается относительной точностью результатов при немонотонных зависимостях и больших интервалах между узлами, что оправдывает громоздкость вычислительных процедур.

6. АППРОКСИМАЦИЯ ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ

Аппроксимация (приближение) функциональных зависимостей интерполяционными многочленами достаточно громоздка и имеет смысл лишь в том случае, когда задана точная табличная модель и требуется высокая точность аппроксимации функции между узлами. Между тем точность результатов экспериментальных исследований, в особенности при их графическом представлении, обычно не превышает двух — четырех верных цифр, а часто оказывается еще более низкой. В таких случаях целесообразно использовать достаточно простые и удобные для вычислений аппроксимирующие функции, не обязательно совпадающие с заданными отсчетами функции в узлах табличной модели, но приближающиеся к ним согласно выбранному критерию отклонения.

Большинство монотонных функциональных зависимостей хорошо приближается простыми аппроксимирующими выражениями с двумя коэффициентами, указанными в табл. 10. Известны [11, 20] несложные способы выбора подобных аппроксимирующих формул, но в некоторых случаях эти способы могут привести к ошибочным выводам.

10. Аппроксимирующие функции с двумя коэффициентами

Функция		Связь с обобщенными величинами			
		f	x	A	B
Линейная	$f(x) = Ax + B$	φ	z	A^0	B^0
Логарифмическая	$f(x) = A\ln x + B$	φ	e^z	A^0	B^0
Гиперболическая	$f(x) = A/x + B$	φ	$1/z$	A^0	B^0
Дробно-рациональная	$f(x) = 1/(Ax + B)$	$1/\varphi$	z	A^0	B^0
Обратная логарифмическая	$f(x) = 1/(A\ln x + B)$	$1/\varphi$	e^z	A^0	B^0
Дробно-рациональная	$f(x) = x/(Ax + B)$	$1/\varphi$	$1/z$	A^0	B^0
Показательная	$f(x) = BA^x$	e^φ	z	e^{A^0}	e^{B^0}
Степенная	$f(x) = Bx^A$	e^φ	e^z	A^0	e^{B^0}

Более надежная оценка качества аппроксимации основана на составлении по двум узлам x_a и x_c аппроксимируемой зависимости

системы уравнений (4.10), решении этой системы относительно коэффициентов A и B аппроксимирующей формулы и вычислении погрешности $\Delta_b = f(x_b) - y_b$ для третьего узла x_b аппроксимируемой зависимости. Сравнивая значения Δ_b для различных аппроксимирующих формул, выбирают наиболее точную, а изменяя выбор узлов x_a, x_b и x_c, находят оптимальные значения коэффициентов A и B для выбранной аппроксимирующей формулы.

Составление программ для решения этой задачи с помощью микрокалькулятора существенно упрощается при представлении аппроксимирующих выражений с двумя коэффициентами обобщенной формулой

$$\varphi(z) = A^{\circ}z + B^{\circ}, \qquad (4.14)$$

связанной с формулами табл. 10 указанными в ней соотношениями, соответствующими преобразованиям функциональных шкал аргументов и их функций.*

Составив для двух узлов x_a и x_c заданной табличной модели систему обобщенных уравнений

$$\varphi_a = A^{\circ}z_a + B^{\circ};\ \varphi_c = A^{\circ}z_c + B^{\circ},$$

несложно найти ее решение относительно коэффициентов

$$A^{\circ} = (\varphi_c - \varphi_a)/(z_c - z_a),\ B^{\circ} = \varphi_a - A^{\circ}z_a. \qquad (4.15)$$

Вычислив по значениям коэффициентов величину $\varphi(z_b) = A^{\circ}z_b + B^{\circ}$, находят соответствующее значение $f(x_b)$ и погрешность аппроксимации $\Delta_b = f(x_b) - y_b$.

При реализации этих соотношений на входном языке ЯМК34 целесообразно выделить регистры *4*, *5*, *6* и *7*, *8*, *9* для хранения исходных и преобразуемых значений соответственно x_a, x_b, x_c и y_a, y_b, y_c, регистр *3* для постоянного хранения исходного значения y_b, регистры A и B — для хранения вычисленных значений обобщенных коэффициентов A° и B°. Вычисление этих коэффициентов по формулам (4.15) и значения $\varphi(z_b)$ реализуются фрагментом

ИП9 ИП7 — ИП6 ИП4 — ÷ ПА ИП7 ИПА
ИП7 × — ПВ ИП5 ИПА × + ⋯ ,

а вычисление логарифма значений x_i или y_i обеспечивается фрагментом

ПО П1 3 П2 КИПО ln КП1 L2 A_1 B/O ... ,

где A_1 — адрес перехода к оператору КИПО. Аналогичный фрагмент при замене оператора ln оператором $1/x$ обеспечивает вычисление обратных значений x_i или y_i при занесении (как и при логарифмировании) в регистры *0* и *1* чисел 7 или 10 соответственно.

---

* Некоторые из указанных в табл. 10 аппроксимирующих формул не могут быть представлены в обобщенной форме при отрицательных значениях аргумента или функции. В этом случае следует изменить начало отсчета аргумента или функции, добавив постоянное слагаемое.

Организовав переходы в программе, обеспечивающие выполнение требуемой последовательности операций, совместим в одной программе вычисление обобщенных коэффициентов и погрешности аппроксимации для всех функций, указанных в табл. 10.

Программа 29/34. Вычисление обобщенных коэффициентов и погрешности аппроксимирующих функций (табл. 10) по трем узлам табличной модели

7	ПП	63	ПП	42	ИП3	—	С/П	БП	00
7	ПП	75	БП	03	ПП	73	ПП	42	$1/x$
БП	05	7	ПП	63	БП	15	7	ПП	75
БП	15	7	ПП	63	ПП	61	ПП	42	e^x
БП	05	ИП9	ИП7	—	ИП6	ИП4	—	÷	ПА
ИП7	ИПА	ИП4	×	—	ПВ	ИП5	ИПА	×	+
В/О	1	0	П0	П1	3	П2	КИП0	ln	КП1
L2	67	В/О	1	0	П0	П1	3	П2	КИП0
$1/x$	КП1	L2	79	В/О					

Инструкция: $x_a = P4$, $x_b = P5$, $x_c = P6$, $y_a = P7$, $y_b = P8 = P3$, $y_c = P9$ (содержимое регистров *4*, ... , *7*, *9* необходимо восстанавливать, если оно изменилось, перед повторными пусками программы); для логарифмической функции $f(x) = A \ln x + B$ нажать клавиши В/О С/П, для линейной функции $f(x) = Ax + B$ — клавиши БП 03 С/П, для гиперболической $f(x) = A/x + B$ — клавиши БП 10 С/П, для дробно-рациональной $f(x) = 1/(Ax + B)$ — клавиши БП 15 С/П, для обратной логарифмической $f(x) = 1/(A \ln x + B)$ — клавиши БП 22 С/П, для дробно-рациональной $f(x) = x/(Ax + B)$ — клавиши БП 27 С/П, для показательной $f(x) = BA^x$ — клавиши БП 35 С/П, для степенной $f(x) = Bx^A$ — клавиши БП 32 С/П; результат: $PX = \Delta_b$, $PA = A^\circ$, $PB = B^\circ$ (для преобразования обобщенных параметров показательной и степенной функций достаточно использовать оператор e^x).

По этой программе для отсчетов $y_a = 0{,}5$; $y_b = 1{,}6$; $y_c = 7{,}5$ в узлах $x_a = 0{,}2$; $x_b = 0{,}8$; $x_c = 1{,}6$ получим погрешность аппроксимации (в порядке перечисления аппроксимирующих функций в табл. 10) $\Delta_b = 3{,}57$; 1,9; 4,9; —0,767; —0,276; 0,90; —0,004; 1,44. Следовательно, наиболее точно аппроксимируется таблично заданная функция показательной функцией, однако при учете бóльшего числа узлов этот вывод следует уточнить.

С помощью программы 29/34 можно подобрать оптимальные значения коэффициентов аппроксимирующей функции, но для проверки погрешности аппроксимации во всех узлах табличной модели необходимо многократно выполнять программу. Для более эффективного решения задачи оптимизации погрешностей требуется хранить аппроксимируемую табличную модель в памяти микрокалькулятора.

При постоянном шаге h табличной модели следует нормировать x_i к целочисленным значениям (номерам) $i = (x_i - x_1)/h + 1 = 1, 2, \ldots, m$. В этом случае достаточно хранить в памяти номера

трех рассматриваемых узлов ($a < b < c$), и количество регистров для хранения отсчетов y_i табличной модели соответственно увеличится. Целесообразно также уменьшить по модулю погрешность Δ_b, увеличив соответственно погрешности Δ_a и Δ_c на каждом шаге оптимизации погрешностей по трем узлам. В простейшем случае для этой цели после вычисления Δ_b следует найти коэффициенты аппроксимирующей функции при значениях $y_a = y_a - \Delta_b/2$ и $y_c = y_c - \Delta_b/2$. С учетом этих условий выполнение каждого шага аппроксимации соответствует следующему алгоритму:

1. Выбрать номера трех узлов ($a < b < c$) табличной модели.
2. Нормировать значения x_i и вычислить нормированные коэффициенты $A_н$ и $B_н$ аппроксимирующей функции.
3. Вычислить погрешность $\Delta_b = f(b) - y_b$.
4. Принять $y_a = y_a - \Delta_b/2$, $y_c = y_c - \Delta_b/2$.
5. Вычислить нормированные значения коэффициентов $A_н$ и $B_н$.
6. Вычислить погрешности аппроксимации $\Delta_t = f(t) - y_i$ для всех узлов табличной модели.

Повторяя подобные шаги при различном выборе узлов и сравнивая погрешности аппроксимации, следует выбрать оптимальные коэффициенты $A_н$ и $B_н$ и, денормируя их, вычислить оптимальные значения A и B.

При реализации этого алгоритма на входном языке ЯМК34 удобно выбрать регистры A, B, C, 9 и $Д$ для хранения соответственно номеров a, b, c узлов табличной модели и вычисляемых коэффициентов A и B. Регистр O целесообразно выделить для временного хранения приращений $-\Delta_b/2$ и адресов регистров памяти при косвенном вызове отсчетов y_i, размещаемых в порядке возрастания их номеров в остальных регистрах $1 \ldots 8$. Таким образом, рассматриваемый алгоритм реализуем на входном языке ЯМК34 для табличных моделей с числом $m \leqslant 8$ равноотстоящих узлов.

Выбор аппроксимирующей функции, реализуемый с помощью программы 29/34, в данном случае неприемлем вследствие ограниченного ресурса памяти. Поэтому учтем, что нормированные коэффициенты $A_н$ и $B_н$ можно вычислить в соответствии с обобщенными формулами (4.31) при преобразовании соответствующих величин согласно данным табл. 10. Для этого реализуем вычисление нормированных коэффициентов фрагментом

```
КИПС O2   КИПА O2   ПД  — ИПС O1 ИПА O1
—    ÷    П9    ИПД ИПА O1 ИП9 × —    O4
ПД   ИП9  O3    П9  ... ,
```

где одноместные операторы O_i предназначены для преобразования операндов в соответствии с данными табл. 10 для каждой из аппроксимирующих функций. Так, при замене всех операторов O_i операторами НОП по этому фрагменту вычисляются нормированные коэффициенты линейной функции, а для вычисления нормированных коэффициентов показательной функции операторы O_1 и O_2 должны быть заменены соответственно операторами НОП и ln, а O_3 и O_4 — операторами e^x.

Последовательное вычисление погрешностей $\Delta_i = y_i - f(x_i)$ во всех узлах табличной модели реализуется фрагментом

$m+1$ П0 КИП0 ИП0 $x\neq 0$ A_1 ИП9 × ИПД +
O_2 — С/П БП A_2 ...,

где A_1— адрес перехода к выполнению следующей части программы при окончании цикла; A_2— адрес перехода к оператору КИП0, символом $m+1$ обозначен оператор набора числа $m+1$.

Денормирование коэффициентов $A^0_н$ согласно обобщенной формуле (4.15) обеспечивается выражениями

$$A^0 = A^0_н(c' - a')/(z_c - z_a),\ B^0 = B^0_н + A^0 z_a - A^0_н a',$$

где c' и a'— номера узлов, преобразованные согласно данным табл. 10.

Возможность введения поправки $\Delta_0 = \Delta_a = \Delta_b$ при реализации рассматриваемого алгоритма обеспечивается в программе со «сменными» операторами.

Программа 30/34. Вычисление погрешностей аппроксимации и коэффициентов аппроксимирующих функций (табл. 10) для табличных моделей с числом 8 равноотстоящих узлов

Cx	П0	ПП	33	ИПВ	O_1	ИП9	×	+	O_3
КИПВ	—	2	/—/	÷	П0	ПП	33	$m+1$	П0
КИП0	ИП0	$x\neq 0$	58	ИП9	×	ИПД	+	O_3	—
С/П	БП	20	ИП0	КИПС	+	O_2	ИП0	КИПА	+
O_2	ПД	—	ИПС	O_1	ИПА	O_1	—	÷	П9
ИПД	ИПА	O_1	ИП9	×	—	ПД	В/О	Cx	С/П
O_1	XY	O_1	П0	—	$1/x$	ИПС	O_1	ИПА	O_1
—	×	ИП9	×	П9	Bx	ИПА	O_1	×	ИПД
XY	—	ИП9	ИП0	×	+	O_5	ПД	NПД	O_4
П9	С/П								

Инструкция: заменить в программе символ $m+1$ оператором набора числа $m+1$; для линейной функции заменить все символы O_i операторами НОП; для логарифмической символы O_1 и остальные O_i— соответственно операторами ln и НОП; для гиперболической — соответственно операторами $1/x$ и НОП; для обратной линейной символы O_2 и O_3— операторами $1/x$ и остальные O_i— операторами НОП; для обратной логарифмической O_1— операторами ln, O_2 и O_3— операторами $1/x$ и остальные O_i— операторами НОП; для показательной функции O_1 и O_2— соответственно операторами НОП и ln и остальные O_i — операторами e^x; для степенной O_1 и O_2 — операторами ln, O_4 — операторами НОП, остальные O_i — операторами e^x; $y_1 = P1$, $y_2 = P2$, ..., $y_m = Pm$, $a = PA$, $b = PB$, $c = PC$ В/О С/П (для начального смещения y_a и y_c ввести поправку Δ_0 в регистр O и нажать клавиши БП 0 2 С/П) $PX = \Delta_m$ С/П $PX = \Delta_{m-1}$ ... С/П $PX = \Delta_1$ С/П $PX = 0$ ($P9 = A^0_н$, $PД = B^0_н$) $x_a = PY$, $x_c = PX$ С/П $PX = P9 = A$, $PY = PД = B$.

После выбора аппроксимирующей функции эту программу можно сократить, устранив операторы НОП и соответственно уменьшив адреса переходов. Можно также не вводить фрагмент программы, обеспечивающий денормирование коэффициентов и, выполнять эту операцию в обычном режиме.

Методику оптимизации погрешностей после выбора вида аппроксимирующей формулы с помощью программы 30/34 рассмотрим на примере табличной модели $y_1 = 0{,}5$; $y_2 = 0{,}7$; $y_3 = 1$; $y_4 = 1{,}6$; $y_5 = 2{,}5$; $y_6 = 3{,}5$; $y_7 = 5$; $y_8 = 7{,}5$ с шагом $h = 0{,}2$ и $x_1 = 0{,}2$.

Предположим, что для аппроксимации табличной модели выбрана (например, с помощью программы 29/34) показательная функция и необходимо найти оптимальные значения ее коэффициентов. В этом случае целесообразно сократить программу 30/34, представив ее для $m = 8$ в следующей записи:

Cx	П0	ПП	32	ИПВ	ИП9	×	+	e^x	КИПВ
—	2	/—/	÷	П0	ПП	32	9	П0	КИП0
ИП0	$x \neq 0$	54	ИП9	×	ИПД	+	e^x	—	С/П
БП	19	ИП0	КИПС	+	ln	ИП0	КИПА	+	ln
ПД	—	ИПС	ИПА	—	÷	П9	ИПД	ИПА	ИП9
×	—	ПД	В/0	Cx	С/П	ХУ	П0	—	$1/x$
ИПС	ИПА	—	×	ИП9	×	П9	Вх	ИПА	×
ИПД	ХУ	—	ИП9	ИП0	×	+	e^x	ПД	ИП9
e^x	П9	С/П							

Для равномерного отклонения по модулю аппроксимирующей функции после ввода в числовую память отсчетов y_i табличной модели введем номера узлов $1 = PA$, $4 = PB$, $8 = PC$ и по составленной программе получим (в порядке уменьшения номеров узлов) $\Delta_i = -0{,}0020$; $-0{,}0980$; $0{,}0356$; $0{,}1458$; $0{,}0002$; $-0{,}0872$; $-0{,}0388$; $-0{,}0020$. Изменяя выбор узлов и поправку Δ_0, находим наиболее равномерное распределение отклонений $\Delta_i = 0{,}0147$; $-0{,}1119$; $0{,}0089$; $0{,}1158$; $-0{,}0282$; $-0{,}1119$; $-0{,}0593$; $-0{,}0186$ при $a = 3$, $b = 5$, $c = 7$, $\Delta_0 = 0{,}03$ и нормированные обобщенные коэффициенты $A_н^0 = 0{,}38137158$; $B_н^0 = -1{,}038026$. После денормирования (при вводе $x_a = 0{,}6$; $x_c = 1{,}4$) получаем $A = 6{,}7319032$ и $B = 0{,}35415511$.

Если требуется минимальная погрешность аппроксимации в определенной области аргумента табличной модели, то узлы с номерами a, b и c следует выбирать в этой области. Так, при необходимости наилучшей аппроксимации рассматриваемой табличной модели в начальной области аргумента следует принять $a = 1$, $b = 2$, $c = 3$. В этом случае по сокращенному варианту программы 30/34 получим $\Delta_i = 1{,}8137$; $0{,}9856$; $0{,}6665$; $0{,}5000$; $0{,}1883$; $0{,}0036$; $-0{,}0033$; $0{,}0036$ и коэффициенты $A_н^0 = 0{,}3483598$; $B_н^0 = -1{,}0486392$.

В тех случаях, когда не требуется повышенная точность аппроксимации на определенных интервалах аргумента и аппроксимирующая функция должна по возможности равномерно приближать аппроксимируемую табличную зависимость, желательно полностью автоматизировать процесс оптимизации погрешностей аппроксимации. Для этого следует выбрать способ определения оптимальных коэффициентов выбранной аппроксимирующей формулы. Остановимся на методе минимизации модуля максимального отклонения аппроксимирующей функции от заданных табличных значений. Так как рассматриваемые аппроксимирующие функции пригодны для гладких зависимостей, то достаточно выбрать в заданном интервале не более шести — семи равноотстоящих узлов.

Минимальное отклонение обобщенной функции $\varphi(z) = A_н^0 z + B_н^0$ от табличных значений φ_i при трех узлах аппроксимации ($x_a < x_b < x_c$) соответствует (рис. 28) системе обобщенных уравнений

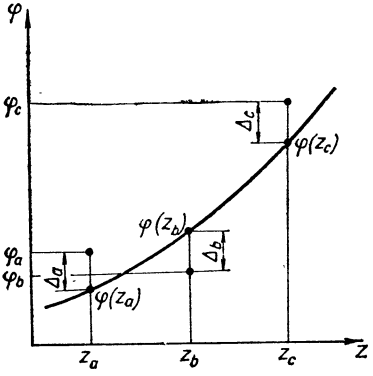


Рис. 28

$$\varphi_a - \varphi(z_a) = \varphi(z_b) - \varphi_b,$$
$$\varphi_c - \varphi(z_c) = \varphi(z_b) - \varphi_b.$$

Решение этой системы уравнений:

$$A^0 = (\varphi_c - \varphi_a)/(z_c - z_a),$$
$$B^0 = (\varphi_b + \varphi_c - A^0(z_b + z_c))/2$$

или, при нормировании значений x_i к номерам $i = (x_i - x_1)/h + 1 = 1, 2, \ldots, m$,

$$A_н^0 = (\varphi_c - \varphi_a)/(c' - a'),$$
$$B_н^0 = (\varphi_b + \varphi_c - A_н^0(b' + c'))/2. \quad (4.16)$$

Связь обобщенных величин (включая преобразованные в общем случае номера i') с коэффициентами и переменными аппроксимирующих функций указана в табл. 10.

Полная автоматизация процесса минимизации максимального отклонения этих функций в равноотстоящих узлах табличной модели

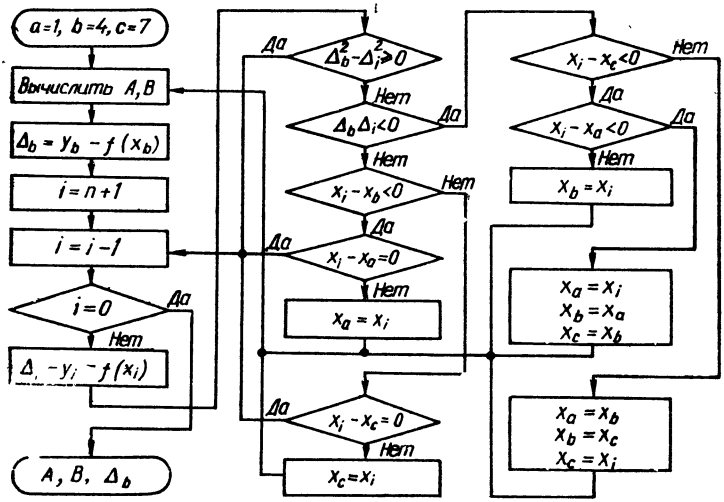


Рис. 29

обеспечивается при выборе схемы алгоритма, показанной на рис. 29. Согласно этому алгоритму по произвольно выбранным узлам с номерами a, b и c табличной модели (целесообразно выбирать крайние и средний узлы или близкие к ним) определяются коэффициенты $A_н$ и $B_н$ аппроксимирующей функции и погрешность аппроксимации $\Delta_b = f(x_b) - y_b$, после чего вычисляются погрешности Δ_i (начиная

с последнего узла), которые сравниваются со значением Δ_b. Если $|\Delta_b| < |\Delta_i|$, то сравниваются знаки отклонений, если они совпадают и узел i лежит между узлами a и c, то узел b заменяется узлом i, в противном случае узлом i заменяется узел a. Если же знаки отклонений противоположны, то узлом i заменяется ближайший узел a или c. После этого цикл вычислений коэффициентов повторяется до достижения минимального значения Δ_b, которому соответствуют оптимальные значения коэффициентов.

Для программной реализации этого алгоритма на входном языке ЯМК34 целесообразно выделить регистры A, B, C для хранения номеров узлов a, b и c, регистры *8*, *9* и *Д* для хранения соответственно коэффициентов $A^0_н$, $B^0_н$ и поправки Δ_b, а регистр *О* использовать в качестве адресного для косвенного вызова значений y_i по их номерам i. В этом случае для хранения отсчетов табличной модели остается семь регистров *1* ... *7*, что обеспечивает аппроксимацию табулированных зависимостей по $m \leqslant 7$ равноотстоящих узлов.

Трудности реализации рассматриваемого алгоритма связаны в основном с недостаточной емкостью программной памяти и даже при отказе от денормирования коэффициентов для размещения программы в памяти приходится использовать все особенности входного языка.

Наиболее громоздок блок перебора узлов табличной модели по вычисляемым значениям Δ_i, не связанный непосредственно с вычислением этих значений. При минимизации числа операторов этот блок можно представить в следующей записи:

↑	x^2	ИПД	x^2	XY	—	$x<0$	A_1	XY	ИПД
×	$x\geqslant 0$	(26)	ИП0	ИПВ	—	$x<0$	(31)	ИПА	ИП0
ПА	—	$x\neq 0$	A_1	БП	A_2	ИПС	ИП0	ПС	БП
(21)	ИП0	ИПС	—	$x\geqslant 0$	(42)	ИПВ	ПА	ИПС	ПВ
БП	(27)	ИП0	ИПА	—	$x<0$	(53)	ИПВ	ПС	ИПА
ПВ	БП	(19)	ИП0	ПВ	БП	A_2	...,		

где A_1— адрес перехода к оператору КИПО в предыдущем фрагменте для вычисления Δ_i (подобном соответствующему фрагменту в программе 30/34), адрес A_2 обеспечивает переход к началу блока вычисления коэффициентов аппроксимирующей функции, в скобках указаны адреса переходов в блоке, отсчитываемые относительно его начального оператора.

Предыдущий фрагмент вычисления Δ_i можно сократить на один оператор, поместив оператор прекращения вычислений в начало программы и использовав для перехода к нему после окончания вычислений Δ_i в цикле оператор косвенного условного перехода К$x \neq 00$. В этом случае операторы безусловного перехода БП A_2 в блоке перебора узлов можно заменить оператором В/О (который при отсутствии переходов к подпрограммам передает управление второму шагу программы) и длина этого блока сокращается еще на два шага.

В остающихся свободными после ввода этого блока ячейках программной памяти удается разместить фрагмент со «сменными» опера-

торами для вычисления отклонений и коэффициентов линейной, логарифмической и гиперболической функции или фрагмент со «сменными» операторами, преобразующими значения линейной, обратной линейной и показательной функций. Однако сократить программу еще на два шага, чтобы обеспечить оптимизацию погрешностей всех аппроксимирующих функций, указанных в табл. 10, не удается. Если же вводить вместо y_i значения $\ln y_i$ или $1/y_i$, то можно оптимизировать отклонения обобщенной функции, но в этом случае погрешности преобразованных функций будут распределяться неравномерно.

Программа 31/34. Оптимизация погрешностей линейной, логарифмической и гиперболической аппроксимирующих функций для табличных моделей с числом $m \leqslant 7$ равноотстоящих узлов

С/П	КИПВ	КИПС	+	КИПА	КИПС	—	ИПА	O_1	ИПС
O_1	—	÷	П8	ИПВ	O_1	ИПС	O_1	+	×
—	2	÷	П9	ИП8	ИПВ	O_1	×	+	КИПВ
—	ПД	$m+1$	П0	КИП0	ИП0	К$x \neq 0$	O_1	ИП8	×
ИП9	+	—	↑	x^2	ИПД	x^2	XY	—	$x<0$
34	XY	ИПД	×	$x \geqslant 0$	73	ИП0	ИПВ	—	$x<0$
68	ИПА	ИП0	ПА	—	$x \neq 0$	34	В/О	ИПС	ИП0
ПС	БП	64	ИП0	ИПС	—	$x \geqslant 0$	84	ИПВ	ПА
ИПС	ПВ	БП	69	ИП0	ИПА	—	$x<0$	95	ИПВ
ПС	ИПА	ПВ	БП	62	ИП0	ПВ	В/О		

Инструкция: заменить в программе символ $m+1$ оператором набора числа $m+1$ и символы O_1 операторами НОП, ln или $1/x$ соответственно для линейной, логарифмической или гиперболической функций; $y_i = Pi$, выбранные номера узлов $a = PA$, $b = PB$, $c = PC$ В/О С/П С/П (клавишу С/П нажать дважды) $PA = a_{\text{opt}}$, $PB = b_{\text{opt}}$, $PC = c_{\text{opt}}$, $PД = \Delta_{\max}$, $P8 = A^0_{\text{н}}$, $P9 = B^0_{\text{н}}$ (денормировать коэффициенты в обычном режиме).

Согласно выражениям (4.16) при подстановке найденных по программе оптимальных номеров узлов и соответствующих значений x_a, x_b и x_c вычисленные коэффициенты денормируются по обобщенным формулам

$$A^0 = A^0_{\text{н}}(c' - a')/(z_c - z_a),\ B^0 = B^0_{\text{н}} + (A^0_{\text{н}}(c' + b') - A^0(z_c + z_b))/2,$$

из которых для линейной функции следует

$$A = A^0_{\text{н}}/h,\ B = B^0_{\text{н}} + A(h - x_1),$$

для логарифмической —

$$A = A^0_{\text{н}} \ln(c/a)/\ln(x_c/x_a),\ B = B^0_{\text{н}} + (A^0_{\text{н}} \ln cb - A \ln x_c x_b)/2$$

и для гиперболической —

$$A = A^0_{\text{н}} x_a x_c/hac,\ B = B^0_{\text{н}} + A^0_{\text{н}}(x_b + x_c)/(2x_b x_c) - A(b + c)/2bc.$$

Например, для табличной модели $y_1 = 0{,}7$; $y_2 = 0{,}5$; $y_3 = 1$; $y_4 = 1{,}2$; $y_5 = 1{,}6$; $y_6 = 1{,}4$; $y_7 = 2$ с шагом $h = 0{,}2$ и $x_1 = 0{,}2$ по программе 28/34 для линейной функции получим (при $1 = PA$, $4 = PB$, $7 = PC$) $a_{opt} = 2$, $b_{opt} = 5$, $c_{opt} = 6$, $\Delta_{max} = -0{,}2125$; $A_н^0 = 0{,}225$; $B_н^0 = 0{,}2625$ или, после денормирования, $f(x) = 1{,}125x + 0{,}2625$.

Заменив первые четыре строки программы 31/34 фрагментом

С/П	КИПВ	O_2	КИПС	O_2	+	КИПА	O_2	КИПС	O_2
—	ИПА	ИПС	—	÷	П8	ИПВ	ИПС	+	×
—	2	÷	П9	ИП8	ИПВ	×	+	КИПВ	O_3
—	ПД	8	П0	КИП0	O_3	ИП0	$Kx \neq 00$	ИП8	×

с заменой символов O_2 и O_3 операторами НОП для линейной функции, операторами $1/x$ для обратной линейной функции и соответственно операторами ln и e^x для показательной функции, получим возможность оптимизировать еще три аппроксимирующие функции.

Рассмотренные способы применимы и для подбора оптимальных коэффициентов других аппроксимирующих функций, не указанных в табл. 10. Если представленная табличной моделью функциональная зависимость явно не монотонна, то для аппроксимации обычно используют многочлен, степень которого равна числу экстремумов аппроксимируемой зависимости. Для подбора оптимальных коэффициентов такого многочлена используют алгоритмы, подобные рассмотренным, или метод наименьших квадратов, описанный в гл. 7.

Среди других методов аппроксимации часто оказывается эффективным приближение исследуемой зависимости ее теоретической моделью с уточнением параметров [13], например, методами неопределенных коэффициентов или минимизации наибольших отклонений. Специальные функции, операторы вычисления которых отсутствуют во входном языке микрокалькулятора, аппроксимируют различными способами, но часто наиболее эффективным оказывается удачный подбор несложной эмпирической формулы. Примером может служить формула (4.3), обеспечивающая вычисление $\operatorname{arctg} x$ с гораздо более высокой точностью, чем многие другие методы [11] аппроксимации этой функции.

Для аппроксимации табличных или графических моделей функциональных зависимостей аналитическими выражениями иногда удобно использовать интерполирующие функции (например, интерполирующие многочлены, вычисляемые по рассмотренным программам или программам 155/21 и 156/21 приложения). В этом случае выбор отсчетов функции для интерполирования изменяют до тех пор, пока погрешность аппроксимации превышает допустимую.

Глава 5

РЕШЕНИЕ УРАВНЕНИЙ И СИСТЕМ УРАВНЕНИЙ

1. ВЫЧИСЛЕНИЕ ВЕЩЕСТВЕННЫХ КОРНЕЙ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Во многих инженерных задачах приходится решать нелинейные уравнения $f(x) = 0$, в которых функцию $f(x)$ называют *алгебраической*, если для ее вычисления необходимы только арифметические операции и возведение аргумента в рациональные степени, и *трансцендентной* в противном случае. Множество значений аргумента x, при которых уравнение обращается в тождество, называют *решением* уравнения, элементы x_k этого множества — корнями, а значение функции $f(x)$ — невязкой уравнения, равной нулю лишь при $x = x_k$.

Вычисление корней нелинейных уравнений точными методами по формулам с заранее известным числом операций возможно лишь для некоторых функций. Поэтому основными являются приближенные численные методы поиска корней, разбиваемого на три этапа: определение начальной области аргумента, в которой находятся корни, определение ее частей, содержащих по одному корню (*отделение* корней), и сокращение этих частей до размеров, обеспечивающих требуемую точность определения корней (*уточнение* корней).

В большинстве практических задач требуется найти только вещественные корни нелинейных уравнений. Обычно в этих случаях начальная область нахождения корней определяется физическими условиями задачи. В противном случае ее приближенно определяют методами математического анализа функции $f(x)$ или с помощью пробных вычислений этой функции для ряда значений аргумента. В последнем случае совмещают этапы определения области нахождения корней и отделения корней, так как на границах интервала нахождения корня значения функции противоположны по знаку.

Вычислив значения функции в некотором интервале аргумента, можно оценить ее поведение и количество корней, но такая методика связана со значительными затратами времени, особенно при необходимости оценки влияния коэффициентов функции $f(x)$ и на этапе отделения корней. Обычно более информативными оказываются результаты анализа функции, основанные на асимптотических оценках значений функции, определении ее особых точек и вычислении значений функции в некоторых характерных точках (например, при $x = 0$).

Рассмотрим в качестве примера уравнение $f(x) \equiv 2^x - 5x - 3 = 0$, для которого несложно определить, что $f(-\infty) \to \infty$, $f(0) = -2$ и $f(\infty) \to \infty$. Приравняв нулю первую производную $f'(x) = 2^x \ln 2 - 5$, получим, что в точке $x = \ln(5/\ln 2)/\ln 2 = 2{,}8507$ функция имеет минимальное значение $f(x) = -10{,}04$ и, следовательно, не более двух корней. Определив, например, что в точках $x = \pm 5$ знак функции положителен, устанавливаем, что корни расположены в интервалах $-5 < x_1 < 0$ и $0 < x_2 < 5$ (рис. 30,*а*).

В более сложных случаях приходится анализировать зависимость числа и расположения корней от значений коэффициентов левой части уравнения. Например, левая часть уравнения $f(x) \equiv \operatorname{tg} x - Bx + C = 0$ имеет особые точки с бесконечным значением при $x = \pi/2 + q\pi$ (где q — целое число), причем при стремлении аргумента к этим точкам слева функция стремится к положительным, а при стремлении справа — к отрицательным бесконечным значениям. Поэтому на каждом интервале $]-\pi/2 + q\pi, \pi/2 + q\pi[$ находится нечетное количество корней. При $x = q\pi$ функция $f(x) = C - Bq\pi/2$ и, если $C > Bq\pi/2$, то один из корней лежит в левой половине q-го интервала, а в противном случае — в правом.

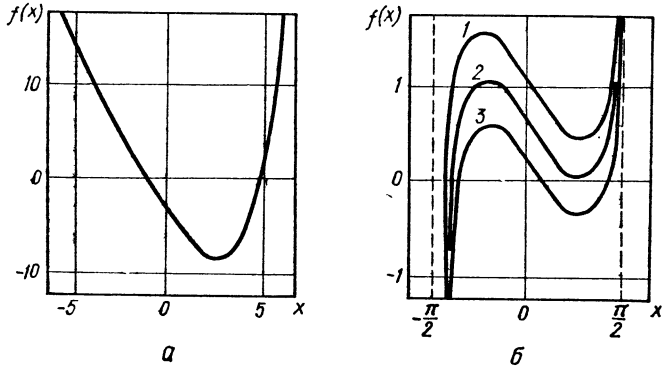


Рис. 30

Часто требуется определить все корни уравнения в заданном интервале аргумента. Например, для рассматриваемого уравнения при $B = 2$, $C = 1$ и $q = 0$ корень находится в левой части интервала $]-\pi/2, \pi/2[$ (кривая *1* на рис. 30,*б*). Однако при других значениях C корни могут появиться и в правой части рассматриваемого интервала, например, два корня при $C = 0{,}2$ (кривая *3* на рис. 30,*б*) или два совпадающих корня (их называют корнем кратности 2) при $C = \pi/2 - 1$ (кривая *2* на рис. 30,*б*). В *кратных* корнях первая производная функции $f'(x) = 0$, причем в корне четной кратности функция не пересекает ось абсцисс, а в корне нечетной кратности — «перегибается», пересекая ось абсцисс.

Иногда удается отделить корни, отыскивая решение уравнения $f'(x) = 0$, соответствующее значениям аргумента для экстремальных значений функции, между которыми расположены корни уравнения $f(x) = 0$, и перегибов функции. Однако найти решение уравнения для производной часто не менее сложно, чем решение исходного уравнения. Кроме того, подобный способ применим только для дифференцируемых функций, а возможность появления кратных корней и в этом случае требует дополнительного анализа*. Поэтому на этапах определе-

* Большинство рассматриваемых в этом разделе методов решения нелинейных уравнений применимо для поиска только *простых* (некратных) корней. Это не является существенным недостатком, так как в практических задачах уравнения с кратными корнями встречаются редко.

ния начального расположения корней и их отделения обычно приходится комбинировать различные методы оценки изменений левой части нелинейного уравнения в заданном интервале аргумента.

Уточнение корней сводится к сокращению интервала нахождения корня (контролируемого по противоположности знаков функции на границах сокращенного интервала) до тех пор, пока ширина Δ_i сокращаемого интервала превышает наперед заданное малое число ε, определяющее требуемую точность вычисления корня. Достижение требуемой точности в этом случае соответствует выполнению неравенства $\Delta_i \leqslant \varepsilon$, которое и служит условием прекращения вычислений. Однако конкретная формулировка этого условия зависит от алгоритма уточнения корня и возможности достижения заданной точности.

Если в процессе уточнения корня на каждой i-й итерации вычисляется граничное значение x_i, для которого значение функции $f(x_i)$ противоположно по знаку значению $f(x_{i-1})$, то условие прекращения вычислений отображается неравенством $|x_i - x_{i-1}| \leqslant \varepsilon$, которое при программной реализации удобнее заменить неравенством $(x_i - x_{i-1})^2 \leqslant \varepsilon^2$. В этом случае при хранении значений x_{i-1} и ε^2 соответственно в регистрах N и M памяти и вычисленного значения x_i в регистре X условие выхода из итерационного цикла на входном языке ЯМК34 можно реализовать фрагментом

$$\cdots \text{ ИПN XY ПN} - x^2 \text{ ИПМ} - x < 0 \; A \ldots,$$

а на входном языке ЯМК21 — фрагментом

$$\cdots \uparrow \text{FN XY PN} - x^2 \uparrow \text{FM} - x < 0 \; A \ldots,$$

где A — указатель адреса начального оператора цикла.

Подобную реализацию прекращения вычислений можно использовать и в тех случаях, когда на каждой итерации вычисляется ширина сокращенного интервала $|\Delta_i| = |x_i - x_{i-1}|$ без операции вычитания (например, делением ширины сокращаемого интервала). Для этого вместо начальных операторов приведенных фрагментов до оператора x^2 вводятся операторы вызова или вычисления Δ_i. Если на каждой итерации вычисляется не граничное значение x_i, а приращение аргумента $\Delta x_i = x_i - x_{i-1}$, то условие выхода из цикла реализуется фрагментом

$$\cdots \text{ ИПN} + \text{ ПN Вх} - x^2 \text{ ИПМ} - x < 0 \; A \cdots$$

или

$$\cdots \uparrow \text{FN} + \text{PN FM XY } x^2 - x \geqslant 0 \; A \ldots,$$

где A — указатель адреса начала цикла.

Величину ε^2 можно записывать непосредственно в текст программы при нехватке регистров памяти или прекращать вычисления при совпадении q цифр очередных значений x_i и x_{i-1}. Последний способ реализуется на входном языке ЯМК34 фрагментом

$$\cdots \uparrow \text{ ВП } s + \text{ Вх} - \text{ ИПN XY ПN} - x = 0 \; A \ldots,$$

где записываемое в программу число $s = 8 - q$; q — требуемое число цифр мантиссы результата после округления; $x_i = PX$, $x_{i-1} = PN$.

Для микрокалькуляторов первых выпусков с входным языком ЯМК21 этот способ реализуется фрагментом

$$\cdots \uparrow \text{ВП } s \text{ XY} + \text{XY} - \uparrow \text{FN XY PN} - x = 0\ A \ \ldots,$$

а на тех же микрокалькуляторах современных выпусков — фрагментом

$$\cdots \uparrow \text{ВП } s \text{ XY} - - \uparrow \text{FN XY PN} - x = 0\ A \cdots$$

Максимальная точность вычислений корня (без учета погрешностей вычисления) соответствует точному равенству значений x_{i-1} и x_i граничных значений сокращаемого интервала. Прекращение вычислений при выполнении этого равенства для $x_{i-1} = PN$, $x_i = PX$ на входном языке ЯМК34 реализуется фрагментом

$$\cdots \text{ИПN XY ПN} - x = 0\ A \ \ldots,$$

а на входном языке ЯМК21 — фрагментом

$$\cdots \uparrow \text{FN XY PN} - x = 0\ A \cdots$$

Однако эти фрагменты допустимо использовать лишь в том случае, когда имеется уверенность в возможности выполнения проверяемого условия. Так, его можно использовать, например, в программах с вычислением $x_i = x_{i-1} + \Delta x_i$ по приращению Δx_i, причем известно, что $|\Delta x_i| < |\Delta x_{i-1}|$ для любых значений i. В этом случае на входном языке ЯМК34 условие выхода из цикла вычислений отображается фрагментом

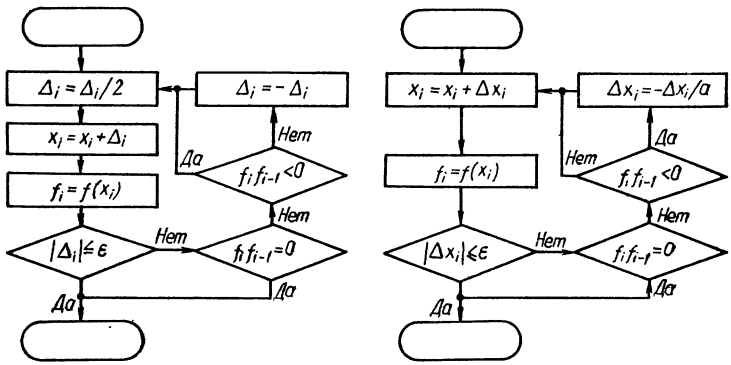
$$\cdots \uparrow \text{ИПN} + \text{ПN} - x = 0\ A \cdots$$

Рассматриваемый критерий неприменим для микрокалькуляторов с входным языком ЯМК21 поздних выпусков, у которых вычитание сколь угодно малого числа из значения x_i приводит к уменьшению этого значения на единицу последнего разряда. С учетом этого обстоятельства достаточно ограничиться сравнением результатов, округленных до семи знаков.

Удобные для применения программируемых микрокалькуляторов итерационные методы уточнения корня нелинейного уравнения можно условно разбить на две группы. К первой относятся методы с гарантированной сходимостью результата вычислений к искомому корню, не требующие вычисления производных и пригодные для решения нелинейных уравнений с недифференцируемой или даже разрывной левой частью. Ко второй группе относятся методы уточнения корня, применимые только для нелинейных уравнений с дифференцируемой левой частью. При использовании этих методов время вычислений относительно небольшое, но необходимо предварительное исследование условий сходимости результата к искомому корню.

Для вычислений на программируемых микрокалькуляторах основной интерес представляют методы уточнения корня, при программной реализации которых сохраняется достаточно большое число ячеек программной памяти, обеспечивающих возможность организации вычисления достаточно сложных функций в левой части решаемых

уравнений. Этим требованиям отвечает относящийся к первой группе метод половинного деления, один из возможных алгоритмов которого представлен схемой на рис. 31,*а*. В этом алгоритме на каждой итерации ширина интервала Δ_{i-1}, содержащего корень, делится пополам с выбором в качестве сокращенного интервала той половины, на границах которой знаки функции противоположны. Вычисления начинаются с задания нижней границы $x_н$, ширины Δ_0 начального интервала нахождения корня, числа ε, определяющего требуемую точность решения, и прекращаются при выполнении условия $|\Delta_i| \leqslant \varepsilon$, соответствующего вычислению корня x_i с предельной абсолютной погрешностью $|\Delta_i|$. На входном языке ЯМК34 этот алгоритм можно реализо-



а *б*

Рис. 31

вать следующей базовой (пригодной для уравнений различного вида) программой.

Программа 32/34. Вычисление корня нелинейного уравнения $f(x) = 0$ методом половинного деления

ИП7	С/П	ИП8	2	÷	П8	ИП7	+	П7	...		
ИП4	XY	П4	×	ИП5	ИП8	x^2	—	$x<0$	00		
ХУ	$x\neq 0$	00	$x<0$	02	ИП8	/—/	П8	БП	02		

Инструкция: заменить многоточие операторами вычисления (при $x = P7$) левой части уравнения, $x_н = P7$, $\Delta_0 = P8$, $f(x_н) = P4$ (можно ввести любое число, совпадающее по знаку с этим значением функции), $\varepsilon^2 = P5$ В/О С/П $PX = x_н$ С/П $PX = P7 = x_i$, $P8 = \Delta_i$, $P4 = f(x_i)$.

Контрольный пример: для вычисления корней уравнения $2^x - 5x - 3 = 0$ (см. рис. 30, *а*) в интервале $-5 < x < 0$ с предельной погрешностью $\varepsilon = 0{,}001$ заменим многоточие в программе 32/34 операторами 2 X^y XY 5 × — 3 — для вычисления левой части уравнения, вводим $x_н = -5 = P7$, $\Delta_0 = 5 = P8$, $1 = P4$, $\varepsilon^2 = 1 \cdot 10^{-6} = P5$ и получаем (время счета 2 мин 15 с) $x_i = -0{,}4534912$;

$\Delta_i = -0{,}00061$; $f(x_i) = -0{,}00227$; для вычисления корня в интервале $0 < x < 5$ вводим $0 = P7$, $5 = P8$, $-1 = P4$ и получаем $x_i = 4{,}7381593$; $\Delta_i = 0{,}00061$; $f(x_i) = -0{,}00206$.

В методе половинного деления ширина Δ_0 исходного интервала через n итераций сокращается до величины $\Delta_n = \Delta_0/2^n$. При условии прекращения вычислений $\Delta_n \leqslant \varepsilon$ можно записать $2^n \leqslant \Delta_0/\varepsilon$, откуда получаем число итераций, требуемое для вычисления корня с заданной точностью ε,

$$n \geqslant (\ln(\Delta_0/\varepsilon))/\ln 2 \qquad (5.1)$$

с округлением правой части неравенства до ближайшего большего целого числа. Следовательно, вычислив предварительно число n по выбранному числу ε в соответствии с этой формулой, можно сократить длину базовой программы метода половинного деления.

Программа 33/34. Вычисление корня нелинейного уравнения методом половинного деления с заданным числом n делений

```
...  ИП4 XY П4 ×   x<0 10 ИП8 /—/ П8
ИП8 2  ÷  П8 ИП7 +   П7 L0  00  С/П
```

Инструкция: заменить многоточие фрагментом вычисления левой части уравнения (при $x = P7$) и увеличить адрес условного перехода на число, на единицу меньшее числа шагов в этом фрагменте, $n = P0$, $x_н = P7$, $\Delta_0 = P8$, $f(x_н) = P4$ (можно вводить любое число, совпадающее по знаку со значением функции при $x = x_н$) В/О С/П $PX = P7 = x_n$, $P8 = \Delta_n$, $P4 = f(x_{n-1})$.

Контрольный пример: для вычисления корня уравнения $2^x - 5x - 3 = 0$ заменяем многоточие в программе фрагментом ИП7 2 X^y XY 5 × — 3 — и адрес 10 условного перехода на 18, вводим $n \geqslant \ln(5 \cdot 10^{-3})/\ln 2 \approx 13$ при $\varepsilon = 0{,}001$ и $x_н = -5$, $\Delta_0 = 5$, $1 = P4$ и получаем (время счета около 2 мин) $x_n = -0{,}4534912$; $\Delta_n = -0{,}00061$; $f(x_{n-1}) = -0{,}005$.

Приближение $x_i = x_{i-1} + \Delta_i$ перестает изменяться, когда Δ_i становится машинным нулем относительно x_{i-1}. Поэтому в методе половинного деления, как и в других методах уточнения корня с вычислением приращения аргумента, максимально достижимая точность вычисления корня ограничена ценой единицы последнего разряда мантиссы. Следовательно, на микрокалькуляторе с разрядностью мантиссы $r = 8$ следует принимать* $\varepsilon \geqslant 1 \cdot 10^{-7}/|x_{гр}|$, где $x_{гр}$ — меньшее по модулю граничное значение аргумента в интервале нахождения корня. В тех случаях, когда необходимо вычислить корень с максимальной точностью, можно воспользоваться следующей базовой программой, где вычисления прекращаются при равенстве приращения x_i машинному нулю относительно x_i.

---

* Оценка минимально допустимого значения ε по погрешностям вычислений в цикле рассмотрена в гл. 2.

Программа 34/34. Вычисление корня нелинейного уравнения методом половинного деления с максимальной точностью

ИП8 2 ÷ П8 ИП7 + П7 Вх — $x = 0$
13 ИП7 С/П ... ИП4 ХУ П4 × $x < 0$ 00
ИП8 /—/ В/О

Инструкция: заменить многоточие операторами вычисления (при $x = P7$) левой части уравнения, $x_н = P7$, $\Delta_0 = P8$, $r(x_н) = P4$ (можно вводить любое число, совпадающее по знаку со значением функции) В/О С/П $PX = P7 = x_m$, $P8 = \Delta_m$, $P4 = f(x_{m-1})$.

Контрольный пример: для вычисления корня уравнения $2^x - 5x - 3 = 0$ вводим $-5 = P7$, $5 = P8$, $1 = P4$ и получаем (время счета около 5 мин) $x_m = -0{,}45399633$; $\Delta_m = 0{,}2323 \cdot 10^{-9}$; $f(x_{m-1}) = -4 \cdot 10^{-7}$.

В тех случаях, когда известна только нижняя граница интервала корня или корни не отделены, возможно использование метода последовательного поиска. В основу метода положено вычисление на i-й итерации левой части $f(x)$ уравнения для ряда значений $x_i = x_{i-1} + \Delta x_i$ с постоянным или возрастающим шагом Δx_i до изменения знака $f(x)$, свидетельствующего о достижении границы интервала нахождения корня и выборе для следующей итерации $\Delta x_{i+1} = -\Delta x_i/a$, где $a > 1$. Вычисления прекращают по условию $|\Delta x_i| \leqslant \varepsilon$ (рис. 31, *б*), причем при разрядности мантиссы $r = 8$ по указанной выше причине следует выбирать $\varepsilon \geqslant 1 \cdot 10^{-7} |x_{гр}|$.

Программа 35/34. Вычисление корня нелинейного уравнения методом равномерного последовательного поиска

ИП7 С/П ИП8 ИП7 + П7 ... ИП4 ХУ П4
× $x \neq 0$ 00 $x < 0$ 02 ИП5 ИП8 x^2 — $x < 0$ ·
00 ИП8 /—/ a ÷ П8 БП 02

Инструкция: заменить в программе букву a оператором набора числа, бо́льшего единицы, многоточие — операторами вычисления (при $x = P7$) левой части уравнения, $x_н = P7$, $\Delta x_0 = P8$, $f(x_н) = P4$ (можно ввести любое число, совпадающее по знаку со значением функции), $\varepsilon^2 = P5$ В/О С/П $PX = P7 = x_н$ С/П $PX = P7 = x_i$, $P8 = \Delta x_i$, $P4 = f(x_i)$.

Контрольный пример: для вычисления корня уравнения $2^x - 5x - 3 = 0$ при $\varepsilon = 0{,}001$, $a = 5$ соответственно изменяем базовую программу, вводим $-5 = P7$, $1 = P8$, $1 = P4$, $10^{-6} = P5$ и получаем (время счета около 2 мин 50 с) $x_i = 0{,}45408$; $\Delta x_i = -0{,}00032$; $f(x_i) = 0{,}00038$.

Затраты времени на уточнение корней нелинейных уравнений с дифференцируемой левой частью можно уменьшить, воспользовавшись методами второй группы с предварительной оценкой сходимости итерационного процесса. Наиболее часто из этих методов используют метод касательных Ньютона, основанный на последовательном вычислении приближений

$$x_{i+1} = x_i - f(x_i)/f'(x_i), \quad i = 0, 1, 2, \ldots, \qquad (5.2)$$

где в качестве начального приближения x_0 выбирают то граничное значение аргумента в интервале нахождения корня, для которого знаки функции $f(x)$ и ее второй производной $f''(x)$ совпадают.

Метод Ньютона сходится, если во всем интервале нахождения корня производные $f'(x)$ и $f''(x)$ левой части уравнения отличны от нуля и не изменяются по знаку.

Непосредственно по формуле (5.2) можно составить следующую базовую программу для вычисления приближения x_i корня с $q = r - s$ верными знаками.

Программа 36/34. Вычисление корня нелинейного уравнения методом касательных Ньютона с точностью до q верных цифр

```
... П2  ... П4 ИП2 ÷  ИП7 XY — ↑
ВП s    +  Вх —  ИП7 XY  П7 — x = 0
00 ИП7 С/П
```

Инструкция: заменить многоточия фрагментами вычисления (при $x = P7$) $f'(x)$ и $f(x)$ соответственно, символ s — оператором набора числа $s = 8 - q \geqslant 1$ (для $r = 8$), $x_0 = P7$ В/О С/П $PX = P7 = x_{i+1}$, $P4 = f(x_i)$.

При $q = 7$ фрагмент программы ↑ ВП s + Вх целесообразно заменить операторами 1 + 1. Например, для вычисления корня уравнения $2^x - 5x - 3 = 0$ с первой производной $f'(x) = 2^x \ln 2 - 5$ при $q = 7$ в соответствии с базовой программой 36/34 можно составить рабочий вариант программы ($x_0 = P7$ В/О С/П $PX = P7 = x_{i+1}$, $P4 = f(x_i)$)

```
ИП7 2  X^y  2  ln  ×  5 — П2  ИП7
2   X^y  XY  5  ×   —  3 — П4  ИП2
÷   ИП7 XY  — 1  +  1 — ИП7 XY
П7  —  x = 0 00 ИП7 С/П
```

Для исходного интервала $-5 < x < 0$ знаки $f(-5)$ и $f''(-5)$ совпадают и при $-5 = P7$ по этой программе получим (время счета около 1 мин) $x_{i+1} = -0{,}4539964$; $f(x_i) = -1 \cdot 10^{-7}$.

Если на исходном интервале нахождения корня производная $f'(x)$ изменяется достаточно мало, то формулу (5.2) можно упростить, приняв

$$x_{i+1} = x_i - f(x_i)/f'(x_0), \quad i = 0, 1, 2, \ldots,$$

что приведет лишь к некоторому увеличению времени вычисления корня с требуемой точностью при предварительном вычислении значения $f'(x_0)$. Например, вычислив для уравнения $2^x - 5x - 3 = 0$ значение $f'(x_н) = 2^{-5} \ln 2 - 5 = -4{,}9783391$ и составив рабочую программу ($x_0 = P7$, $f'(x_0) = P8$ В/О С/П $PX = P7 = x_{i+1}$, $P4 = f(x_i)$) при $q = 7$

```
ИП7 2  X^y  XY  5  ×   —  3 — П4
ИП8 ÷  ИП7 XY  — 1  +  1 — ИП7
XY  П7 —  x = 0 00 ИП7 С/П
```

для интервала $-5 < x < 0$ при $x_0 = x_н = -5$ получим (время счета около 1 мин 50 с) $x_{i+1} = -0{,}4539964$; $f(x_i) = -1 \cdot 10^{-7}$.

Необходимость в вычислении производной по аналитическому выражению отпадает при численном дифференцировании согласно формуле

$$f'(x) \approx (f(x_i) - f(x_i - \Delta x_i))/\Delta x_i, \qquad (5.3)$$

где малость значения Δx_i можно обеспечить по условию $\delta = \Delta x_i / x_i \ll 1$.

При численном дифференцировании формулу (5.2) можно представить выражением

$$x_{i+1} = x_i (1 + f(x_i)\,\delta/(f(x_i - x_i\delta) - f(x_i)),$$

реализуемым при $q = 7$ следующей базовой программой.

Программа 37/34. Вычисление корня нелинейного уравнения методом касательных Ньютона с 7 верными цифрами при численном дифференцировании

ИП7	ПП	32	П4	ИП7	↑	ИП8	×	—	ПП
32	ИП4	—	ИП4	XY	÷	ИП8	×	1	0
+	9	—	ИП7	×	П7	Вх	—	$x=0$	00
ИП7	С/П	...	В/О						

Инструкция: заменить многоточие фрагментом вычисления левой части (при $x = P7$) уравнения, $x_0 = P7$, $\delta = P8$ В/О С/П $PX = P7 = x_{i+1}$, $P4 = f(x_i)$.

Приняв, например, $\delta = 1 \cdot 10^{-5}$, $x_0 = -5$ для уравнения $2^x - 5x - 3 = 0$, по этой программе получим (время счета около 1 мин) $x_{i+1} = -0{,}4539964$; $f(x_i) = -1 \cdot 10^{-7}$.

Для решения нелинейных уравнений с помощью микрокалькуляторов часто оказывается удобным метод простых итераций, основанный на приведении уравнения $f(x) = 0$ к виду $x = \varphi(x)$ и последовательном вычислении приближений

$$x_{i+1} = \varphi(x_i), \quad i = 0, 1, 2, \ldots, \qquad (5.4)$$

где в качестве начального приближения x_0 можно принять любое значение аргумента в исходном интервале нахождения корня. Метод простых итераций сходится, если в этом интервале $|\varphi(x)| < 1$.

Программа 38/34. Вычисление корня нелинейного уравнения методом простых итераций с точностью до 7 верных цифр

П7	ИП7	...	1	+	1	—	ИП7	XY	П7
—	$x=0$	01	ИП7	С/П					

Инструкция: заменить многоточие операторами вычисления $\varphi(x)$ при $x = P7$, ввести $x_0 = PX$ В/О С/П $PX = P7 = x_{i+1}$.

Для сравнения уточним методом простых итераций корень уравнения $2^x - 5x - 3 = 0$ в интервале $-5 < x < 0$. Представив уравнение в форме $x = (2^x - 3)/5$ и записав в программу 38/34 вместо мно-

готочия операторы 2 X^y 3 — 5 ÷, при x_0= —5 получим (время счета около 1 мин) x_{i+1}= —0,4539964.

Аналогичные рассмотренным программы на входном языке ЯМК21 приведены в приложении. Все эти программы можно модифицировать (если после ввода операторов вычисления функции остаются свободные ячейки программной памяти) описанными ранее способами для повышения уровня автоматизации ввода и вывода данных, но в этом нет особой необходимости, так как время выполнения программ вычисления корня обычно значительно превышает затраты времени на ввод и вывод информации.

2. РЕШЕНИЕ НЕЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Алгебраические функции можно представить степенными многочленами. Поэтому алгебраические уравнения в общем случае описывают формулами

$$A(p) \equiv a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_0 = 0, \qquad (5.5)$$

где часть коэффициентов (кроме a_n) может быть равными нулю.

Алгебраические уравнения с вещественными коэффициентами и степенью n многочлена $A(p)$ имеют n вещественных или комплексно-сопряженных корней (корень k-й кратности рассматривается как k корней). На плоскости комплексного аргумента $p = x + jy$ комплексно-сопряженные корни, а также линии равных значений модуля и фазового угла комплексных значений многочлена $A(p)$ расположены симметрично относительно вещественной оси. На линиях равных фазовых углов $\varphi_A = \pi/2 + q\pi$ ($q = 0, 1, 2, \ldots$) значение $\mathrm{Re}A(p) = 0$, а на линиях равных фазовых углов $\varphi_A = \pi + q\pi$ значение $\mathrm{Im}A(p) = 0$. В корнях сходятся линии фазовых углов в интервале $2\pi k$, где k — кратность корня, а значения модуля равны нулю. Для иллюстрации на рис. 32 показаны некоторые линии равных значений модуля и фазового угла многочлена $A(p) = p^4 + 6p^3 + 17p^2 + 20p + 8$ с вещественным корнем двойной кратности $p_{1,2} = -1$ и парой комплексно-сопряженных корней $p_{3,4} = -2 \pm j2$.

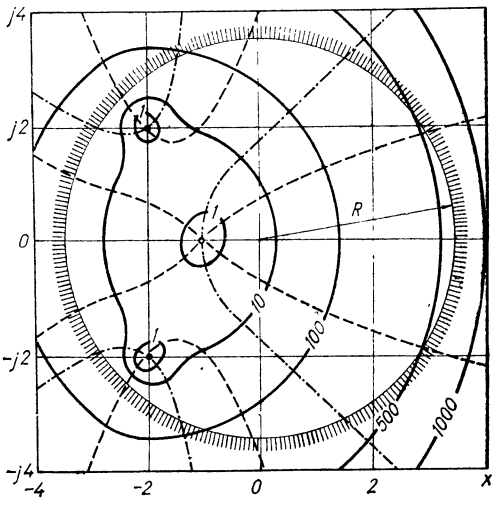


Рис. 32

Все корни алгебраического уравнения лежат внутри кругового кольца с радиусами

$$r = 1/(1 + |a^0_{\max}|/a_0), \quad R = 1 + |a^n_{\max}|/a_n, \qquad (5.6)$$

где $r < R$, $|a^0_{\max}|$ и $|a^n_{\max}|$ — наибольшие модули соответственно из множеств коэффициентов a_n, a_{n-1}, ..., a_2, a_1 и a_{n-1}, a_{n-2}, ..., a_1, a_0.

Расположение корней алгебраического многочлена уточняют, в частности, согласно следующему правилу: внутри любого замкнутого контура на плоскости p число корней многочлена

$$s = \varphi_\Sigma / 2\pi, \tag{5.7}$$

где φ_Σ — суммарное изменение (набег) фазового угла при обходе по контуру, причем φ_Σ изменяется скачком на πk, если контур пересекает корень k-й кратности.

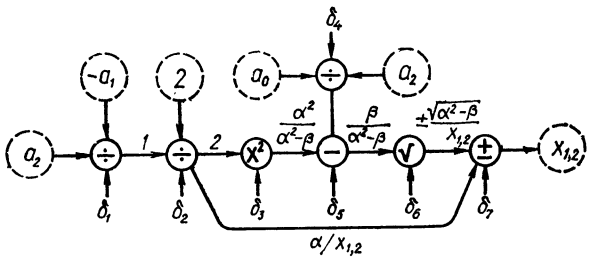


Рис. 33

В общем случае прямыми методами (по формулам с заранее известным числом операций) можно найти решение алгебраических уравнений при степени многочлена $n < 4$. При этом особое внимание следует уделять точности вычисления корней, которая может оказаться весьма низкой. Так, при вычислении корней даже квадратного уравнения $a_2p^2 + a_1p + a_0 = 0$ по формуле

$$p_{1,2} = -a_1/2a_2 \pm \sqrt{(a_1/2a_2) - a_0/a_2} = \alpha \pm \sqrt{\alpha^2 - \beta} \tag{5.8}$$

при $\alpha^2 \gg \beta$ операционная погрешность меньшего по модулю корня может оказаться весьма большой.

Воспользовавшись методикой оценки операционной погрешности, рассмотренной в гл. 2, по графу накопления относительной погрешности (рис. 33) при вычислениях по формуле (5.8) находим относительную погрешность вычисления вещественных корней

$$\delta p_{1,2} \leqslant ((\delta_1 + \delta_2)\,2 + \delta_3)\,\alpha^2/(\alpha^2 - \beta) + (\beta\delta_4/(\alpha^2 - \beta) + \delta_5)/2 + \\ + \delta_6)\,(\pm\sqrt{\alpha^2 - \beta}/p_{1,2}) + (\delta_1 + \delta_2)\,\alpha/p_{1,2} + \delta_7.$$

Приняв $\delta_1 = \delta_2 = \ldots = \delta_7 = 1 \cdot 10^{-7}$, при $\alpha^2 \gg \beta$ получим $\delta p_{1,2} \leqslant$ $\leqslant 10^{-7}(3 + 2|\alpha/p_{1,2}|)$ или, учитывая, что $\alpha = -(p_1 + p_2)/2$, получим $\delta p_{1,2} \leqslant 10^{-7}(4 + |p_{2,1}/p_{1,2}|)$. Следовательно, если корни значительно отличаются по величине, то погрешность вычисления меньшего из них может оказаться значительной. Например, при $p_1/p_2 =$ $= 10^7$ результат вычисления корня p_2 не будет содержать верных цифр. Можно избежать этой погрешности, воспользовавшись для вы-

числения меньшего по модулю корня соотношением $p_1 p_2 = \beta$, откуда $p_2 = \beta/p_1$.

Например, при решении уравнения $p^2 + 7000p + 1 = 0$ по формуле (5.8) получим $p_1 = -3499{,}9999$ и $p_2 = -2 \cdot 10^{-4}$ (для микрокалькулятора с разрядностью мантиссы $r = 8$). Вычисление меньшего корня по значению большего дает $p_2 = \beta/p_1 = -1{,}428571 \cdot 10^{-4}$ и, следовательно, его значение, вычисленное по формуле (5.8), имеет относительную погрешность около 40 %.

При составлении программ решения квадратного уравнения для индикации вида вычисленных корней можно использовать разные способы — индикацию кода вида корня или номеров регистров, в которых хранятся корни различного вида [11], или символ переполнения [19] в случае комплексно-сопряженных корней.

Программа 39/34. Вычисление корней алгебраического уравнения $a_2 p^2 + a_1 p + a_0$

П2	С/П	ИП2	/—/	2	×	÷	П1	С/П	ИП2
÷	↑	ИП1	x^2	—	$x<0$	30	/—/	$\sqrt{}$	ИП1
$x<0$	25	XY	—	0	+	÷	Bx	БП	36
$\sqrt{}$	0	÷	Cx	XY	ИП1	С/П	БП	00	

Инструкция: $a_2 = PX$ В/О С/П $a_1 = PX$ С/П $a_0 = PX$ С/П, если корни вещественные, то $PX = p_1$, $PY = p_2$, если корни комплексно-сопряженные, то $PX =$ ЕГГОГ С/П $PX = \mathrm{Re}p_{1,2}$, $PY = \mathrm{Im}p_{1,2}$.

Контрольный пример: для уравнения $2p^2 + 8p + 10 = 0$ получим комплексно-сопряженные корни $p_{1,2} = -2 \pm j1$, для уравнения $3p^2 + 6p - 45 = 0$ получим вещественные корни $p_1 = -5$, $p_2 = 3$.

Эту программу и ее аналог 160/21 в Приложении можно несколько упростить, если оба корня вычислять по формуле (5.8), но в этом случае при $\alpha^2 \gg \beta$ точность меньшего по модулю корня будет низкой.

В связи с симметрией комплексно-сопряженных корней многочлена относительно вещественной оси из корней многочленов нечетных степеней, по крайней мере, один является вещественным. Для его поиска можно воспользоваться методами, рассмотренными в предыдущем разделе и применимыми для простых корней. Остановимся на методе половинного деления, гарантирующего сходимость вычислений даже в случае кратного корня нечетной кратности.

При использовании этого метода исходный интервал на первой же итерации сокращается вдвое, в связи с чем отпадает необходимость в методах точного определения исходного интервала [11], применение которых связано со значительными затратами времени, и достаточно ограничиться исходными интервалами нахождения вещественных корней $[-R, 0]$ и $[0, R]$, где R определено, как и для формулы (5. 6). При этом полезно учесть, что в случае $A(0) > 0$ в интервале $[-R, 0]$ обязательно находится отрицательный вещественный корень нечетного многочлена, так как при $a_n > 0$ значения $A(-\infty) > 0$ и $A(-R) > 0$. Кроме того, если все коэффициенты многочлена положительны, то не существует положительных вещественных корней.

Так как корни алгебраического уравнения не изменяются при делении его правой и левой частей на отличающееся от нуля число, то удобнее рассматривать нормированные ($a_n = 1$) многочлены нечетной степени, для которых $R = 1 + a^n_{\max}$. Организовав занесение аргумента во все регистры операционного стека с помощью операторов ↑ ↑, при $R = PA$ сравнительно просто реализуем вычисление вещественного корня многочлена нечетной степени методом половинного деления с максимальной точностью по программе

```
Сх ХУ Вх ↑   ↑ ... x<0 12 → ИПА
+ ↑  → ИПА 2 ÷ ПА  — — x=0
02 С/П
```

при замене многоточия операторами вычисления многочлена.

Если решение задачи не ограничивается вычислением одного вещественного корня, то целесообразно совместить подобную программу с фрагментом выделения множителя вычисленного корня p_1 и вычисления по схеме Горнера (4.5) коэффициентов остаточного многочлена четной степени $B(p) = A(p)/(p - p_1)$.

Вычисление текущей невязки уравнения при поиске корня методом половинного деления и схему Горнера при достаточном ресурсе памяти целесообразно совместить. Например, при хранении аргумента в операционном стеке вычисление невязки кубического уравнения и коэффициентов остаточного уравнения второй степени $P3 = b_0$ и $P4 = b_1$ реализуется фрагментом

```
ИП2 + П4 × ИП1 + П3 × ИП0 + ·
```

Дополнив эти фрагменты блоком вычисления корней остаточного квадратного уравнения второй степени, получим программу, одновременно вычисляющую все три корня нормированного кубического уравнения.

Программа 40/34. Вычисление корней нормированного алгебраического уравнения третьей степени $p^3 + a_2p^2 + a_1p + a_0 = 0$

```
Сх  ХУ  Вх  ↑   ↑   ИП2 +  П4  ×  ИП1
+   П3  ×   ИП0 +   x<0 21 →  ИПА +
↑   →   ИПА 2   ÷   ПА   —  ПД  —  x=0
02  ИП3 ИП4 /—/ 2   ÷   ↑   x²  ИП3 —
x<0 51  /—/ √   0   ÷   Сх  →   ХУ  БП
56  √   ХУ  —   С/П ÷   С/П ИПД С/П
```

Инструкция: ($a_0 = P0$, $a_1 = P1$, $a_2 = P2$) $1 + |a^n_{\max}| = PA$ В/О С/П $PX = p_1$ (если все корни вещественны) С/П $PX = p_2$ С/П $PX = p_3$ или $PX =$ ЕГГОГ (если имеется пара комплексно-сопряженных корней) С/П $PX = \mathrm{Re}\, p_{1,2}$, $PY = \mathrm{Im}\, p_{1,2}$ С/П $PX = p_3$.

Контрольный пример: для уравнения $p^3 + p^2 + 4p - 170 = 0$ при $PA = 171$ получим (время счета около 2 мин) $p_{1,2} = -2{,}999999 \pm$ $\pm$ j4,999999, $p_3 = 4{,}999998$ (или $p_{1,2} = -3 \pm$ j5, $p_3 = 5$).

Аналогичная программа 161/21 приведена в приложении. Дополнив ее нормирующим фрагментом, несложно составить и программную реализацию вычисления корней ненормированного уравнения третьей степени.

Реализация метода последовательного деления на входном языке ЯМК34, в котором предусмотрено использование 14 регистров числовой памяти, обеспечивает выделение вещественного корня алгебраических уравнений с нечетным многочленом степени $n < 11$, но при $n > 7$ коэффициенты остаточного многочлена степени $n - 1$ приходится вычислять после вычисления вещественного корня и размещать в регистрах, ранее занятых коэффициентами исходного многочлена.

Программа 41/34. Вычисление вещественного корня алгебраического уравнения с многочленом нечетной степени $n \leqslant 11$

ИП0	ПВ	Сх	ХУ	Вх	↑	↑	ИПС	П0	→
КИП0	+	×	ИП0	$x = 0$	09	Вх	+	ХУ	→
ИПВ	+	$x < 0$	28	→	ИПД	+	↑	→	ИПД
2	÷	ПД	—	—	$x = 0$	04	+	ИП1	ПД
→	ИПС	П0	П1	→	КИП0	+	КП1	×	ИП0
2	—	$x = 0$	44	→	ИПД	+	ПВ	→	ПД
3	П0	КИП0	КП0	ИПС	ИП0	3	+	П0	—
$x < 0$	62	ИПВ	П0	ИПД	С/П				

Инструкция: $a_0 = P0$, $a_1 = P1$, ..., $a_9 = P9$, $a_{10} = PA$, $n = PC$, $1 + |a^n_{\max}| = PД$ В/О С/П $PX = PД = p_1$, $P0 = b_0$, $P1 = b_1$, ..., $P9 = b_9$.

Контрольный пример: для многочлена $p^5 + 9p^4 + 8p^3 + 5p^2 + 4p + 10 = 0$ при $5 = PC$, $11 = PД$ получим (время счета около 8 мин) $p_1 = -8{,}081397$; $b_3 = 0{,}918603$; $b_2 = 0{,}5764045$; $b_1 = 0{,}3418464$; $b_0 = 1{,}2374035$.

Эта универсальная программа имеет низкое быстродействие и ее целесообразно использовать только при необходимости решать уравнения различной степени или близкой к предельной. При более низких степенях целесообразно составлять отдельные программы, заменив блок вычисления многочлена в цикле блоком с более быстрым вычислением многочлена непосредственно по формуле (4.6).

При выборе алгоритма решения алгебраических уравнений четной степени приходится учитывать возможное отсутствие вещественных корней. В этом случае можно использовать либо методы двумерного поиска на комплексной плоскости, рассмотренные в гл. 8, либо одну из возможных процедур выделения из исходного уравнения квадратичного множителя по разложению

$$p^{2m} + a_{2m-1}p^{2m-1} + \cdots + a_1 p + a_0 = (p^{2(m-1)} + b_{2m-3}p^{2m-3} + \\ + \cdots + b_1 p + b_0)(p^2 + sp + q). \qquad (5.9)$$

Большинство известных методов такого разложения (например, метод Берстоу [17], использованный в работе [13] для решения уравнения шестой степени) не гарантирует сходимости вычислений или

неприемлемо для реализации на входных языках микрокалькуляторов с ограниченной емкостью памяти.

Предлагаемый ниже метод, относящийся к той же группе, гарантирует сходимость вычислений и реализуется относительно просто. Алгоритм выделения квадратичного множителя этим методом основан на решении системы нелинейных уравнений, связывающих коэффициенты исходного многочлена a_i с неизвестными коэффициентами b_i и коэффициентами s и q квадратичного множителя:

$$\left.\begin{aligned}
a_{2m-1} &= b_{2m-3} + s;\\
a_{2m-2} &= b_{2m-4} + b_{2m-3}s + q;\\
a_{2m-3} &= b_{2m-5} + b_{2m-4}s + b_{2m-3}q;\\
a_{2m-4} &= b_{2m-6} + b_{2m-5}s + b_{2m-4}q;\\
&\dots\\
a_i &= b_{i-2} + b_{i-1}s + b_iq;\\
&\dots\\
a_2 &= b_0 + b_1s + b_2q;\\
a_1 &= b_0s + b_1q;\\
a_0 &= b_0q.
\end{aligned}\right\} \qquad (5.10)$$

Если задаться значением s, то эту систему можно привести к виду

$$\left.\begin{aligned}
b_{2m-3} &= a^{(1)}_{2m-1};\\
b_{2m-4} &= a^{(2)}_{2m-2} - q;\\
b_{2m-5} &= a^{(1)}_{2m-3} - a^{(2)}_{2m-2}q;\\
b_{2m-6} &= a^{(1)}_{2m-4} - a^{(2)}_{2m-3}q + q^2;\\
&\dots\\
b_{2i} &= a^{(1)}_{2i+2} - a^{(2)}_{2i+4} + \dots + (-q)^{m-(i+1)};\\
b_{2i-1} &= a^{(1)}_{2i+1} - a^{(2)}_{2i+3} + \dots + a^{(m-i)}_{2m-1}(-q)^{m-(i+1)};\\
&\dots\\
b_0 &= a^{(1)}_2 - a^{(2)}_4 + \dots + q^{m-1};\\
0 &= a^{(1)}_1 - a^{(2)}_3q + \dots + (-q)^{m-1}a^{(m)}_{2m-1};\\
0 &= a_0 - a^{(1)}_2q + \dots + (-q)^m,
\end{aligned}\right\} \qquad (5.11)$$

где коэффициенты $a^{(k)}_j$ вычисляются по рекуррентным формулам

$$a^{(k)}_{2m-1} = a^{(k-1)}_{2m-1} - s, \quad a^{(k)}_{2m-l} = a^{(k-1)}_{2m-l} - a^{(k)}_{2m-l+1}, \quad l = 2, 4, \dots, 2m$$

при начальных значениях $a^{(0)}_{2m-l} = a_{2m-l}$, $l = 1, 2, \dots, 2m$.

Если выбранное значение s является корнем системы (5.10), то значение q, полученное по одному из двух последних уравнений системы (5.11), будет удовлетворять систему уравнений (5.10). Если же выбранное значение s не является корнем системы (5.10), то по двум последним уравнениям системы (5.11) можно вычислить коэффициенты

c_1 и c_2 остатка $(c_1+c_2p)/(p^2+sp+q)$, определяющие невязку системы уравнений (5.10) при выбранных значениях s и q.

С учетом коэффициентов остатка представим два последних уравнения системы (5.11) их невязками

$$F_1 = (-1)^{m-1}a_{2m-1}^{(m)}q^{m-1} + \ldots + a_3^{(2)}q + a_1^{(1)} - c_2;$$
$$F_2 = (-q)^m + \ldots + a_2^{(1)}q + a_0 - c_1 \qquad (5.12)$$

и будем вычислять q по следующей итерационной схеме:

$$\Phi_{j-1} = \beta_{j-1}\Phi_j - q\Phi_{j-1}, \quad \Phi_{j-2} = \beta'_{j-1}\Phi_{j-1} - \beta_{j-1}\Phi'_{j-1}, \quad j = j, \; m-1, \ldots, 3, \qquad (5.13)$$

где $\Phi_m = F_2$; $\Phi_{m-1} = F_1$; β_{j-1} — коэффициент при старшей степени функции Φ_{j-1}; β'_{j-1} — коэффициент при старшей степени Φ'_{j-1}.

По схеме (5.13) получим невязку

$$\Phi_1 = \beta_1 q + \beta_0 - f(c_1, c_2),$$

где линейная функция $f(c_1, c_2) = c_1P_1(q) + c_2P_2(q)$, $P_1(q)$ и $P_2(q)$ — многочлены аргумента q.

При $f(c_1, c_2) = 0$ значение $q = -\beta_0/\beta_1$ будет удовлетворять систему уравнений (5.10), если корень $P_1(q)$ или $P_2(q)$ случайно не совпадет с q. Когда есть основания опасаться подобной ситуации, в качестве функции невязки можно использовать обе функции F_1 и F_2.

Рассмотренная схема обеспечивает сравнительно простое вычисление корня q системы уравнений (5.10). Существование таких вещественных корней следует из анализа различных вариантов объединения корней в пары — если даже все $n = 2m$ корней решаемого уравнения комплексно-сопряженные, то число вещественных корней системы (5.10) будет равно m, что соответствует числу квадратичных множителей. Если же часть k корней решаемого уравнения вещественна, то число вещественных корней системы (5.10) возрастет до $m-k+k^2/2$. Например, система (5.10) для решения алгебраического уравнения степени $n = 6$ может содержать 3,7 или 15 вещественных корней.

Так как коэффициент s связан с корнями решаемого уравнения соотношением $s = -(p_i + p_j)$, то $|s| \leqslant |p_i| + |p_j| \leqslant 2\,|p_{\max}| < 2(1+|a^n_{\max}|)$, что позволяет определить интервал нахождения корней системы (5.10). Задача отделения корней упрощается, если в этом интервале находится только нечетное число корней (например, для решения уравнения шестой степени). При четном числе корней системы (например, при решении уравнения восьмой степени функция невязки может иметь 4, 8, 16 или 28 корней) возникающие затруднения можно разрешить численным поиском интервала с противоположными знаками функции невязки $F(s)$ на его границах.

Предложенный алгоритм при использовании входного языка с операторами косвенного обращения к памяти обеспечивает составление универсальной программы решения алгебраических уравнений произвольной степени, содержащую 150 ... 200 шагов. При этом для хра-

нения текущей информации потребуется приблизительно $2m + 5$ регистров памяти.

Следует учитывать, что при последовательном вычислении корней уравнений высокой степени их точность уменьшается. По грубой оценке число верных цифр уменьшается при вычислении следующего корня, и при решении последовательным методом уравнения уже восьмой степени последний из вычисленных корней может не содержать ни одной верной цифры. Для повышения точности можно повторить вычисления несколько раз, изменяя порядок вычисления корней (изменяя выбор начальных интервалов) и используя при повторных расчетах относительно точные значения корней, вычисленные ранее.

Возможности используемых входных языков ЯМК21 и ЯМК34 не обеспечивают программную реализацию универсального алгоритма решения алгебраических уравнений и поэтому рассмотрим частные случаи.

Для составления программы решения уравнения степени $n = 4$ запишем два последних уравнения системы (5.11)

$$a_1^{(1)} - a_3^{(2)} q = c_1, \quad a_0 - a_2^{(1)} q + q^2 = c_2$$

и вычислим q и невязку $F(s)$ по формулам

$$a_3^{(2)} = a_3^{(1)} - s = (a_3 - s) - s = a_3 - 2s;$$
$$a_2^{(1)} = a_2 - s a_3^{(1)} = a_2 - s a_3 + s^2;$$
$$a_1^{(1)} = a_1 - s a_2^{(1)} = a_1 - s a_2 + s^2 a_3 - s^3;$$
$$q = a_1^{(1)} / a_3^{(2)}, \quad F(s) = s^2 - s a_2^{(1)} + a_0.$$

В этом случае $f(c_1, c_2) = q c_1 + a_3^{(2)} c_2$ и ложное решение возможно только при $q = 0$, что соответствует вырожденному уравнению с $a_0 = 0$ и, следовательно, такого решения можно не опасаться.

Вычисления по приведенным формулам реализуются фрагментом программы ($a_i = Pi$, $s = РД$, $F5 = a_2^{(1)}$, $F6 = a_3 - s$, $F7 = q$, $PX = F(s)$)

```
ИПД /—/ ↑   ↑   ИПЗ +   П6 × ИП2 +
П5   ×  ИП1 +   XY  ИП6 + ÷ П7   ↑
ИП5  —  ×  ИП0  +  ...
```

Для уточнения интервала, в котором следует искать корень уравнения, рассмотрим асимптотическое значение функции невязки $F(s)$ при $s \to \infty$. Так как в этом случае $a_3^{(2)} \to -2s$, $a_2^{(1)} \to s^2$, $a_1^{(1)} \to -s^2$ и $q \to s^2/2$, то $F(s) \to (s^4/4 - s^4/2) = -s^4/4 \to -\infty$ на обоих концах интервала $[-R, R]$, где $R = 1 + |a_{\max}^n|$. Если при $2s \to a_3$ значение $b_1^{(1)} \to (a_3(a_3^2 - 4a_2) + 8a_1)/8$ не равно нулю, то $q \to \infty$. Но тогда $F(q) \to q^2 \to \infty$ и на каждом из интервалов $[-R, a_3/2]$ и $[a_3/2, R]$ будет заключено нечетное число корней. Если же при $s = a_3/2$ значение $(a_3(a_3^2 - 4a_2) + 8a_1)/8 = 0$, то возможны два варианта — значение $s = a_3/2$ является корнем системы (5.11) и ему соответствуют значе-

ния $q_{1,2} = a_1/a_3 \pm \sqrt{(a_1/a_3)^2 - a_0}$ при $(a_1/a_3)^2 > a_0$ и (при невыполнении этого неравенства) значение $a_3/2$ не является вещественным корнем системы. В последнем случае из асимптотического приближения функции $F(s)$ при $s \to a_3/2$ следует, что $q \to (4a_2 - a_3^2)/8$, $a_2^{(1)} \to$ $\to (4a_2 - a_3^2)/4$ или, с учетом равенства $a_2 = a_3^2/4 - 2a_1/a_3$, получаем $q \to a_1/a_3$ и $a_2^{(1)} \to 2(a_1/a_3)$, откуда $F(s) \to ((a_1/a_3)^2 - a_0)$ и при $(a_1/a_3)^2 -$ $- a_0 > 0$ корни находятся в указанных интервалах.

Наиболее неприятен первый случай, при котором q определяется с низкой точностью и может возникнуть переполнение при вычислении q для $s = a_3/2$. Однако коэффициент s окажется верным и следует опасаться лишь близких к нулю значений a_3, при которых время счета возрастет.

По найденной паре значений s_1 и q_1 легко определить вторую пару $s_2 = a_3 - s_1$, $q_2 = a_0/q_1$, но целесообразно предварительно найти решение первого квадратного уравнения.

Программа 42/34. Вычисление корней алгебраического уравнения $p^4 + a_3p^3 + a_2p^2 + a_1p + a_0 = 0$

ИП3	/—/	2	÷	ПД	ИПД	↑	↑	ИП3	+
П6	×	ИП2	+	П5	×	ИП1	+	$x = 0$	65
ИП1	ИП3	÷	↑	x^2	ИП0	—	$x \geqslant 0$	65	$\sqrt{}$
+	П7	ИПД	2	÷	↑	x^2	ИП7	—	$x < 0$
50	/—/	$\sqrt{}$	0	÷	Cx	→	XY	БП	54
$\sqrt{}$	+	÷	Bx	С/П	ИП3	/—/	ИПД	—	ПД
ИП0	ИП7	÷	БП	31	XY	ИП6	+	$x \neq 0$	84
÷	П7	↑	ИП5	—	×	ИП0	+	$x < 0$	84
ИПД	ИПА	—	ПД	ИПА	2	÷	ПА	ИПД	+
ПД	Bx	—	$x = 0$	05	ИП7	БП	32		

Инструкция: ($a_0 = P0$, $a_1 = P1$, $a_2 = P2$, $a_3 = P3$), $2(1 + |a^n_{max}|) = PA$ В/О С/П $PX = p_1$, $PY = p_2$ или $PX =$ ЕГГОГ С/П $PX = \mathrm{Re}\, p_{1,2}$, $PY = \mathrm{Im}\, p_{1,2}$; для вычисления второй пары корней нажать клавишу С/П с аналогичным продолжением; для каждого квадратичного множителя $PД = -s$, $P7 = q$.

Контрольный пример: для $p^4 + 9p^3 + 31p^2 + 59p + 60 = 0$ при $2(1 + |a^n_{max}|) = 122 = PA$ получим (время счета около 7 мин) $p_{1,2} =$ $= -1 \pm j1{,}9999999$ и (время счета около 10 с) $p_3 = -3$, $p_4 = -4$.

В этой программе предусмотрена проверка условия $a_3(a_3^2 - 4a_2) +$ $+ 8a_1 = 0$, при выполнении которого $q_1 = a_1/a_3 + \sqrt{(a_1/a_3)^2 - a_0}$ и $s_1 = a_3/2$. Значение $2(1 + |a^n_{max}|)$ можно вводить и с отрицательным знаком, что приведет к изменению порядка вычисления корней и позволит уточнить их значения.

Если исключить из рассмотренной программы блок проверки условия $a_3(a_3^2 - 4a_2) + 8a_1 = 0$, то ее можно дополнить фрагментом вычисле-

ния пятого вещественного корня. В такой программе стандартный блок реализации метода половинного деления целесообразно использовать также для разложения остаточного уравнения четвертой степени на квадратичные множители с помощью программно формируемого оператора проверки флага (рис. 34).

Для решения уравнения шестой степени два последние уравнения системы (5.11) примут вид

$$\left.\begin{aligned} a_1^{(1)} - qa_3^{(2)} + q^2a_5^{(3)} &= c_1; \\ a_0 - qa_2^{(1)} + q^2a_4^{(2)} - q^3 &= c_2, \end{aligned}\right\} \tag{5.14}$$

где

$$a_5^{(3)} = a_5^{(2)} - s = (a_5^{(1)} - s) - s = ((a_5 - s) - s) - s = a_5 - 3s;$$

$$a_4^{(2)} = a_4^{(1)} - sa_5^{(2)} = a_4 - 2sa_5 + 3s^2;$$

$$a_3^{(2)} = a_3^{(1)} - sa_4^{(2)} = a_3 - 2sa_4 + 3s^2a_5 - 4s^3;$$

$$a_2^{(1)} = a_2 - sa_3^{(1)} = a_2 - sa_3 + s^2a_4 - s^3a_5 + s^4;$$

$$a_1^{(1)} = a_1 - sa_2^{(1)} = a_1 - sa_2 + s^2a_3 - s^3a_4 + s^4a_5 - s^5.$$

Программная реализация этих формул несколько короче реализации рекурсивной процедуры вычисления $a_i^{(j)}$ и не требует дополнительных регистров-счетчиков. Значение q по составленным уравнениям (5.14) целесообразно вычислять по аналитическому выражению

$$q = \frac{a_1^{(1)}(a_4^{(2)}a_5^{(3)} - a_3^{(2)}) - a_0(a_5^{(3)})^2}{a_5^{(3)}(a_1^{(1)} - a_2^{(1)}a_5^{(3)}) + a_3^{(2)}(a_4^{(2)}a_5^{(3)} - a_3^{(2)})}. \tag{5.15}$$

В этом случае $f(c_1, c_2) = (\alpha + \beta q)c_1 + \gamma c_2$, где α, β и γ — постоянные коэффициенты. Поэтому при оценке функции невязки $F(s)$ во избежание ложного решения целесообразно воспользоваться первым из уравнений (5.14). Все необходимые вычисления, включая оценку $F(s)$, выполнимы одним фрагментом программы длиной менее 70 шагов.

Рассмотрим асимптотическое приближение функции $F(s)$ для уравнения шестой степени. При $s \to \infty$ получим $a_1^{(1)} \to -s^5$, $a_3^{(1)} \to s^4$, $a_3^{(2)} \to -4s^3$, $a_4^{(2)} \to 3s^2$, $a_5^{(2)} \to -3s$, а $q \to (-s^5(-9s^3 + 4s^3))/(-3s \times (-s^5 + 3s^5) - 4s^3(-9s^3 + 4s^3)) = 5s^2/14$. С учетом этих приближе-

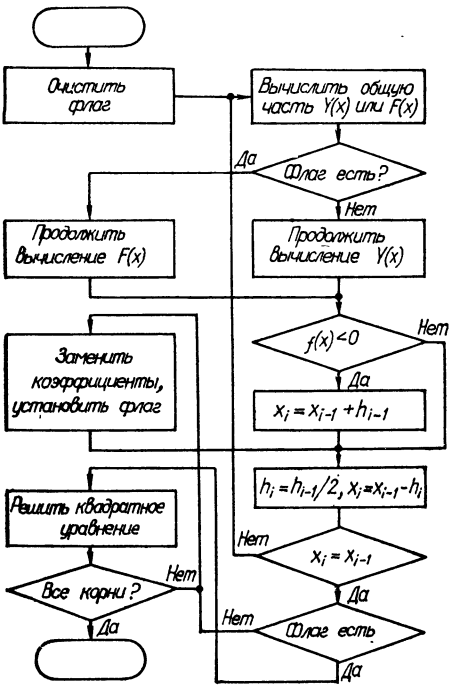


Рис. 34

ний при определении невязки по первому из уравнений (5.14) получим $F(s) \to -s^5 - (-4s^3)\,5s^2/14 + (-3s)\,25s^4/14^2 \to 9s^5/196$.

Следовательно, при $s \to \pm\infty$ значение $F(s) \to \pm\infty$ и, если начинать поиск одного из трех (или большего нечетного числа) корней в интервале $[-R, R]$ с нижней границы $-R$, выполнение неравенства $f(x_i) > 0$ означает, что пройден один (или нечетное число) корень и необходимо вернуться на шаг назад. При пересечении такой особой точки знак коэффициента q будет изменяться, но невязка $F(s)$ первого из уравнений (5.14) независимо от значения q будет стремиться к $\pm\infty$ в зависимости от знака коэффициента $a_5^{(3)}$. Если в этой особой точке $a_5^{(3)} > 0$, то при поиске слева направо следует сохранить направление следующего шага, а при поиске справа налево — вернуться на шаг назад. Одновременное обращение в нуль числителя и знаменателя формулы (5.15) не изменяет стратегии поиска, а неприятности возникают, когда в особой точке $q = 0$ и, следовательно, эта точка является корнем системы уравнений (5.14). Если по программе вычисляются именно эти значения s и q, то погрешность вычисления q может оказаться большой.

Дополнив основной блок фрагментом проверки неравенства нулю правой части выражения (5.15) и стандартным фрагментом половинного деления, получим следующую программу выделения квадратичного множителя левой части уравнения шестой степени.

Программа 43/34. Выделение квадратичного множителя $p^2 + sp + q$ левой части алгебраического уравнения $p^6 + a_5p^5 + a_4p^4 + a_3p^3 + a_2p^2 + a_1p + a_0 = 0$

Сх	ПД	ИПД	↑	↑	ИП5	—	×	ИП4	+
×	ИП3	—	П8	×	ИП2	+	П7	×	ИП1
—	П6	→	3	×	ИП5	—	ПВ	ИП5	—
×	ИП4	+	П9	×	ИП8	+	П8	ИП9	ИПВ
×	—	ПС	ИП6	×	ИПВ	x^2	ИП0	×	+
ИПВ	ИП7	×	ИП6	—	ИПВ	×	ИПС	ИП8	×
+	$x \neq 0$	93	÷	ПС	↑	ИПВ	×	ИП8	—
×	ИП6	+	$x < 0$	79	ИПД	ИПА	—	ПД	ИПА
2	÷	ПА	ИПД	+	ПД	Вх	—	$x = 0$	02
ИПС	ИПД	С/П	ИПВ	БП	73				

Инструкция: ($a_0 = P0$, $a_1 = P1$, ..., $a_5 = P5$) 3 $(1 + |a^n_{\max}|) =$ $= PA$ В/О С/П $PX = s$, $PY = q$.

Контрольный пример: для уравнения $p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 0{,}5 = 0$ при $18 = PA$ получим (время счета около 14 мин) коэффициенты квадратичного множителя $p^2 - 0{,}48834684\ p + 0{,}42211151$ (остаточный многочлен четвертой степени $p^4 + 5{,}4883468p^3 + 6{,}2581053p^2 + 3{,}7394315p + 1{,}1845213$).

Корни полученных по этой программе многочленов вычисляются с помощью ранее приведенных программ. Для решения уравнений с многочленами степени $n > 7$ следует воспользоваться методами нелинейного программирования, рассмотренными в гл. 8.

3. РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ

В инженерных задачах часто возникает необходимость поиска корней (решения) x_i системы линейных уравнений (2.6), обычно записываемой в матричной форме

$$\begin{bmatrix} a_{11}a_{12} & \dots & a_{1n} \\ a_{21}a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots \\ a_{n1}a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_n \end{bmatrix}. \qquad (5.16)$$

Автоматический поиск решения этой системы с $n \leqslant 3$ несложно программировать (см. программы 130/34 и 167/21 в Приложении) с помощью формулы

$$x_i = \sum_{j=1}^{n} \Delta_{ji} q_j / \Delta, \quad i = 1, 2, \dots, n, \qquad (5.17)$$

где Δ и Δ_{ji} — определитель и алгебраические дополнения квадратной матрицы коэффициентов системы уравнений (5.16).

При большем числе n уравнений возникают затруднения с размещением программы и n^2+n исходных данных (коэффициентов a_{ij} и свободных членов q_i) в памяти. В таких случаях приходится прибегать к тщательному выбору численного метода решения системы линейных уравнений. Среди них наиболее известен вариант метода исключений Гаусса, называемый схемой единственного деления [4] и основанный на преобразовании исходной системы к треугольному виду (прямой ход).

По этой схеме на каждом p-м шаге прямого хода коэффициенты a_{ij} и $a_{i,\,n+1} = q_i$ очередного p-го уравнения преобразуют по формуле

$$a_{pj}^{(p)} = a_{pj}^{(p-1)} / a_{pp}^{(p-1)}, \quad j = p,\ p+1,\ p+2,\ \dots,\ n+1, \qquad (5.18)$$

а коэффициенты следующих ($i > p$) уравнений — по формуле

$$a_{ij}^{(p)} = a_{ij}^{(p-1)} - a_{ip}^{(p-1)} a_{pj}^{(p)}, \quad j = p,\ p+1,\ p+2,\ \dots,\ n+1$$
$$i = p+1,\ \dots,\ n. \qquad (5.19)$$

После выполнения n таких шагов получают треугольную систему уравнений

$$x_i + \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j = a_{i,\,n+1}^{(i)}, \quad i = 1,\ 2,\ \dots,\ n$$

с известным корнем $x_n = q_n^{(n)} = a_{n,\,n+1}^{(n)}$, по которой в процессе обратного хода вычисляют остальные корни x_{n-1}, x_{n-2}, ..., x_1.

Непосредственная реализация этих формул при $n > 2$ на входном языке ЯМК21 не обеспечивается емкостью программной памяти, но при предварительном преобразовании первого уравнения, последовательном вводе коэффициентов a_{21} и a_{31}, а также выполнении поворотов

кольцевого стека памяти в обычном режиме удается на этом языке составить программу 168/21 приложения для $n = 3$.

Емкость числовой памяти микрокалькуляторов с входными языками ЯМК21 и ЯМК34 при $n > 3$ недостаточна для размещения всех коэффициентов и свободных членов системы уравнений (5.16) при вычислении корней по схеме единственного деления, и приходится прибегать к методам, допускающим одновременное хранение в памяти лишь части коэффициентов. К таким методам относится вариант метода Гаусса, называемый методом оптимального исключения [2].

При использовании метода оптимального исключения на первом шаге коэффициенты первого уравнения (первой строки) системы (5.16) преобразуют по формулам

$$a_{1j}^{(1)} = a_{1j}/a_{11}, \quad j = 1, 2, \ldots, n+1,$$

а на втором шаге исключают переменную x_1 из второго уравнения, преобразуя его коэффициенты по формуле

$$a_{2j}^{(2)} = (a_{2j} - a_{21}a_{1j}^{(1)})/(a_{22} - a_{21}a_{12}^{(1)}), \quad j = 2, 3, \ldots, n+1$$

и исключают переменную x_2 из первого уравнения, преобразуя его коэффициенты по формуле

$$a_{1j}^{(2)} = a_{1j}^{(1)} - a_{2j}^{(2)}a_{12}^{(1)}, \quad j = 2, 3, \ldots, n+1.$$

На третьем шаге исключают переменные x_1 и x_2 из третьего уравнения, выполнив преобразование

$$a_{3j}^{(3)} = (a_{3j} - (a_{31}a_{1j}^{(2)} + a_{32}a_{2j}^{(1)}))/(a_{33} - (a_{31}a_{13}^{(2)} + a_{32}a_{23}^{(2)}),$$
$$j = 4, \ldots, n+1,$$

и исключают переменную x_3 из первого и второго уравнений, преобразуя их коэффициенты по формулам

$$a_{ij}^{(3)} = a_{ij}^{(2)} - a_{3j}^{(3)}a_{i3}^{(2)}, \quad i = 1, 2; \; j = 4, 5, \ldots, n+1.$$

В результате после третьего шага три первых уравнения преобразуются к виду

$$x_1 + a_{14}^{(3)}x_4 + \cdots + a_{1n}^{(3)}x_n = a_{1,n+1}^{(3)};$$
$$x_2 + a_{24}^{(3)}x_4 + \cdots + a_{2n}^{(3)}x_n = a_{2,n+1}^{(3)};$$
$$x_3 + a_{34}^{(3)}x_4 + \cdots + a_{3n}^{(3)}x_n = a_{3,n+1}^{(3)}$$

и в дальнейшем требуется хранить только $3n - 6$ коэффициентов трех первых уравнений вместо $3(n+1)$ в схеме единственного деления.

Выполняя на каждом следующем p-м шаге преобразования коэффициентов по формулам

$$a_{pj}^{(p)} = (a_{pj} - \sum_{r=1}^{p-1} a_{rj}a_{pr})/(a_{pp} - \sum_{r=1}^{p-1} a_{rp}^{(p-1)}a_{pr}), \quad j = p+1, \ldots, n+1$$

и

$$a_{ij}^{(p)} = a_{ij}^{(p-1)} - a_{pj}^{(p)}a_{ip}^{(p-1)}, \; i = 1, 2, \ldots, p-1; \; j = p+1, \ldots, n+1,$$

после n шагов только прямого хода получают искомое решение:

$$x_i = q_i^{(n)} = a_{i,\,n+1}^{(n)}.$$

Так как на каждом шаге в вычислениях используется $n + 1$ коэффициентов, а после выполнения p-го шага исключается $2p - 1$ коэффициентов, то в памяти необходимо хранить число коэффициентов

$$m_p = p(n + 1) - \sum_{i=1}^{p} (2i - 1) = p(n + 1 - p).$$

Максимальным будет число коэффициентов, которые должны храниться в памяти перед следующим шагом

$$m'_{p+1} = m_p + n + 1 = (p + 1)(p - 1) - p^2$$

в случае $p_m = (n + 1)/2$ и, например при $n = 4$, получим $p_m = 2{,}5$; $m_2 = 11$, $m_3 = 11$, а при $n = 5$ получим $p_m = 3$, $m_2 = 12$, $m_3 = 13$, $m_4 = 14$. Следовательно, емкость числовой памяти микрокалькуляторов с входным языком ЯМК34 (с учетом операционных регистров) достаточна для решения методом оптимального исключения систем из $n \leqslant 4$ линейных уравнений в программируемом режиме.

Вычисления по этому и другим методам с многократными повторениями вычислительных процедур целесообразно предварительно реализовать универсальной программой (без учета ее длины), пригодной для систем с числом уравнений, ограниченным лишь емкостью числовой памяти. Для этого следует выбрать рабочий алгоритм с использованием косвенной адресации и определить размещение операндов на каждом этапе вычислений. Предполагая, что не равные тождественно нулю или единице коэффициенты, используемые на следующем шаге, будут перемещаться в регистрах числовой памяти, начиная с регистра с номером *5*, номер k регистра для хранения коэффициента a_{ij}^{p} при $j > p$ можно определить по формуле $k = (i - 1)(n + 1 - p) + (j - p)$, где n — число уравнений. По этому номеру и организуется косвенный вызов очередного операнда.

В качестве адресных во входном языке ЯМК34 целесообразно выбрать регистры с номерами от *0* до *4*, содержимое которых уменьшается на единицу после каждого выполнения оператора косвенного вызова из памяти. В этом случае коэффициенты следует размещать в порядке возрастания индекса j в регистрах с наибольшими номерами (регистру *Д* соответствует номер 13). При решении системы из четырех уравнений участвующие в вычислениях коэффициенты после p-го шага исключения удобно размещать в памяти, как указано в табл. 11.

Если предусмотреть предварительную очистку регистров, заполняемых коэффициентами на следующем шаге (в табл. 11 таким регистрам соответствует их нулевое содержимое), то при последовательном вводе коэффициентов алгоритм вычислений описывается следующим образом:

1. Для каждого вводимого операнда a_{ij} при $i < j$ вычислить

$$s_l^{(j)} = s_l^{(j-1)} + a_{lj} a_{jl}^{(p)}, \quad s_l^0 = 0, \quad l = 1, 2, \ldots, n - p.$$

11. Изменение содержимого регистров памяти после выполнения p-го шага исключения для $n = 4$

p	$P5$	$P6$	$P7$	$P8$	$P9$	PA	PB	PC	$PД$
1	0	0	0	0	0	$q_1^{(1)}$	$a_{14}^{(1)}$	$a_{13}^{(1)}$	$a_{12}^{(1)}$
2	0	0	0	$q_2^{(2)}$	$a_{24}^{(2)}$	$a_{23}^{(2)}$	$q_1^{(2)}$	$a_{14}^{(2)}$	$a_{13}^{(2)}$
3	0	0	0	$q_3^{(3)}$	$a_{34}^{(3)}$	$q_2^{(3)}$	$a_{24}^{(3)}$	$q_1^{(3)}$	$a_{14}^{(3)}$
4	0	0	0	0	0	$q_4^{(4)}$	$q_3^{(4)}$	$q_2^{(4)}$	$q_1^{(4)}$

2. Вычислить $a_{ii}^{(p+1)} = a_{ii}^{(p)} - a_{p+1,j}^{p+1} s_1^{(i)}$.
3. Вычислить $a_{ij}^{(p+1)} = (a_{ij}^{(p)} - s_{i-1})/a_{ii}^{(p+1)}$, $i > j$.
4. Вычислить $a_{ij}^{(p+1)} = a_{ij}^{(p)} - a_{(p+1,j)}^{(p+1)} \; a_{i,p+1}^{(p)} \; j > p + 1$.
5. Переслать вычисленные коэффициенты в регистры памяти с большими номерами, исключить коэффициенты $a_{i,\,p+1}^{(p)}$, $i = 1, 2, \ldots, p + 1$ и очистить $n - p + 1$ регистров (кроме адресных) с меньшими номерами.

Составление программы упрощается, если описание алгоритма представить схемой вычислений (например, для $n = 4$) в виде таблицы с указанием операций, выполняемых после ввода в вычисления очередного коэффициента. Программа составляется из следующих основных блоков.

Блок вычислений при вводе коэффициентов a_{ij}, $i > j$

С/П ↑ КИП3 × КИП1 + КП2 — L0 A_2
→ П0 ИП2 + П1 П2 ИП3 $x = 0$ A_1 ...,

с адресами A_1 и A_2 передачи управления соответственно первому (С/П) и второму (↑) операторам программы.

В этом блоке вычисления в цикле выполняются $n + 2 - i$ раз по счетчику итераций на адресном регистре *0*. Содержимое адресных регистров *1* и *2* изменяется от максимального $(k - (i - 1)(n + 1 - i))$ до минимального $(k - i\,(n + 1 - i))$ значений и восстанавливается при каждом обращении к этому блоку по формуле $P1 = P2 = P1 + n + 1 - i$. Содержимое адресного регистра *3* изменяется от максимального значения k до минимального значения $k - (i - 1)(n + 1 - i)$, равного номеру последнего регистра, заполненного на предыдущем шаге исключения. Значение $n + 1 - i$ хранится в операционном стеке, вычисления в блоке продолжаются до выполнения условия $P2 + n + 1 - i = P3$.

При вводе диагональных элементов матрицы коэффициентов системы уравнений вычисления реализуются фрагментом ... КИП2 — П4 С/П ..., а при вводе коэффициентов a_{ij}, $i < j$ — фрагментом

С/П КИП2 — ИП4 ÷ КП1 L3 A ИП1 ИП0
+ П3 1 4 П1 П2 ...,

где A — адрес первого оператора фрагмента.

Выполнение этого фрагмента при вводе $q_i = a_{i,\,n+1}$ повторяется для всех значений $i < j \leqslant n + 1$ согласно содержимому адресного регистра *3*. После выполнения вычислений в фрагменте $P1 = P2 = k$, $P3 = k - i(n + 1 - i)$ и исключения переменной x_i выполняется преобразование коэффициентов предыдущих уравнений с помощью фрагмента

КИП1	/—/	↑	КИП3	+	КИП1	+	КП2	→	L0
A_3	→	П0	ИП3	+	П3	ИП1	—	$x = 0$	A_1

с адресами A_1 и A_3 переходов соответственно к первому и третьему операторам фрагмента. В этом фрагменте цикл повторяется $n + 1 - i$ раз, а выполнение фрагмента повторяется $i - 1$ раз (двойной цикл). При этом стираются коэффициенты a_{si}, $s = 1, 2, \ldots, i - 1$, а содержимое регистров *1* и *2* изменяется от значения k до $k - (i - 1)(n + 2 - i)$ и $k - (i - 1)(n + 1 - i)$ соответственно. Содержимое регистра *3* изменяется от $k - (i - 1)(n + 2 - i)$ до $k - i(n + 2 - i)$ с восстановлением суммы содержимого регистров *0* и *3* после каждого выполнения внутреннего цикла.

Очередной шаг исключения завершается пересылкой («упаковкой») преобразованных коэффициентов n-го уравнения с помощью фрагмента КИП3 КП2 L1 *A* ..., где *A* — адрес перехода к оператору КИП3, и очисткой регистров, используемых на следующем шаге, с помощью фрагмента Сх КП2 L1 *A*, где *A* — адрес перехода к оператору Сх.

Вывод вычисленных корней системы уравнений целесообразно организовать последовательно с вызовом корней в регистр *X*. Связав рассмотренные фрагменты операторами установки начальных значений содержимого адресных регистров, получим следующую универсальную программу.

Программа 44/34. Вычисление корней системы из $n \leqslant 4$ линейных уравнений методом оптимального исключения

П4	1	4	П2	ИП0	П1	С/П	ИП4	÷	КП2
L1	06	1	4	П3	ИП0	П1	L1	23	КИП3
С/П	БП	19	Сх	КП2	L1	24	КП2	ИП0	ИП2
+	П1	П2	ИП3	—	$x \neq 0$	42	С/П	ПП	84
БП	28	КИП0	ИП0	П3	С/П	КИП2	—	П4	С/П
КИП2	—	ИП4	÷	КП1	L3	49	ИП1	ИП0	+
П3	1	4	П1	П2	КИП1	/—/	ПП	84	ИП3
+	П3	ИП1	—	$x = 0$	65	ИП0	П1	КИП3	КП2
L1	78	БП	12	ИП0	XY	↑	КИП3	×	КИП1
+	КП2	→	L0	86	→	П0	В/О		

Инструкция: $n = P0$, $a_{11} = PX$ В/О С/П $a_{12} = PX$ С/П $a_{13} = PX$ С/П ... $q_1 = PX$ С/П $a_{21} = PX$ С/П $a_{22} = PX$ С/П ... $q_2 = PX$ С/П ... $a_{n1} = PX$ С/П $a_{n2} = PX$ С/П ... $q_n = PX$ С/П $PX = PД = x_1$ С/П $PX = PC = x_2$ С/П $PX = PB = x_3$ С/П $PX = PA = x_4$.

Время выполнения программы зависит от числа n уравнений. Так, корни $x_1 = 3$, $x_2 = 1$ системы из двух уравнений $5x_1 - 10x_2 = 5$, $4x_1 -$

$-2x_2=10$ вычисляются примерно за 50 с, корни $x_1=30$, $x_2=20$, $x_3=10$ (с небольшой операционной погрешностью) системы из трех уравнений $x_1+5x_2+6x_3=190$, $2x_1-x_2+4x_4=80$, $3x_1+3x_2+3x_3=180$ вычисляются примерно за 135 с, а время вычисления корней $x_1=1$, $x_2=3$, $x_3=-2$, $x_4=10$ системы из четырех уравнений $2x_1+12x_2+20x_3+2x_4=18$, $6x_1+2x_2+6x_3+2x_4=20$, $5x_1+5x_2+10x_3=0$, $10x_1+2x_2+4x_3+x_4=18$ составляет около 250 с.

Нулевые значения коэффициентов при вычислениях по этой программе (как и по другим программам, например, следующей, в которых операционный стек используется для хранения данных) необходимо вводить оператором 0 (а не Сх), а для исправления ошибок при вводе — стирать содержимое регистра X только оператором Сх, чтобы не изменить содержимого остальных регистров операционного стека.

При использовании метода оптимального исключения, как и других вариантов метода исключения Гаусса, возможно переполнение даже при существовании решения системы уравнений. В таких случаях, как и для уменьшения операционной погрешности результата, целесообразно так переставить уравнения исходной системы, чтобы вводимые в вычисления диагональные элементы матрицы коэффициентов были возможно большими.

Реализация в программе 44/34 схемы исключений, при которой вычисления выполняются над группами коэффициентов, и организация хранения промежуточных результатов в операционных регистрах позволили сократить необходимое число регистров памяти до $(p+1)(n+1-p)$. Следовательно, для решения системы из пяти уравнений потребуется $(p+1)(6-p)$ или не более 12 регистров памяти для одновременного хранения коэффициентов. Однако в этом случае для косвенной адресации остается только два адресных регистра, а при программной реализации требуемых пересылок емкость программной памяти оказывается недостаточной. По этой причине в следующей программе предусмотрено предварительное нормирование коэффициентов первого уравнения системы.

Программа 45/34. Вычисление корней системы из пяти линейных уравнений методом оптимального исключения

ИПД	XY	↑	→	↑	КИП0	×	КИПД	+	ПП
89	$x<0$	03	→	XY	ПД	С/П	ИПД	ИП0	−
3	+	$x\geqslant 0$	03	→	КИПД	−	ИПД	П0	→
С/П	КИП0	−	XY	÷	Вх	XY	ПП	89	$x=0$
29	→	XY	ПД	1	3	П0	П1	→	КИП1
/−/	↑	→	↑	КИПД	×	КИП1	+	КП0	ПП
90	$x=0$	52	→	XY	ПД	ИП1	ИП0	6	−
−	−	$x=0$	48	П1	→	КИПД	КП0	Сх	ПП
89	$x<0$	75	+	1	3	П0	БП	14	КПД
→	ИПД	1	−	ПД	1	−	В/О		

Инструкция: $(13=P0)\,5=РД$, $a_{12}/a_{11}=PC$, $a_{13}/a_{11}=PB$, $a_{14}/a_{11}=PA$, $a_{15}/a_{11}=P9$, $q_1/a_{11}=P8$, $0=P3=P4=P5=P6=P7$,

$a_{21} = PX$ В/О С/П $a_{22} = PX$ С/П ... $a_{25} = PX$ С/П $q_2 = PX$ С/П ... ... $a_{51} = PX$ С/П ... $q_5 = PX$ С/П $PC = x_1$, $PB = x_2$, $PA = x_3$, $P9 = x_4$, $P8 = x_5$.

Контрольный пример: для системы уравнений

$$\begin{bmatrix} 2 & 10 & 16 & -6 & 8 \\ 3 & 2 & 4 & 0 & 8 \\ 6 & -4 & 1 & 3 & 5 \\ 8 & -2 & 4 & -3 & 1 \\ 3 & 2 & 4 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 20 \\ 20 \\ 10 \\ 0 \\ 30 \end{bmatrix}$$

получим (время счета около 11 мин) решение $x_1 = 6{,}8205123$; $x_2 = 10{,}517949$; $x_3 = -3{,}1282053$; $x_4 = 6{,}6307692$; $x_5 = -1{,}1230768$.

Универсальная программа, подобная программе 44/34, для микрокалькулятора с большей емкостью запоминающих устройств обеспечивает вычисление корней систем из относительно бо́льшего числа уравнений. Например, с помощью подобной программы на микрокалькуляторе типа Programmable-59 (фирмы Техас Инструментс) при выделении 100 регистров памяти для хранения операндов можно решать системы, содержащие до 17 линейных уравнений (вместо 10 по стандартной программе, хранящейся в блоке прикладных программ), но время решения составит несколько часов.

При использовании микрокалькуляторов с входными языками ЯМК21 и ЯМК34 решение систем из $n > 3$ линейных уравнений обеспечивается полуавтоматическим режимом вычислений с записью промежуточных результатов в вычислительный бланк, выполняющий функции внешней памяти. Подобный бланк оказывается достаточно компактным для прямого хода схемы единственного деления [11], но необходимость вычисления корней по формулам обратного хода приводит к дополнительным затратам времени. Поэтому для решения системы линейных уравнений в полуавтоматическом режиме целесообразно использовать метод Жордана [2], обеспечивающий (как и метод оптимального исключения) получение искомого решения после выполнения только прямого хода. Согласно этому методу на каждом p-м шаге коэффициенты p-го уравнения преобразуют по формуле (5.18), а коэффициенты всех остальных (а не только следующих) уравнений — по формуле (5.19). Для удобства заполнения вычислительного бланка программу следует составлять так, чтобы на каждом шаге коэффициенты, преобразуемые по формуле (5.18), вычислялись последними, что приводит к циклической перестановке строк преобразуемых коэффициентов, но на последнем шаге их исходный порядок восстанавливается, а коэффициенты $a^n_{i,n+1} = q^n_i$ равны искомым корням x_i.

Программа 46/34. Вычисление корней системы линейных уравнений по методу Жордана с циклической перестановкой строк в вычислительном бланке

↑	Cx	XY	ИП1	÷	П0	XY	1	ПС	XY
С/П	ИПД	ИПС	1	+	ПС	—	$x \geqslant 0$	26	XY
КИПС	ИП0	×	—	БП	10	XY	ИП0	БП	02

Перед вычислениями по этой программе заготовляют вычислительный бланк с $n+1$ частями, каждая из которых содержит n строк и $n+2$ столбцов (табл. 12). В нулевой части бланка n столбцов заполняют коэффициентами a_{ij} матрицы параметров в уравнении (5.16), $(n+1)$-й столбец — свободными членами $q_i = a_{i,\,n+1}$, а в $(n+2)$-й столбец записывают суммы остальных элементов соответствующих строк для контроля точности вычислений. Число n уравнений вводят в регистр $Д$.

12. Вычислительный бланк для решения системы линейных уравнений методом Жордана—Гаусса с циклической перестановкой строк

p	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	Σ
0	2	12	20	2	18	54
	6	2	6	2	20	36
	5	5	10	0	0	20
	10	2	4	1	18	35
1		—34	—54	—4	—34	—126
		—25	—40	—5	—45	—115
		—58	—96	—9	—72	—235
	1	6	10	1	9	27
2			—0,29412	—2,0588238	—20	—22,35294
			—3,882359	—2,1764711	—14	—20,05883
		1	0,470589	0,2941177	3	4,764706
		1	1,5882352	0,11764705	1	3,7058823
3				24,999828	249,9983	274,99808
			1	—2,9999794	—28,999796	—30,99977
			1	—10,999912	—106,99913	—116,99902
			1	6,9999449	67,999456	75,999388
4				1	0,999998	1,999997
				1	2,99999	3,99999
				1	—1,999993	—0,999991
				1	10	10,999998

При заполнении очередной p-й части бланка в p нижних строк p-го столбца записывают единицы, вводят в память микрокалькулятора элементы p-го столбца $(p-1)$-й части бланка $a_{1p}^{(p-1)} = P1$, $a_{2p}^{(p-1)} = P2$, ..., $a_{np}^{(n-1)} = Pn$ (при $n \leqslant 11$) и элемент первой строки $(p+1)$-го столбца $(p-1)$-й части бланка $a_{1,\,p+1}^{(p-1)} = PX$, после чего нажимают клавиши В/О и С/П, что приводит к высвечиванию нуля. Затем в регистр X поочередно вводят остальные элементы $(p+1)$-го и следующих столбцов $(p-1)$-й части, нажимая после каждого ввода

только клавишу С/П и записывая высвечиваемые значения $a_{ij}^{(p)}$ на места, соответствующие предыдущим элементам $(p-1)$-й части бланка (например, после ввода $a_{2,\,p+1}^{(p-1)}$ записывают индицируемое значение $a_{1,p+1}^{(p)}$, после ввода $a_{1,\,p+2}^{(p-1)}$ записывают $a_{n,\,p}^{(p)}$ и т. п). После ввода последнего элемента контрольного столбца $a_{n,\,n+1}^{(p-1)}$ в бланк записывают индицируемое значение $a_{n-1,\,n+1}^{(p)}$, а затем нажимают клавишу С/П и регистрируют вычисляемое значение $a_{n,\,n+1}^{(p)}$.

В случае ошибки достаточно ввести в регистр X первый элемент a_{1j}^{p-1} соответствующего столбца, нажать клавиши В/О и С/П ($PX = 0$) и повторить вычисление следующих элементов. Правильность вычислений контролируют по совпадению (с точностью до операционных погрешностей) вычисленных элементов контрольного столбца с суммами остальных элементов (включая приписанную единицу) соответствующих строк.

Если $n < 11$, то следует разбить каждую часть бланка на блоки с 11 или менее строками и преобразовать коэффициенты каждого блока в соответствии с описанной методикой. Для уменьшения погрешности вычисления корней целесообразно так переставить уравнения системы, чтобы элементы $a_{1p}^{(p-1)}$ были возможно большими.

Большинство методов решения систем нелинейных уравнений

$$f_i(x_1, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, n \tag{5.20}$$

связано с необходимостью предварительной оценки условий сходимости вычислительного процесса для выбранных начальных приближений x_i. При выполнении этих условий для метода Ньютона, обеспечивающего относительно быстрый поиск корней, его целесообразно использовать при решении систем (5.20) с небольшим числом уравнений.

В этом случае, например, для системы из двух уравнений $f(x, y) = 0$, $\varphi(x, y) = 0$ последние после оценки сходимости [1] представляют разложением левых частей в ряд Тейлора с сохранением линейных частей и на каждой итерации находят поправки очередных приближений

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \begin{bmatrix}\partial f/\partial x & \partial f/\partial y\\ \partial\varphi/\partial x & \partial\varphi/\partial y\end{bmatrix}^{-1}\begin{bmatrix}f(x, y)\\ \varphi(x, y)\end{bmatrix}.$$

Вычисления прекращают, когда невязки уравнений становятся меньше наперед заданных малых величин.

При программной реализации метода Ньютона для ее упрощения целесообразно заменить аналитическое дифференцирование численным. Вычисляя обе функции в трех точках $f_1 = f(x, y)$, $\varphi_1 = \varphi(x, y)$, $f_2 = f(x + \delta, y)$, $\varphi_2 = \varphi(x + \delta, y)$, $f_3 = f(x + \delta, y + \delta)$, $\varphi_3 = \varphi(x + \delta, y + \delta)$, для достаточно малых значений δ можно принять

$$\partial f/\partial x = (f_2 - f_1)/\delta, \quad \partial f/\partial y = (f_3 - f_2)/\delta;$$
$$\partial\varphi/\partial x = (\varphi_2 - \varphi_1)/\delta, \quad \partial\varphi/\partial y = (\varphi_3 - \varphi_2)/\delta.$$

Подставляя эти значения в уравнения для поправок, получим

$$\Delta x \approx \delta \frac{\varphi_3 (f_3 - f_2) - f_3 (\varphi_3 - \varphi_2)}{(f_2 - f_1)(\varphi_3 - \varphi_2) - (f_3 - f_2)(\varphi_2 - \varphi_1)};$$

$$\Delta y \approx \delta \frac{f_3 (\varphi_2 - \varphi_1) - \varphi_3 (f_2 - f_1)}{(f_2 - f_1)(\varphi_3 - \varphi_2) - (f_3 - f_2)(\varphi_2 - \varphi_1)}.$$

Знаменатели этих выражений можно упростить, но это может привести к потере точности вследствие вычитания близких чисел. Поэтому в приведенной ниже программе эти выражения реализованы в представленном виде.

Программа 47/34. Решение системы из двух нелинейных уравнений методом Ньютона

ПП	68	П4	ИПД	П3	С/П	ИП1	ИП0	+	П1
ПП	68	П6	ИП4	—	П4	ИПД	П5	ИП3	—
П3	ИП2	ИП0	+	П2	ПП	68	П7	ИПД	ИП5
—	П5	×	ИП7	ИП6	—	П6	ИПД	×	—
ИП3	ИП6	×	ИП4	ИП5	×	—	ИП0	÷	П5
÷	ИП1	+	П1	ИПД	ИП4	×	ИП3	ИП7	×
—	ИП5	÷	ИП2	+	П2	БП	00	...	В/О

Инструкция: заменить в программе многоточие фрагментом вычисления функций $f(x, y)$ и $\varphi(x, y)$ при $x = P1$, $y = P2$ с занесением невязок f и φ соответственно в регистры $Д$ и X (при вычислении невязок могут быть использованы регистры $7, \ldots, C$); $\delta = P0$, начальные приближения $x_0 = P1$, $y_0 = P2$ (ввести также в соответствующие регистры коэффициенты для вычисления невязок) В/О С/П $PX = f^{(0)}(x, y)$, $PY = \varphi^{(0)}(x, y)$ С/П $PX = f^{(1)}(x, y)$, $PY = \varphi^{(1)}(x, y)$ С/П $PX = f^{(2)}(x, y)$, $PY = \varphi^{(2)}(x, y)$ ... Повторив итерации до получения невязок требуемой малости, значения корней вызывают соответственно из регистров *1* и *2*.

Точность решения зависит от вида уравнений и выбранного значения δ — если оно мало, то точность вычисления производных может казаться малой или возникнет переполнение при их близости к нулю, если велико, то точность решения снижается вследствие нелинейности уравнений.

В качестве примера рассмотрим решение системы уравнений

$$\sqrt{(x(y+5)-1)/2} - x = 0, \quad \sqrt{x + 3\ln x} - y = 0,$$

вычисление невязок которых реализуется подпрограммой

ИП1	ИП2	5	+	×	1	—	2	÷	$\sqrt{}$
ИП1	—	ПД	ИП1	ln	3	×	ИП1	+	$\sqrt{}$
ИП2	—								

При начальных значениях $x_0 = 4$; $y_0 = 2{,}5$; $\delta = 10^{-5}$ получим невязки $f^{(0)} = -0{,}1921135$; $\varphi^{(0)} = 0{,}3563758$. После первой итерации $f^{(1)} = -4{,}3813 \cdot 10^{-3}$; $\varphi^{(1)} = -3{,}2787 \cdot 10^{-3}$ (что свидетельствует о достаточно быстрой сходимости); $x^{(1)} = 3{,}7697783$; $y^{(1)} = 2{,}7873093$ (время счета около 45 с). Выполнив еще две итерации, получим $f^{(2)} = -9{,}45 \cdot 10^{-5}$; $\varphi^{(2)} = -3{,}8 \cdot 10^{-5}$; $x = 3{,}757093$; $y =$

$= 2{,}7799713$ и $f^{(3)} = -7 \cdot 10^{-7}$; $\varphi^{(3)} = -9 \cdot 10^{-7}$; $x = 3{,}756836$; $y = 2{,}7798511$. Таким образом, уже после первых итераций получена точность корней в четыре верных цифры по сравнению с методом итераций Зейделя, обеспечившим после шести итераций только три верные цифры [11].

При использовании микрокалькуляторов с входным языком ЯМК 21 приходится прибегать к более простым методам, например, итерационному методу Зейделя с более «жесткими» условиями сходимости [4]. При необходимости программные реализации таких методов несложно составить и на входном языке ЯМК34.

Корни многих систем с большим числом нелинейных уравнений удобно вычислять методом простых итераций [4] с предварительным преобразованием системы (5.20) к виду $x_i = \varphi_i(x_1, \ldots, x_n)$, $i = 1,2, \ldots, n$ и последовательным вычислением приближений корней по формуле

$$x_i^{(k+1)} = \varphi_i(x_1^{(k)}, \ldots, x_n^{(k)}), \quad i = 1,2, \ldots, n; \quad k = 0, 1, 2, \ldots$$

до совпадения очередных значений с требуемым числом цифр.

Программная реализация этого метода упрощается при вычислении на каждой итерации очередной переменной для значений предыдущих переменных, вычисленных на этой же итерации. Например, представив уравнения из примера к программе 47/34 в форме $x = \sqrt{(x(y+5)-1)/2}$ и $y = \sqrt{x + 3\ln x}$, несложно составить для метода простых итераций программу

ИП2	5	+	ИП1	×	1	—	2	÷	√
П1	ln	3	×	ИП1	+	√	П2	ИП1	С/П
БП	00								

с инструкцией: $x^{(0)} = P1$, $y^{(0)} = P2$ (В/О) С/П $PX = P1 = x^{(k)}$, $PY = P2 = y^{(k)}$. При $x^{(0)} = 1$, $y^{(0)} = 0$ по этой программе значения корней с четырьмя верными цифрами вычисляются после 23 итераций.

Простота реализации метода простых итераций и возможность решения систем с большим числом уравнений (n ограничивается лишь емкостью программной памяти, так как емкость числовой памяти микрокалькуляторов с входным языком ЯМК34 допускает решение систем из $n \leqslant 14$ нелинейных уравнений) позволяет во многих случаях избежать предварительной оценки условий сходимости, проверяя ее для выбранных начальных приближений в процессе вычислений. Однако для некоторых систем уравнений условия сходимости рассмотренных методов оказываются чрезмерно жесткими. Так, вычисления по методу простых итераций для несложной системы уравнений

$$x_1^2 + x_2^2 = 1, \quad x_1^3 - x_2 = 0$$

сходятся только при выборе начальных приближений в ближайших окрестностях корней.

Более общий подход к решению систем нелинейных уравнений основан на использовании методов нелинейного программирования (гл. 8).

4. ОПЕРАЦИИ НАД МАТРИЦАМИ

Решение многих инженерных задач связано с операциями над упорядоченными в виде таблиц (матриц) множествами чисел. Полная автоматизация таких операций с помощью микрокалькуляторов возможна лишь для матриц небольшого размера, и в общем случае приходится выполнять такие операции по частям или в полуавтоматическом режиме с регистрацией промежуточных результатов в вычислительных бланках.

Простейшими операциями являются суммирование (сложение или вычитание) и умножение матриц. Сумма $C = A + B$ двух матриц одинакового размера равна матрице такого же размера с элементами $c_{ij} = a_{ij} + b_{ij}$. Суммирование матриц несложно выполнить в обычном режиме, но затраты времени уменьшаются при использовании режима автоматических вычислений. Для полного использования ресурса памяти целесообразно заносить в нее элементы одной из суммируемых матриц, а элементы второй вводить в регистр X с последующим суммированием. Примером подобной реализации суммирования матриц может служить следующая программа на входном языке ЯМК34:

```
КИП0 + ИП0 1 + П0 → КП0 ИП0 С/П
БП   00
```

Перед выполнением этой программы в регистр *0* заносят число элементов (не более 13) суммируемых матриц (одинаковые элементы матриц с нулевым значением можно не заносить), значения элементов a_{ij} первой матрицы вводят в регистры *1*, *2*, ..., *Д*. После этого в регистр X вводят элементы b_{ij} второй матрицы в порядке, соответствующем хранимым в памяти (начиная с регистра с наибольшим номером) элементам первой матрицы с нажатием после каждого ввода клавиши С/П (первый раз — В/О и С/П). После каждого выполнения программы в соответствующем очередном регистре значение a_{ij} заменяется суммой $a_{ij} + b_{ij}$, а на индикаторе высвечивается номер (код) регистра, содержимое которого должно суммироваться следующим.

Если число элементов матриц больше 13, то их можно разбить на части, содержащие не более 13 элементов, и выполнить суммирование для каждой части. Для суммирования элементов матриц размера 2×2 или 3×3 элементы первой матрицы удобно размещать в регистрах, клавиши набора номеров которых расположены аналогично соответствующим элементам матрицы. Специальные программы суммирования в этом случае особенно удобны при сложении большого числа матриц одинакового размера. Для сложения матриц с комплексными элементами алгебраическое суммирование элементов выполняется дважды — для их вещественных и мнимых частей.

Произведение $C = A \times B$ двух матриц размера $m \times p$ и $p \times n$ равно матрице размера $m \times n$ с элементами

$$c_{ij} = \sum_{k+1}^{p} a_{ik} b_{kj}.$$

Умножение матриц с вещественными элементами, для которых $mn \leqslant 12$, несложно выполнить в соответствии с этой формулой при предварительном занесении в память элементов матрицы A и последовательным вводом элементов столбцов матрицы B с вычислением элементов столбцов произведения C после каждого ввода. Примером может служить следующая программа.

Программа 48/34. Умножение вещественных матриц размером 3×3

```
ИП1 ИП0 ×   ИП2 ИПА ×   +   ИП3 ИПВ ×
+   ПД  ИП4 ИП0 ×   ИП5 ИПА ×   +   ИП6
ИПВ ×   +   ПС  ИП7 ИП0 ×   ИП8 ИПА ×
+   ИП9 ИПВ ×   +   С/П БП  00
```

Инструкция: ($a_{11}= P7, a_{12}= P8, a_{13}= P9, a_{21}= P4, a_{22}= P5, a_{23}= P6, a_{31}= P1, a_{32}= P2, a_{33}= P3$) $b_{11}= P0$, $b_{21}= PA$, $b_{31}= PB$ В/О С/П $PX = c_{11}$, $PC = c_{21}$, $PД = c_{31}$, $b_{12}= P0$, $b_{22}= PA$, $b_{32}= PB$ С/П $PX = c_{12}$, $PC = c_{22}$, $PД = c_{32}$, $b_{13}= P0$, $b_{23}= PA$, $b_{33}= PB$ С/П $PX = c_{13}$, $PC = c_{23}$, $PД = c_{33}$.

Для вычисления элементов произведения матриц бо́льших размеров следует составлять программы, обеспечивающие автоматизацию вычислений произведений элементов строк первой матрицы на элементы столбцов второй матрицы в соответствии с приведенной формулой.

Матрицы с комплексными элементами содержат в общем случае удвоенное число составляющих и возможность полной автоматизации операций над такими матрицами еще более ограничена. В частности, каждый комплексный элемент произведения двух комплексных матриц определяется через вещественные и мнимые составляющие исходных матриц выражением

$$\operatorname{Re} c_{ij} + \mathrm{j} \operatorname{Im} c_{ij} = c_{rij} + \mathrm{j}\, c_{iij} = \sum_{k=1}^{p} ((a_{rik} b_{rkj} - a_{iik} b_{ikj}) + \mathrm{j}\,(a_{rik} b_{ikj} + a_{iik} b_{rkj})).$$

На входном языке ЯМК34 полностью автоматизировать вычисление произведения двух комплексных матриц удается, если каждая из них содержит не более 6 комплексных элементов (имеет размер 2×3 или 3×2). В частности, для умножения двух комплексных матриц размером 2×2 можно воспользоваться следующей программой.

Программа 49/34. Умножение комплексных матриц размером 2×2

```
ИП1 ИП9 ×   ИП0 ИП6 ×   —   ИП2 ИП3 ×
+   ИПА ИПВ ×   —   ПС  ИП1 ИП6 ×   ИП0
ИП9 ×   +   ИП2 ИПВ ×   +   ИПА ИП3 ×
+   ПД  ИП7 ИП6 ×   ИП4 ИП9 ×   +   ИП8
ИПВ ×   +   ИП6 ИП3 ×   +   ИП7 ИП9 ×
ИП4 ИП6 ×   —   ИП8 ИП3 ×   +   ИП5 ИПВ
×   +   С/П БП  00
```

Инструкция: ($a_{r11}=P7$, $a_{i11}=P4$, $a_{r12}=P8$, $a_{i12}=P5$, $a_{r21}=P1$, $a_{i21}=P0$, $a_{r22}=P2$, $a_{i22}=PA$) $b_{r11}=P9$, $b_{i11}=P6$, $b_{r21}=P3$, $b_{j21}=$ $=PB$ В/О С/П $PX=c_{r11}$, $PY=c_{i11}$, $PC=c_{r21}$, $PД=c_{i21}$, $b_{r12}=P9$, $b_{i12}=P6$, $b_{r22}=P3$, $b_{i22}=PB$ С/П $PX=c_{r12}$, $PY=c_{i12}$, $PC=c_{r22}$, $PД=c_{i22}$.

Часто встречающейся и относительно громоздкой операцией является вычисление определителя квадратной матрицы. Сравнительно просто удается автоматизировать вычисление определителя матриц второго и третьего порядков (см. приложение), но для комплексных матриц даже в этом случае вычисление определителя достаточно громоздко, о чем свидетельствует и следующая программа.

Программа 50/34. Вычисление определителя комплексной матрицы третьего порядка

ПП	40	Cx	XY	—	ПП	40	ИП0	ИП2	П0
XY	ИП4	П2	XY	П4	ИП1	ИП3	П1	XY	ИП5
П3	XY	П5	ИП6	ИП8	П6	XY	ИПА	П8	XY
ПА	ИП7	ИП9	П7	XY	ИПВ	П9	XY	ПВ	С/П
ИПД	ИПС	ПД	→	ПС	→	XY	↑	ИП2	×
ИПА	ПП	80	ИП3	×	ИПА	ПП	87	ИП3	ПП
85	ИП2	ПП	85	ИП8	×	ИП4	ПП	87	ИП9
×	ИП4	ПП	80	ИП9	ПП	78	ИП8	×	ИП5
×	ИПД	+	БП	91	×	ИПВ	×	ИПД	XY
—	ПД	→	XY	↑	В/О				

Инструкция: $\mathrm{Re}\,a_{21}=P0$, $\mathrm{Im}\,a_{21}=P1$, $\mathrm{Re}\,a_{22}=P2$, $\mathrm{Im}\,a_{22}=P3$, $\mathrm{Re}\,a_{23}=P4$, $\mathrm{Im}\,a_{23}=P5$, $\mathrm{Re}\,a_{31}=P6$, $\mathrm{Im}\,a_{31}=P7$, $\mathrm{Re}\,a_{32}=P8$, $\mathrm{Im}\,a_{32}=P9$, $\mathrm{Re}\,a_{33}=PA$, $\mathrm{Im}\,a_{33}=PB$, $0=PC=PД$, $\mathrm{Re}\,a_{11}=PY$, $\mathrm{Im}\,a_{11}=PX$ В/О С/П $\mathrm{Re}\,a_{12}=PY$, $\mathrm{Im}\,a_{12}=PX$ В/О С/П $\mathrm{Re}\,a_{13}=$ $=PY$, $\mathrm{Im}\,a_{13}=PX$ В/О С/П $PC=\mathrm{Re}\,\Delta$, $PД=\mathrm{Im}\,\Delta$. Время счета после каждого пуска около 1,5 мин (всего около 4,5 мин)

Контрольный пример:

$$\Delta=\begin{vmatrix} 1+j2 & 3+j5 & j10 \\ 3+j1 & 4+j5 & 6 \\ 4+j3 & j6 & 3-j2 \end{vmatrix}=154+j73.$$

Для полного использования ресурсов памяти микрокалькулятора, входной язык которого содержит операторы косвенного обращения к памяти, при вычислении определителей вещественных матриц целесообразно модифицировать соответствующим образом метод оптимального исключения, рассмотренный в предыдущем разделе. Такая модификация реализована в следующей программе, непосредственно пригодной для вычисления определителей матриц порядка $n \leqslant 5$ и, при использовании описанной ниже процедуры, определителей матриц шестого порядка.

Программа 51/34. Вычисление определителей вещественных матриц порядка $n \leqslant 5$

П4	1	4	П2	КИП0	ИП0	П1	С/П	ИП4	÷
КП2	L1	07	1	4	П3	ИП0	П1	Сх	КП2
L1	19	ИП0	ИП2	+	П1	П2	ИП3	—	$x\neq 0$
36	С/П	ПП	83	БП	23	КИП0	ИП4	С/П	КИП2
—	×	П4	ИП0	П3	$x\neq 0$	37	Вх	С/П	КИП2
—	XY	÷	КП1	L3	47	ИП1	ИП0	+	П3
1	4	П1	П2	КИП1	/—/	ПП	83	ИП3	+
П3	ИП1	—	$x=0$	64	ИП0	П1	КИП3	КП2	L1
77	БП	13	ИП0	XY	↑	КИП3	×	КИП1	+
КП2	→	L0	85	→	П0	В/О			

Инструкция: $n = P0$, $a_{11} = PX$ В/О С/П $a_{12} = PX$ С/П ... $a_{1n} = PX$ С/П $a_{21} = PX$ С/П ... $a_{nn} = PX$ С/П $PX = \Delta$. Необходимо учитывать, что при вводе элементов матрицы нельзя изменять содержимое остальных регистров операционного стека, поэтому для нулевых элементов следует вводить 0, а при исправлении ошибок очищать регистр X только вводом оператора Сх.

Контрольные примеры:

$$\Delta = \begin{vmatrix} 10 & 4 \\ 5 & 6 \end{vmatrix} = 40$$ (время счета около 23 с);

$$\Delta = \begin{vmatrix} 1 & 5 & 6 \\ 2 & -1 & 4 \\ 3 & 3 & 3 \end{vmatrix} = 69{,}000001$$ (время счета около 72 с);

$$\Delta = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{vmatrix} = -60$$ (время счета около 140 с);

$$\Delta = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 3 & 4 & 5 \\ 3 & 2 & 1 & 4 & 5 \\ 4 & 3 & 2 & 1 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{vmatrix} = 360{,}00001$$ (время счета около 260 с).

Программу 51/34 можно использовать также для вычисления определителя матрицы шестого порядка согласно следующей процедуре:

1. Вычеркивая первую строку и по одному столбцу исходной матрицы, вычислить по программе определители полученных матриц пятого порядка, равных минорам $M_{11}, M_{12}, \ldots, M_{16}$ исходной матрицы.

2. В соответствии с теоремой Лапласа в ручном режиме (или с помощью несложной программы автоматических вычислений) вычислить искомый определитель $\Delta = a_{11}M_{11} - a_{12}M_{12} + a_{13}M_{13} - a_{14}M_{14} + a_{15}M_{15} - a_{16}M_{16}$.

Подобный метод достаточно громоздок и для матриц высокого порядка целесообразно использовать рассматриваемую далее методику вычисления определителей. Кроме того, последняя программа и аналогичные ей не обеспечивают текущего контроля погрешностей промежуточных результатов вычислений.

Между тем при вычислении определителей и решении систем линейных уравнений вычитание близких по величине больших чисел сопровождается значительными операционными погрешностями, накопление которых может привести к ошибочному результату вычислений. Поэтому при вычислении определителей матриц высокого порядка и систем из большого числа линейных уравнений необходимо контролировать точность промежуточных результатов вычислений. Это относится как к различным вариантам метода исключений Гаусса, так и к использованию различного рода разложений, например по теореме Лапласа или формулам Крамера.

Для определителей матриц порядка $n > 3$ можно использовать полуавтоматический режим вычислений с регистрацией промежуточных результатов в бланке. В этом случае целесообразно воспользоваться схемой единственного деления Гаусса с последовательным заполнением n частей вычислительного бланка [11]. Для сокращения затрат времени на решение задачи элементы «ведущей» строки в бланк записывать не будем, вычисляя регистрируемые элементы p-й части бланка по элементам $(p-1)$-й части согласно формуле

$$a_{ij}^{(p)} = a_{i-1,\,j}^{(p-1)} - a_{i-1,\,p-1}^{(p-1)}\, a_{1j}^{(p-1)} / a_{1,\,p-1}^{(p-1)},\quad i \geqslant 1,\ j \geqslant p \qquad (5.21)$$

с хранением второго члена правой части формулы в памяти микрокалькулятора. В этом случае можно использовать бланк упрощенной формы с меньшим (по сравнению с обычным бланком) числом строк.

В n столбцов первой части такого бланка (табл. 13) заносят элементы матрицы, а в последний контрольный столбец — суммы элементов каждой строки матрицы. Вычисление элементов каждой последующей части бланка по элементам предыдущей части согласно формуле (5.21) упрощается при использовании следующей программы.

Программа 52/34. Вычисление определителя матрицы порядка $n \leqslant 10$ по схеме единственного деления

ПВ	ИПС	×	ПС	ИПД	1	—	ПД	П0	ИПС
С/П	ИПВ	÷	КП0	ИП0	1	—	$x=0$	10	С/П
ПА	ИПД	П0	С/П	КИП0	ИПА	×	—	ИП0	1
—	$x=0$	36	→	БП	19	→	БП	23	

Инструкция: 1) $n+1 = РД$, $1 = РС$; 2) $a_{1,\,p-1}^{(p-1)} = РХ$ В/О С/П $РХ = a_{11}$; 3) ввести в регистр X остальные элементы первого столбца $(p-1)$-й части с нажатием после каждого ввода только клавиши С/П с контролем по высвечиваемым значениям $РХ = n - i - 1$; 4) для каждого j-го столбца $(p-1)$-й части выполнить $a_{1j}^{(p-1)} = РХ$ С/П $РХ = n + 1 - p$, после чего ввести остальные элементы этого столбца с нажатием после каждого ввода только клавиши С/П и регистрацией в р-й части бланка высвечиваемых значений $a_{ij}^{(p)}$;

13. Вычисление определителя по схеме единственного деления Гаусса с помощью программы 52/34

A

p	$j=1$	$j=2$	$j=3$	$j=4$	Σ
1	1 2 3 4	2 1 2 3	3 3 1 2	4 4 4 1	10 10 10 10
2		—3 —4 —5	—3 —8 —10	—4 —8 —15	—10 —20 —30
3			—4,0000001 —5	—2,6666668 —8,333334	—6,6666667 —13,333334
4				—5,0000008	—5,0000004

$\Delta = 60{,}000009$

B

p	$j=1$	$j=2$	$j=3$	$j=4$	Σ
1	4 1 2 3	3 2 1 2	2 3 3 1	1 4 4 4	10 10 10 10
2		1,25 —0,5 —0,25	2,5 2 —0,5	3,75 3,5 3,25	7,5 5 2,5
3			3 0	5 4	8 4
4				4	4

$\Delta = 60$

5) после заполнения p-й части бланка (включая контрольный столбец) повторить выполнение инструкции с п. 2 до заполнения всех n частей бланка; 6) после вычисления значения $a^{(n)}_{1,\,n+1}$ в контрольном столбце нажать клавиши В/О и С/П и зарегистрировать высвечиваемое значение Δ. Если $a^{(n)}_{1,\,n+1}$ не совпадает со значением $a^{(n)}_{1n}$ с достаточной точностью, то следует после вычисления $a^{(n)}_{1,\,n+1}$ набрать в регистре X значение $a^{(n)}_{k1}$ и нажать клавиши В/О и С/П для индикации значения определителя.

Так как в схеме единственного деления согласно формуле (5.21) может возникнуть значительная операционная погрешность при малых значениях $a_{1p}^{(p)}$, то для повышения точности результата целесообразно переставить строки или столбцы исходной матрицы так, чтобы значения $a_{1p}^{(p)}$ были по возможности бóльшими. Например, при вычислениях определителя матрицы, записанной в первой части табл. 13А, получим $\Delta = 60{,}000009$, тогда как при предварительной перестановке строк матрицы (табл. 13В) получим точное значение $\Delta = 60$. С помощью программы 52/34 можно вычислять и определители матриц порядка $n > 10$, если разбить строки таких матриц на блоки с числом строк $m \leqslant 10$ и вычислять преобразованные элементы для каждого блока в отдельности.

В некоторых задачах приходится вычислять элементы матрицы $B = A^{-1}$, обратной неособенной (с отличным от нуля определителем) матрицы A. Элементы обратной матрицы B выражаются через определитель Δ и алгебраические дополнения Δ_{ij} элементов матрицы A согласно формуле

$$B = \begin{bmatrix} \Delta_{11}/\Delta & \Delta_{21}/\Delta & \dots & \Delta_{n1}/\Delta \\ \Delta_{12}/\Delta & \Delta_{22}/\Delta & \dots & \Delta_{n2}/\Delta \\ \dots & \dots & \dots & \dots \\ \Delta_{1n}/\Delta & \Delta_{2n}/\Delta & \dots & \Delta_{nn}/\Delta \end{bmatrix}.$$

Обращение матриц второго и третьего порядков с вещественными элементами несложно автоматизировать в соответствии с этой формулой, но при $n > 3$ вычисление определителя и алгебраических дополнений матрицы A связано со значительными затратами времени. В этих случаях можно учесть, что, в соответствии с формулой (5.17), при всех нулевых, кроме $q_k = 1$, свободных членах решение системы уравнений $AX = Q$ равно множеству элементов k-го столбца обратной A матрицы

$$x_i = b_{ik} = \Delta_{ki}/\Delta,\ i = 1,2, \dots, n.$$

Поэтому в тех случаях, когда требуется определить один или несколько элементов k-го столбца матрицы, достаточно использовать любой метод решения системы линейных уравнений при нулевых свободных членах, кроме $q_k = 1$. Однако, если требуется вычислить все элементы обратной матрицы, то решение системы линейных уравнений приходится находить n раз при $k = 1, 2, \dots, n$.

Затраты времени можно уменьшить, дополнив преобразование матрицы A в процессе исключения переменных преобразованием единичной матрицы порядка n. Например, при использовании метода Жордана—Гаусса следует заменить в вычислительном бланке столбец свободных членов единичной матрицей (табл. 14). Для этого можно воспользоваться и схемой единственного деления, но по окончании ее прямого хода будут вычислены только n элементов обратной матрицы, и остальные ее элементы приходится вычислять по формулам обратного хода [4].

14. Вычисление обратной матрицы методом Жордана—Гаусса с циклической перестановкой строк

p	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	Σ
0	1,8	—3,8	0,7	—3,7	1	0	0	0	—4
	0,7	2,1	—2,6	—2,8	0	1	0	0	—3,6
	7,3	8,1	1,7	—4,9	0	0	1	0	13,2
	1,9	—4,3	—4,9	—4,7	0	0	0	1	—11
1		3,5777778	—2,8722222	—1,3611111	—0,38888889	1	0	0	—0,044445
		23,511111	—1,1388888	10,105555	—4,0555555	0	1	0	29,422222
		—0,2888889	—5,6388889	—0,7944446	—1,0555555	0	0	1	—6,777778
	1	—2,1111111	0,38888888	—2,0555555	0,55555555	0	0	0	—2,2222222
2			17,735714	19,049999	—1,5000001	—6,5714284	1	0	29,130158
			—5,8708075	—0,90434802	—1,0869565	0,080745371	0	1	—6,7741893
		1	—1,3059006	—2,8586956	0,32608696	0,59006209	0	0	—2,1959972
		1	—0,80279501	—0,38043477	—0,10869565	0,2795031	0	0	0,012422575
3				5,4015086	—1,5834807	—2,0945034	0,33101613	1	2,8683625
			1	—1,4560229	0,21564026	0,10620043	0,073631126	1	—0,051111
			1	0,4818504	—0,17659213	—0,01794812	0,045261126	0	1,3309791
			1	1,0741038	—0,084575117	—0,37051952	0,05638408	0	1,6424575
4				1	—0,21120051	—0,45839089	0,16285936	0,26955855	0,72208066
				1	—0,03533515	0,16889548	0,015735485	—0,089206631	1,0751021
				1	0,23030405	0,04597783	—0,009439323	—0,19885255	1,0720763
				1	—0,29315526	—0,38776267	0,061282162	0,1851346	0,53102988

5. РЕШЕНИЕ ЗАДАЧ В ОБЩЕМ ВИДЕ

Практическое решение задач обычно связано с представлением исходной математической модели в виде системы уравнений с буквенными обозначениями физических величин и преобразованием такой модели к формулам, связывающим в явном виде исходные данные с искомыми результатами решения задачи. Такие формулы используют для вычислений, но их основное назначение заключается в представлении зависимости искомых результатов от исходных данных в *символьном* (буквенном) виде, позволяющем качественно оценить влияние различных исходных данных на результаты решения задачи.

Основной формой представления исходных условий задачи является система в общем случае операторных уравнений (2.6). Поэтому решение задачи в общем (символьном) виде заключается в составлении формул (5.17) с выражением определителя и алгебраических дополнений матрицы A через символы элементов этой матрицы. В случае нелинейных или дифференциальных уравнений элементы матрицы A являются операторами, которые должны быть выражены через символы других переменных в соответствии с математическими моделями этих операторов.

Алгебраические дополнения $\Delta_{ji} = (-1)^{j+i} M_{ji}$ только знаком отличаются от миноров M_{ji}, равных определителям матрицы A с вычеркнутыми j-й строкой и i-м столбцом. Поэтому решение задачи в символьном виде с точностью до символов элементов матрицы A заключается в разложении определителей по этим символам. Для этого удобно воспользоваться алгоритмами метода обобщенных чисел [10], основанными на отображении структуры матрицы множеством индексов ml ее ненулевых элементов a_{ml}. Для этого матрица представляется частично-упорядоченным множеством (*матричным числом*) β_M из строк, в каждой m-й из которых содержится множество номеров l столбцов, на пересечении которых с m-й строкой матрицы расположены ненулевые элементы a_{ml}.

Определитель матрицы порядка n содержит до $n!$ членов в виде произведений n элементов матрицы, взятых по одному от каждой строки и каждого столбца, причем у отрицательных членов подстановка индексов нечетная. Следовательно, каждый член определителя соответствует подстановке с различными индексами m и различными индексами l. Если, сохранив порядок номеров m строк матричного числа, выбрать из каждой строки элементы, образующие множество различных индексов l, то получим множество подстановок членов определителя (*контурное число*) β. Оно связано с матричным числом и определителем формулой

$$\Delta = \det \beta = \det (\beta_M)_{\mathrm{mod}\, 2},$$

где индекс mod 2 означает [10], что матричное число преобразуется в контурное умножением элементов строк над полем модуля 2.

Эту формулу удобно реализовать согласно следующему алгоритму:

1. Представить матрицу A порядка n матричным числом с n строками.

2. Принять $m = 1$.

3. Под каждым выписанным элементом m-й строки матричного числа провести линию, под которой выписать элементы $(m-1)$-й строки, отличающиеся от отделенных линиями верхних элементов.

4. Принять $m = m + 1$.

5. Если $m = n$, то перейти к п. 6, иначе к п. 3.

6. Изъять неполные столбцы и линии, отделяющие по одному элементу, и подчеркнуть столбцы (содержащие элемент n-й строки и расположенные над ним элементы остальных строк) с нечетным числом инверсий.

7. Заменить каждый столбец контурного числа с элементами α_m произведением элементов $a_{m\alpha}$ исходной матрицы A, приписав отрицательные знаки членам определителя, соответствующим подчеркнутым столбцам; линии, отделяющие общие элементы столбцов, соответствуют скобкам, отделяющим общие множители в определителе.

В качестве примера рассмотрим составление математической модели неизвестной x_4 (учитывая, что для вычисления минора M_{pq} по приведенному алгоритму достаточно предварительно вычеркнуть в матричном числе p-ю строку и элементы $\alpha_m = q$) по системе уравнений

$$\begin{vmatrix} a_{11} & 0 & a_{13} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 & a_{26} \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ a_{41} & 0 & a_{43} & a_{44} & 0 & 0 \\ a_{51} & 0 & a_{53} & a_{54} & a_{55} & 0 \\ a_{61} & 0 & a_{63} & a_{64} & 0 & a_{66} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}.$$

В соответствии с формулой (5.17) искомая переменная выражается через алгебраическое дополнение и определитель соотношением $x_4 = \Delta_{14}/\Delta$. Для разложения определителя Δ по символам элементов матрицы A согласно описанному алгоритму последовательно находим

$$\Delta = \det \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{vmatrix} 1 & 3 & & \\ 1 & 2 & 6 & \\ 2 & 3 & 4 & \\ 1 & 3 & 4 & \\ 1 & 3 & 4 & 5 \\ 1 & 3 & 4 & 6 \end{vmatrix}_{\mathrm{mod}\,2} = \det \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{vmatrix} 1 & & 3 & \\ \underline{2} & \underline{6} & \overline{1} & \overline{2\ 6} \\ \underline{3\ 4} & \underline{2} & \underline{2} & \overline{4\ 2} \\ 4 & 3 & \underline{3\ 4} & \overline{4\ 1\ 1\ 4} \\ 5 & 5 & 5 & 5\ 5\ 5\ 5\ 5 \\ 6 & \underline{6} & \underline{4} & 3\ 6\ 6\ \underline{4\ 1} \end{vmatrix} =$$

$$= a_{11}(a_{22}(a_{33}a_{44}a_{55}a_{66} - a_{34}a_{43}a_{55}a_{66}) + a_{26}a_{32}(a_{44}a_{55}a_{63} - a_{43}a_{55}a_{64})) + a_{13}(a_{21} \times$$
$$\times a_{32}a_{44}a_{55}a_{66} + a_{22}a_{34}a_{41}a_{55}a_{66} - a_{26}a_{32}(a_{41}a_{55}a_{64} + a_{44}a_{55}a_{61})).$$

Согласно тому же алгоритму после вычеркивания в матричном числе первой строки и элементов 4 находим

$$\Delta_{14} = -M_{14} = -\det \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{vmatrix} 1 & 2 & 6 \\ 2 & 3 & \\ 1 & 3 & \\ 1 & 3 & 5 \\ 1 & 3 & 6 \end{vmatrix}_{\mathrm{mod}\,2} = -\det \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \begin{vmatrix} 1 & 2 & 6 & \\ 2 & 3 & \overline{2} & \\ 3 & 1 & \underline{1}\ \overline{3} & \\ 5 & 5 & 5 & 5 \\ 6 & \underline{6} & \underline{3} & \underline{1} \end{vmatrix} =$$

$$= -a_{21}a_{32}a_{43}a_{55}a_{66} - a_{22}a_{33}a_{41}a_{55}a_{66} + a_{26}a_{32}(a_{11}a_{55}a_{63} + a_{43}a_{55}a_{61}).$$

Таким образом, символьное разложение определителей сводится к простейшим операциям сравнения цифр, причем очередное значение

α_m отбрасывается, если оно совпадает с одним из ранее выбранных элементов формируемого столбца контурного числа. Несмотря на простоту этих операций формирование контурных чисел целесообразно автоматизировать, так как при выполнении описанного алгоритма с помощью карандаша и бумаги возможны ошибки, устранение которых связано со значительными затратами времени.

Алгоритмы метода обобщенных чисел относительно просто реализуются на универсальных ЭВМ [14] и задача программирования в основном сводится к выбору наиболее быстрого способа определения принадлежности очередного элемента строки матричного числа к формируемому множеству элементов столбца контурного числа. Решение этой задачи с помощью программируемых микрокалькуляторов с входными языками ЯМК21 и ЯМК34 ограничено малой емкостью числовой памяти, недостаточной для размещения элементов строк матричного числа и столбцов контурного числа в отдельных регистрах. Однако можно снизить это ограничение, записывая при $n < 10$ однозначные элементы (цифры) строк матричного и столбцов контурного чисел слитно (как записываются многозначные числа) и храня множества этих элементов в отдельных регистрах. Требования к памяти в этом случае существенно снижаются, но предельный порядок исходных матриц ограничивается разрядностью r мантиссы на индикаторе.

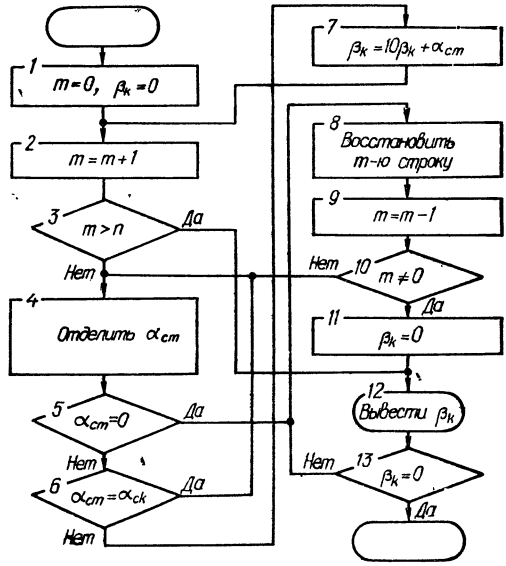


Рис. 35

Обозначив символом α_{cm} элемент строки матричного числа с номером m, выделяемый для сравнения с элементами формируемого множества β_k элементов столбца контурного числа и дополняющий это множество, если в нем отсутствует такой элемент, опишем процесс формирования контурного числа (без учета числа инверсий в его столбцах) следующим алгоритмом, схема которого показана на рис. 35:

1. Принять $m = 0$, $\beta_k = 0$.
2. Принять $m = m + 1$.
3. Если $m > n$, то перейти к п. 12, иначе к п. 4.
4. Отделить очередной элемент α_{cm}.
5. Если $\alpha_{cm} = 0$, то перейти к п. 8, иначе к п. 6.
6. Если $\alpha_{cm} = \alpha_{ck}$ для одного из элементов формируемого столбца β_k, то перейти к п. 4, иначе к п. 7.
7. Принять $\beta_k = 10\ \beta_k + \alpha_{cm}$ и перейти к п. 2.

8. Восстановить m-ю строку.
9. Принять $m = m - 1$.
10. Если $m \neq 0$, то перейти к п. 4, иначе к п. 11.
11. Принять $\beta_k = 0$.
12. Вывести β_k.
13. Если $\beta_k = 0$, то прекратить вычисления, иначе перейти к п. 8.

Отделение очередного элемента α_{cm} в п. 4 этого алгоритма можно реализовать с помощью запятой, перемещаемой умножением или делением строки на 10, и операторов косвенной адресации, отбрасывающих дробную часть числа с ненулевой целой частью. При программировании на входном языке ЯМК34 для выделения α_{cm} в строке, записанной в виде числа с нулевой целой частью, достаточно умножить строку на 10 (для выделения ее первого элемента), занести результат в регистр с номером $N \geqslant 7$ и выполнить оператор КИПN, отбрасывающий дробную часть содержимого регистра N. Подобный способ неприменим для входных языков, не содержащих операторы косвенной адресации, а выделение α_{cm} другими способами приводит к значительному удлинению программы.

Определенные затруднения связаны с организацией сравнения элемента α_{cm} со всеми элементами формируемого столбца в п. 6 алгоритма. В простейшем случае такое сравнение реализуемо двумя шагами:

1. Выделить из формируемого столбца очередной элемент α_{ck}. Если $\alpha_{ck} = 0$, то перейти к п. 7 алгоритма, иначе выполнить следующий шаг.

2. Если $\alpha_{ck} = \alpha_{cm}$, то перейти к п. 4, иначе повторить первый шаг.

Подобная реализация связана со значительными затратами времени, так как для матрицы порядка n требуется повторять эти два шага от одного до $n - 1$ раз для всех элементов очередных строк матричного числа. Можно ускорить выполнение п. 6 алгоритма, выделив n регистров для хранения элементов формируемого столбца или, для уменьшения требований к емкости памяти, только информацию о наличии в столбце элемента $\alpha_k \in 1, 2, \ldots, n$. В последнем случае достаточно, например, изменить лишь знак содержимого регистра и, следовательно, для хранения подобной информации можно использовать n регистров, в которых хранятся элементы строк матричного числа, отображаемые положительными многозначными числами.

Простейший способ восстановления m-й строки в п. 8 алгоритма заключается в выделении $2n$ регистров памяти для хранения n исходных и n преобразуемых отделением элементов α_{cm} строк матричного числа. Так как дополнительно требуется не менее двух адресных регистров для косвенной адресации, то на ПМК с входным языком ЯМК34 при 14 регистрах числовой памяти в этом случае можно автоматизировать построение контурных чисел для матриц порядка $n \leqslant 6$.

Для полного использования разрядности мантиссы и, соответственно, предельного порядка исходной матрицы необходимы другие способы восстановления строк. Один из них заключается в отделении элемента α_{cm} следующей за ним запятой, перемещаемой слева направо. В этом случае исходная строка будет отличаться от преобразуемой

только запятой и восстановление строки, хранимой в регистре N, заключается в ее нормализации, например, с помощью фрагмента

ИПN 1 0 ÷ ПN 1 — $x < 0$ A ... ,

где A — адрес начального оператора фрагмента. Подобный способ реализован в следующей программе.

Программа 53/34. Формирование контурных чисел для матриц порядка $n \leqslant 8$

П9	1	0	ПС	Сх	П0	1	+	ПА	КИПА
ПВ	$x<0$	14	/—/	1	+	ПД	КИПД	XY	ИПД
—	$x\neq 0$	83	ИПС	×	ПД	КИПД	$x\geqslant 0$	78	ИП0
ИПС	×	ИПД	+	ИП9	ИПА	—	$x=0$	68	+
С/П	1	ИП0	ИПС	÷	+	П0	КИП0	Вх	ИП0
—	ИПС	×	ПД	КИПД	/—/	КПД	ИПА	1	—
ПА	$x\neq 0$	40	КИПА	ИПС	×	БП	10	XY	П0
ИПВ	КПА	КИПД	/—/	КПД	ИПА	БП	06	ИПВ	ИПС
×	БП	10	КИПА	x^2	1	—	$x\geqslant 0$	41	КИПА
ИПС	÷	КПА	БП	84					

Инструкция: нормализованные (с запятой перед первой цифрой) множества элементов строк матричного числа занести в порядке номеров строк в регистры *1*, *2*, ..., *n*, выполнить n = PX В/О С/П PX = $= \beta_1$ С/П $PX = \beta_2$ ... С/П $PX = 0$.

Для проверки программы воспользуемся в качестве исходной матрицей коэффициентов, определитель которой был найден вручную по приведенному алгоритму метода обобщенных чисел. Введя исходные данные 0,13 = *P1*; 0,126 = *P2*; 0,234 = *P3*; 0,134 = *P4*; 0,1345 = = *P5*; 0,1346 = *P6*; 6 = *PX*, получим β_1= 123456 (время счета около 2 мин 30 с), β_2= 124356 (2 мин 40 с), β_3= 162354 (4 мин 10 с), β_4= 162453 (2 мин 35 с), β_5= 312456 (9 мин 50 с), β_6 = 324156 (4 мин 30 с), β_7= 362154 (5 мин 10 с), β_8 = 362451 (2 мин 35 с), β_9= 0 (4 мин 5 с). Таким образом, общие затраты времени на формирование столбцов контурного числа составили около 38 мин.

Затраты времени в случае $n \leqslant 6$ уменьшаются при отказе от восстановления строк нормализацией и использовании для преобразуемых строк дополнительных регистров числовой памяти.

Программа 54/34. Формирование контурных чисел для матриц порядка $n \leqslant 6$

Сх	П0	7	БП	18	КИПС	/—/	КПС	ИП0	1
0	×	ИПС	+	П0	ИПД	1	+	ПД	6
—	ПС	КИПС	$x<0$	26	/—/	ПС	КПД	КИПС	$x\geqslant 0$
45	ИПД	1	2	—	$x=0$	05	ИП0	ВП	1
ИПС	+	С/П	БП	53	XY	ИПС	—	1	0
×	$x=0$	26	ИПД	1	—	ПД	КИПД	ПС	КИПС
/—/	КПС	XY	ИП0	ИПС	—	1	0	÷	П0
$x<0$	45	Сх	С/П						

Инструкция: при $n < 6$ вместо операторов 1 2 по адресам 32 и 33 записать в программу операторы набора числа $6 + n$ (например, при $n = 3$ записать операторы 0 9); множества элементов m-х строк с запятой после первой цифры ввести в регистры с номерами m; В/О С/П $PX = \beta_1$ С/П $PX = \beta_2$ ... С/П $PX = 0$.

При выполнении этой программы на том же микрокалькуляторе после ввода $1{,}3 = P1$; $1{,}26 = P2$; $2{,}34 = P3$; $1{,}34 = P4$; $1{,}345 = P5$; $1{,}346 = P6$ получим искомый результат при общих затратах времени около 23 мин.

Затраты времени при $n = 6$ можно дополнительно уменьшить, сократив число операторов косвенной адресации. Длина программы в этом случае увеличится, но, так как затраты времени на ее ввод в программную память значительно меньше времени выполнения программы при $n > 3$, общие затраты времени на решение задачи уменьшатся.

Программа 55/34. Ускоренное формирование контурных чисел для матриц порядка $n = 6$

ИП1	ПП	60	$x \neq 0$	85	П7	ИП2	ПП	60	$x \neq 0$
55	П8	ИП3	ПП	60	$x \neq 0$	50	П9	ИП4	ПП
60	$x \neq 0$	45	ПА	ИП5	ПП	60	$x \neq 0$	40	ПВ
ИП6	ПП	60	$x = 0$	85	ИПВ	ПП	79	БП	27
ИПА	ПП	79	БП	21	ИП9	ПП	79	БП	15
ИП8	ПП	79	БП	09	ИП7	ПП	79	БП	03
$x < 0$	63	/—/	ПД	КИПД	$x < 0$	75	XY	ИПД	—
ИП0	×	$x = 0$	63	В/О	/—/	КПД	XY	В/О	ПД
КИПД	/—/	КПД	БП	67	ПС	С/П	КИПС	/—/	КПС
БП	35								

Инструкция: множества элементов (с запятой после первой цифры) m-х строк занести в регистры с номерами m; (10 = *P0*) (В/О) С/П множества элементов очередного столбца хранятся в виде целых частей содержимого регистров *7*, *8*, ..., *C* ($PX = PC$); $PX = 0$.

При выполнении рассмотренного примера на том же калькуляторе по этой программе время формирования контурного числа сокращается до 16 мин.

Для сокращения времени счета в программах 50/34—52/34 не предусмотрено определение четности подстановок членов определителя, от которой зависит знак этого члена. Ее несложно определить по числу инверсий (бóльших индексов над меньшими) в столбцах контурного числа непосредственно или автоматизировать ее определение с помощью следующей программы:

П1	1	ПД	ПС	ИПС	С/П	ПВ	ИПД	1	+
ПД	П0	ИПВ	КПД	ИПВ	КИП0	—	$x < 0$	22	ИПС
/—/	ПС	ИП0	1	—	$x = 0$	14	БП	04	

По этой программе можно определять четность ($PX = 1$) или нечетность ($PX = -1$) в последовательностях до 12 порядковых чисел,

вводимых в регистр X с нажатием после каждого ввода клавиши (В/О) С/П.

Применение программируемых микрокалькуляторов обеспечивает автоматизацию решения в общем виде и многих других частных задач. К ним относится достаточно громоздкая задача составления в общем (буквенном) виде произведения матриц $C = A \times B$ с элементами

$$c_{mq} = \sum_{k=1}^{n} a_{mk} b_{kq}.$$

Составляющие элементов матрицы-произведения, как следует из этой формулы, расположенные на пересечении m-й строки и q-го столбца, образованы произведениями ненулевых членов a_{ml} и b_{pq} матриц-множителей с совпадающими индексами $l = q$. Перебор таких сочетаний и, следовательно, соответствующих им произведений элементов несложно выполнить вручную [10] или автоматизировать с помощью следующей программы.

Программа 56/34. Символьное умножение матриц $A \times B = C$

ПС	1	2	П0	ИП0	1	—	$x \neq 0$	34	КИП0
$x \neq 0$	09	↑	ИПС	ВП	2	+	ПД	→	ПП
37	1	0	÷	ИПС	ПП	37	ИПС	XY	—
—	$x = 0$	04	ИПД	С/П	БП	04	1	ВП	8
+	Вх	—	В/О						

Инструкция: ввести индекс pq до 12 ненулевых элементов матрицы B в регистры $1, \ldots, B$; ввести в регистр X индексы ml ненулевых элементов матрицы A, нажимая после каждого ввода клавишу С/П (первый раз — В/О С/П) и регистрируя высвечиваемые сочетания $mllq$ до индикации нуля; если число элементов матрицы B, отличающихся от нуля, более 12, то повторить выполнение программы для остальных элементов; по вычисленным сочетаниям индексов $mllq$ составить матрицу с произведениями $a_{ml}b_{lq}$, суммируемыми на пересечении m-й строки и q-го столбца.

Так, для определения с помощью этой программы произведения матриц

$$A = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & 0 & b_{13} \\ 0 & b_{22} & b_{23} \\ 0 & 0 & b_{33} \\ b_{41} & 0 & b_{43} \end{bmatrix}$$

следует ввести $11 = P1$, $13 = P2$, $22 = P3$, $23 = P4$, $33 = P5$, $41 = P6$ и, вводя в регистр X последовательно индексы ненулевых элементов первой матрицы, получить $mllq = 1111$, 1113, 2111, 2113, 2222, 2223, 2333, 3333, 3441, 3443 или

$$C = \begin{bmatrix} a_{11}b_{11} & 0 & a_{11}b_{13} \\ a_{21}b_{11} & a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{34}b_{41} & 0 & a_{33}b_{33} + a_{34}b_{43} \end{bmatrix}.$$

Соответствующим образом можно автоматизировать и решение в общем виде многих других частных задач.

Глава 6

ИНТЕГРИРОВАНИЕ И ВЫЧИСЛЕНИЕ СПЕЦИАЛЬНЫХ ФУНКЦИЙ

1. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

Обыкновенный определенный интеграл I в интервале интегрирования $[a, b]$ при известной первообразной $F(x)$ для подынтегральной функции $f(x) = F'(x)$ вычисляют по формуле Ньютона—Лейбница $I = F(b) - F(a)$. Если первообразную найти трудно или невозможно, то прибегают к методам численного интегрирования, основанным на аппроксимации (интерполировании) подынтегральной функции в интервале интегрирования и приближенном представлении интеграла квадратурной формулой

$$I = \int_a^b f(x)\,dx \approx \sum_{i=0}^{n} A_i f(x_i),$$

где весовые коэффициенты A_i и значения аргумента x_i в узлах определяются выбранным методом приближения. При разбиении интервала интегрирования на несколько частей, в каждой из которых подынтегральную функцию аппроксимируют (интерполируют) определенным способом, квадратурную формулу называют составной.

Различают замкнутые и открытые квадратурные формулы. При использовании замкнутых формул интервал интегрирования разбивают на n обычно равных (например, для квадратурных формул Ньютона—Котеса [4, 6]) элементарных отрезков, границы которых являются узлами интерполяции подынтегральной функции в каждой части интервала интегрирования, содержащей m элементарных отрезков. При увеличении чисел n и m методическая погрешность уменьшается, но увеличиваются операционные погрешности и время вычислений. Поэтому при программировании универсальных ЭВМ для вычисления определенного интеграла по заранее неизвестной подынтегральной функции обычно используют квадратурные формулы с интерполирующими многочленами второй или, реже, третьей степени, например составную формулу Симпсона (формулу парабол)

$$I \approx h\left(\left(f(a) + f(b) + 2\sum_{i=1}^{n/2-1} f(a+2ih) + 4\sum_{i=1}^{n/2} f(a-h+2ih)\right)/3, \quad (6.1)\right.$$

где n — четное число разбиений интервала интегрирования на элементарные отрезки; $h = (b-a)/n$ — шаг интегрирования (длина элементарного отрезка).

При вычислениях по этой формуле в каждой части интервала, содержащей два элементарных отрезка, подынтегральная функция интерполируется квадратичным многочленом. Формула достаточно удобна для вычисления определенных интегралов с помощью програм-

мируемых микрокалькуляторов (см. программы 132/34 и 174/21 приложения).

Для оценки точности вычисление интеграла по формуле (6.1) приходится повторять для пар близких значений n и принимать верными совпадающие цифры результатов. При заданной точности результата такие оценки приходится повторять при увеличении n, однако общие затраты времени оказываются меньшими, чем при вычислении пары приближений интеграла для заведомо больших значений n, когда существенно возрастают время счета и влияние операционных погрешностей.

Например, для интеграла вероятности

$$I = \sqrt{\frac{2}{\pi}} \int_0^4 e^{-x^2/2}\, dx \qquad (6.2)$$

по программе 132/34 при $n = 4$ получим $I = 0{,}9951122$ (время счета около 50 с), при $n = 8$ и $n = 10$ — соответственно $I = 0{,}9999329$ (время счета около 95 с) и $I = 0{,}99993496$ (время счета около 115 с). Следовательно, $I = 0{,}99993$ с предельной абсолютной погрешностью $\varepsilon \leqslant 1 \cdot 10^{-5}$.

Более высокую точность интегрирования при близких затратах времени (но при более сложной программной реализации) обеспечивает использование открытых квадратурных формул, в которых узлы интегрирования x_i не совпадают с границами интервалов аппроксимации подынтегральной функции и определяются, как и весовые коэффициенты A_i, с помощью ортогональных многочленов [4, 6]. Примером может служить квадратурная формула Гаусса второго порядка

$$I \approx (b - a)(8f(x_0) + 5f(x_1) + 5f(x_2))/18,$$

где $x_i = ((b + a + (b - a)t_i)/2$, $t_0 = 0$, $t_{1,2} = \pm 0{,}77459667$.

При вычислении интеграла по составной формуле подобного типа аппроксимация подынтегральной функции повторяется для каждого из k отрезков интервала интегрирования.

Программа 57/34. Вычисление определенного интеграла I в интервале $[a, b]$ по составной формуле Гаусса второго порядка

ИПВ	ИПА	—	XY	÷	ПД	Cx	П8	ПП	28
÷	8	×	П8	ПП	25	ПП	25	×	ИПД
×	1	8	÷	С/П	ИПС	/—/	ПС	1	+
ИПД	2	÷	×	ИПА	+	П9	(ИП9)	...	ИП8
+	П8	ИП9	ИПД	+	П9	ИПВ	—	$x \geqslant 0$	37
ИП8	5	В/О							

Инструкция: заменить многоточие в программе операторами вычисления подынтегральной функции при $x = P9$ (если вычисления не начинаются вызовом аргумента, то взятый в скобки оператор ИП9 не нужен), ($a = PA$, $b = PB$; $0{,}77459667 = PC$) $k = PX$ В/О С/П $PX = I$.

Для интеграла (6.2) по этой программе при $k = 2$ получим $I = 0{,}99961561$ (время счета около 60 с), при $k = 3$ и $k = 4$ соответственно $I = 0{,}99993622$ (время счета около 80 с) и $I = 0{,}99993672$ (время счета около 120 с). Следовательно, при близких затратах времени погрешность результата интегрирования по программе 57/34 на порядок меньше, чем по программе 132/34.

Вычисление интеграла с заданной точностью можно полностью автоматизировать, учитывая, что методическая погрешность по формуле Симпсона (6.1) уменьшается примерно в 15 раз при уменьшении шага интегрирования в 2 раза. Алгоритм вычисления интеграла с заданной точностью реализован в следующей базовой программе, при выполнении которой с шагом h_i вычисляется и заносится в регистр 7 сумма

$$S_1 = f(a) + f(b) + 2\sum_{k=1}^{n-1} f(a + kh_i),$$

после чего с шагом $h_{i+1} = h_i/2$ вычисляется сумма

$$S_2 = \sum_{k=0}^{n-1} f(a + h_{i+1} + kh_i)$$

и значение интеграла $I_{i+1} = h_{i+1}(S_1 + 4S_2)/3$. Если условие $(15\varepsilon)^2 < (I_{i+1} - I_i)^2$ не выполняется, то принимается $S_1 = S_1 + 2S_2$ и цикл вычислений повторяется.

Программа 58/34. Вычисление определенного интеграла I по составной формуле Симпсона с заданной предельной погрешностью

Сх	П8	ПД	ИПВ	ИПА	—	П6	ИПА	ПП	47
ИП8	П7	Сх	П8	ИПА	ИП6	2	÷	+	ПП
47	ИПС	ИП7	ИП8	2	×	+	П7	Вх	+
ИП6	2	÷	П6	×	3	÷	ИПД	XY	ПД
—	x^2	—	$x \geqslant 0$	12	ИПД	С/П	П9	(ИП9)	...
ИП8	+	П8	ИПВ	ИП9	ИП6	+	П9	—	$x < 0$
48	В/О								

Инструкция: заменить в программе многоточие операторами вычисления подынтегральной функции при $x = P9$ (если вычисления не начинаются вызовом аргумента, то взятый в скобки оператор ИП9 не нужен), ($a = PA$, $b = PB$, $(15\varepsilon)^2 = PC$) В/О С/П $PX = I$, $P6 = h_{i+1}$.

По этой программе для интеграла (6.2) при $\varepsilon = 1 \cdot 10^{-5}$ получим $I = 0{,}99993633$ (время счета около 3 мин) при точном значении $I = 0{,}99993667$.

Для сокращения времени интегрирования достаточно гладких подынтегральных функций выбирают простые интерполирующие выражения. В подобных случаях часто прибегают к кусочно-линейной интерполяции на элементарных отрезках в соответствии с составной формулой трапеций

$$I \approx \sum_{i=0}^{n-1} h(f(a + ih) + f(a + ih + h))/2. \qquad (6.3)$$

Меньшей методической погрешностью (при меньшем на единицу числе узлов интерполяции) отличается интерполирование функции $f(x)$ на элементарном отрезке постоянным числом, равным значению функции на средине отрезка. Такая прямоугольная (кусочно-постоянная) интерполяция соответствует открытой квадратурной формуле Гаусса нулевого порядка

$$I \approx \sum_{i=0}^{n-1} hf(a + (0{,}5 + i)h),$$

реализованной в следующей программе.

Программа 59/34. Вычисление определенного интеграла I по составной формуле Гаусса нулевого порядка

ПС	ИПВ	ИПА	—	ИПС	÷	ПС	Cx	ПД	ИПА
ИПС	2	÷	+	П9	(ИП9)	...	ИПД	+	ПД
ИП9	ИПС	+	П9	ИПВ	—	$x \geqslant 0$	15	ИПД	ИПС
×	С/П	БП	00						

Инструкция: заменить в программе многоточие операторами вычисления подынтегральной функции при $x = P9$ (если вычисления не начинаются вызовом аргумента, то взятый в скобки оператор ИП9 не нужен), ($a = PA$, $b = PB$) $n = PX$ В/О С/П $PX = I$.

Подобные программы целесообразно использовать, когда требуется минимальное число операторов при относительно невысокой точности результата, например, если численное интегрирование является частью вычислений, реализуемых в сложной программе. В последнем случае можно изъять первые семь операторов программы 59/34, изменив адрес перехода и задавая h в исходных данных (подобным приемом можно несколько сократить и ранее рассмотренные программы).

По программе 59/34 для интеграла (6.2) при $n = 4$ получим $I = 0{,}99996779$ (время счета около 45 с), при $n = 10$ и $n = 11$ соответственно получим $I = 0{,}99994332$ (время счета около 90 с) и $I = 0{,}9999422$ (время счета около 100 с).

Внимательного подхода требует интегрирование функций с особыми точками. Если в интервале интегрирования подынтегральная функция или ее производная обращаются в бесконечность, то непосредственное численное интегрирование невозможно или сопровождается большими погрешностями. В таких случаях следует попытаться аналитическими преобразованиями устранить особые точки в интервале интегрирования. Например, интеграл

$$p(z) = (1/\sqrt{2\pi^3}\sigma) \int_{-A}^{A} (e^{-(x-z)^2/2\sigma^2}/\sqrt{A^2 - x^2})\, dx$$

обращается в бесконечность на границах интервала интегрирования, но при замене $x = A \cos y$ интеграл приводится к виду

$$p(z) = (1/\sqrt{2\pi^3}\sigma) \int_{0}^{\pi} e^{-(A \cos y - z)^2/2\sigma^2}\, dy,$$

где подынтегральная функция конечна во всем интервале интегрирования.

Вычисления упрощаются и ускоряются при нормировке переменных. Обозначив в рассматриваемом интеграле $B = A/\sigma$, $t = z/\sigma$, получим

$$p(t) = (1/\sqrt{2\pi^3}) \int_0^{\pi} e^{-(B\cos y - t)^2/2}\, dy$$

и вычисления соответственно упростятся.

Если подынтегральная функция в интервале интегрирования имеет особые точки с неопределенностью вида 0/0, то следует либо исключить эту точку, разбив на две части интервал интегрирования, либо раскрыть неопределенность в программе. Например, при вычислении интегрального синуса

$$\mathrm{Si}(x) = \int_0^x (\sin t/t)\, dt \qquad (6.4)$$

на нижней границе интервала достаточно принять значение подынтегральной функции равным единице. Если один из пределов интегрирования фиксирован (например, равен нулю), то базовую программу целесообразно соответственно упростить.

Интегрируемая функция часто содержит постоянный множитель и в этом случае для ускорения вычислений целесообразно организовать в программе умножение на этот множитель после выполнения интегрирования, а влияние множителя на абсолютную погрешность результата учесть при выборе допустимого значения ε.

Вычисление несобственных интегралов вида

$$I = \int_a^{\infty} f(x)\, dx \qquad (6.5)$$

заменой $y = 1/x$ можно свести к вычислению собственного интеграла

$$I = \int_0^{1/a} (f(y)/y^2)\, dy,$$

но такое преобразование бесполезно, если нижний предел интегрирования равен нулю или на нижней границе интервала преобразованная подынтегральная функция стремится к бесконечности.

Первое из этих затруднений иногда удается устранить, разбивая интервал интегрирования на две части:

$$I = \int_0^{\infty} f(x)\, dx = \int_0^a f(x)\, dx + \int_a^{\infty} f(x)\, dx = \int_0^a f(x)\, dx + \int_0^{1/a} (f(y)/y^2)\, dy.$$

Например,

$$I = \int_0^{\infty} \frac{x^2\, dx}{1+x^4} = \int_0^a \frac{x^2\, dx}{1+x^4} + \int_a^{\infty} \frac{x^2\, dx}{1+x^4},$$

что при $a = 1$ обеспечивает вычисление

$$I = \int_0^1 \frac{x^2\,dx}{1+x^4} + \int_0^1 \frac{dx^2}{1+x^4} = \int_0^1 \frac{(1+x^2)\,dx}{1+x^4}.$$

Если несобственный интеграл (6.5) не удается свести к собственному, то его интегрирование можно выполнить с помощью базовой программы 55/34, дополнив ее фрагментом

ИПВ ПА 2 × ПВ ИПД ИП5 + П5 Вх
— $x = 0$ 00 ИП5 ...,

которым заменяется оператор ИПД перед оператором С/П с заменой адресов 47 и 48 обращений к подпрограмме соответственно адресами 60 и 61. В этом случае в регистре *5* накапливается сумма парциальных интегралов, вычисляемых в интервалах $[a, b_0]$, $[b_0, 2b_0]$... с удвоением верхнего предела интегрирования до тех пор, пока величина очередного парциального интеграла влияет на результат вычислений (критерий максимальной точности).

Инструкцию к такой модифицированной программе следует дополнить требованием очистки регистра *5* перед каждым выполнением программы (эту операцию несложно автоматизировать) и вводом выбранного начального верхнего предела b_0 в регистр *B*. Так, для вычисления интеграла

$$I = \int_0^\infty x e^{-x}\,dx$$

следует заменить в модифицированной программе многоточие операторами ИП9 ↑ e^x ÷ и, приняв, например, $\varepsilon = 5 \cdot 10^{-5}$ и $b_0 = 2$, получить примерно через 8 мин $I = 0{,}99996415$ (при точном значении $I = 1$).

В модифицированной программе контролируется абсолютная погрешность парциальных интегралов. Полную погрешность, которая может превысить заданную, можно оценить по начальному значению b_0 и значению b_{max} после выполнения программы. Например, после вычисления приведенного выше интеграла в регистре *B* хранится $b_{max} = 128$ и, следовательно, исходное значение b_0 верхнего предела удваивалось семикратно, было вычислено шесть парциальных интегралов и предельная полная абсолютная погрешность результата $\varepsilon_\Sigma = 6 \cdot \varepsilon = 3 \cdot 10^{-4}$.

Если точность результата вычислений необходимо гарантировать, то при интегрировании функций, убывающих в среднем быстрее $1/x$ при $x \to \infty$, на первый парциальный интеграл следует выделить половину допустимой погрешности, а на каждый следующий в два раза меньше. В этом случае суммарная погрешность не превысит заданную, а для программной реализации этого способа оценки в базовой про-

грамме 58/34 оператор ИПД перед оператором С/П следует заменить фрагментом

ИПВ ПА 2 × ПВ ИПС 4 ÷ ПС ИПД
ИП5 + П5 Вх — $x=0$ 00 ИП5

с изменением адресов обращений к подпрограмме соответственно на 63 и 64. В этом случае b_0 следует выбирать так, чтобы в интервале $[a, b_0]$ заключалось основное значение интеграла, так как в противном случае вычисления окажутся длительными или не окончатся.

Значения многих несобственных интегралов вида (6.5) известны, что можно использовать для вычисления таких интегралов в конечных пределах, если верхний предел достаточно велик. Например, зная, что $\Phi(\infty)=1$, вместо

$$\Phi(4)=\sqrt{2/\pi}\int_0^4 e^{-x^2/2}\,dx$$

можно записать

$$\Phi(4)=1-\sqrt{2/\pi}\int_4^\infty e^{-x^2/2}\,dx$$

и с помощью модифицированной программы найти значение $\Phi(4)$ быстрее или (в зависимости от выбранного шага интегрирования) точнее, чем при непосредственном интегрировании.

Аналогично при вычислении интегрального синуса (6.4) в пределах $[0, a]$ следует учесть $\mathrm{Si}(\infty)=\pi/2$ и при $a\gg 1$ принять

$$\mathrm{Si}(a)=\pi/2-\int_a^\infty (\sin x/x)\,dx.$$

Для ускорения сходимости вычисления полученного интеграла целесообразно воспользоваться интегрированием по частям:

$$\int_a^\infty \frac{\sin x}{x}\,dx=-\frac{\cos x}{x}\Big|_a^\infty-\int_a^\infty \frac{\cos x}{x^2}\,dx=\frac{\cos a}{a}-\int_a^\infty \frac{\cos x}{x^2}\,dx=$$

$$=\frac{\cos a}{a}+\frac{\sin a}{a}-2\int_a^\infty \frac{\sin x}{x^3}\,dx.$$

При повышении порядка $m>2$ методическая погрешность составных формул Ньютона — Котеса уменьшается мало (иногда даже возрастает), так как зависит от максимального значения производной, порядок которой возрастает при увеличении порядка расчетной формулы. В таких случаях для ускорения вычисления интеграла с заданной точностью целесообразно использовать программу 57/34 или другие программы, реализующие метод Гаусса, в котором подынтегральная функция аппроксимируется степенным рядом Тейлора.

При табличном задании подынтегральной функции с шагом h и не очень высоких требованиях к точности результата вычислений удобно

использовать составную формулу трапеций (6.3), реализовав ее, например, следующей программой.

Программа 60/34. Численное интегрирование по формуле трапеций табулированных функций с постоянным шагом h аргумента

ПА Сх ПС С/П ИПА XY ПА + 2 ÷
ИПД × ИПС + БП 02

Инструкция: $h = РД$, $f_0 = РХ$ В/О С/П $РХ = 0$, $f_1 = РХ$ С/П $РХ = I_1$, $f_2 = РХ$ С/П $РХ = I_2$ ... $f_n = РХ$ С/П $РХ = I_n = I$.

Например, для табличной модели подынтегральной функции в выражении (6.2) в виде последовательности $f(0) = 0{,}79789$; $f(1) = 0{,}48394$; $f(2) = 0{,}10798$; $f(3) = 0{,}0088637$; $f(4) = 0{,}0002677$ с шагом $h = 1$ по этой программе получим $I_1 = 0{,}640915$; $I_2 = 0{,}936875$; $I_3 = 0{,}99529685$; $I_4 = I = 0{,}99986255$.

При неравноотстоящих узлах табличной модели также целесообразно использовать формулу трапеций.

Программа 61/34. Численное интегрирование по формуле трапеций табулированных функций с неравноотстоящими узлами аргумента

П8 XY П7 Сх П9 С/П ИП8 XY П8 +
XY ИП7 XY П7 XY — × 2 ÷ ИП9
+ БП 04

Инструкция: $x_0 = РУ$, $f_0 = РХ$ В/О С/П $РХ = 0$, $x_1 = РУ$, $f_1 = РХ$ С/П $РХ = I_1$, $x_2 = РУ$, $f_2 = РХ$ С/П $РХ = I_2$ ... $x_n = РУ$, $f_n = РХ$ С/П $РХ = I_n = I$. Для проверки можно воспользоваться данными предыдущего контрольного примера.

Точность интегрирования табулированных функций обычно повышается при интерполировании многочленами второй или более высокой степени. В следующей программе после ввода отсчета табличной модели f_i, $i = 0, 1, 2, \ldots, n$ с четным номером $i \geqslant 2$ интегрирование реализовано по формуле Симпсона

$$I \approx h(f_i + 4f_{i-1} + f_{i-2})/3,$$

а после ввода отсчета с нечетным номером $i \geqslant 3$ интегрирование выполняется по формуле Ньютона

$$I \approx 3h(f_i + 3f_{i-1} + 3f_{i-2} + f_{i-3})/8$$

с суммированием предыдущего результата вычислений по формуле Симпсона.

Программа 62/34. Численное интегрирование табулированных функций с постоянным шагом аргумента по формулам Симпсона и Ньютона

П1 Сх ПД С/П П2 Сх ПС С/П П3 ИПС
ИПД + ПД ИП1 ИП2 4 × + ИП3 +
ИП0 × 3 ÷ ПС + С/П ИП1 ИП2 ИП3
П1 + 3 × + XY П2 + ИП0 ×
3 × 8 ÷ ИПД + БП 07

Инструкция: ($h = PO$), $f_0 = PX$ В/О С/П $PX = 0$, $f_1 = PX$ С/П $PX = 0$, $f_2 = PX$ С/П $PX = I_2$ ... $f_n = PX$ С/П $PX = I_n = I$.

Результат вычислений по этой программе может оказаться менее точным, чем по предыдущей, так как закон изменения табулированной функции между узлами не определен. Например, для табличной модели из контрольного примера к программе 60/34 по программе 62/34 получим $I = 0{,}99511083$, что соответствует точности вычислений по формуле Симпсона при $n = 4$, но случайно уступает точности результата вычислений по программе 60/34.

Приведенные алгоритмы и программы применимы для вычисления криволинейных интегралов

$$\int_{AB} P(x_1, x_2)\,dx_1 + Q(x_1, x_2)\,dx_2$$

после подстановки $x_1 = \varphi(t)$, $x_2 = \psi(t)$ и перехода к обыкновенному интегралу

$$\int_a^b (P(\varphi(t), \psi(t))\varphi'(t) + Q(\varphi(t), \psi(t))\psi'(t))\,dt.$$

Кратные определенные интегралы вычисляют, разбивая прямоугольной сеткой область аргумента на элементарные квадраты для двойных интегралов или элементарные кубы для тройных интегралов. Если границы области интегрирования не параллельны координатным осям в пространстве переменных, то погрешность моделирования уменьшают выбором меньшего шага интегрирования, принимаемого равным половине стороны элементарного квадрата или куба.

Элементарный двойной интеграл вычисляют [6] по формуле

$$\int_{-h}^{h}\int_{-h}^{h} f(x_1, x_2)\,dx_1\,dx_2 \approx 2h^2 (f_{-1,0} + f_{0,1} + 2f_{0,0} + f_{0,-1} + f_{1,0})/3,$$

где h — шаг интегрирования; $f_{0,0}$ — значение подынтегральной функции в центре элементарного квадрата; $f_{i,j}$ — значения функции на средине сторон квадрата.

Так как область интегрирования замещается несколькими рядами элементарных квадратов, причем в общем случае каждый ряд может содержать различное их число, то при программной реализации вычисления двойного интеграла целесообразно автоматизировать при каждом выполнении программы вычисление и суммирование элементарных интегралов одного ряда. В этом случае перед каждым пуском программы удобно задавать координаты переменных, соответствующих средине левой стороны крайнего левого элементарного квадрата очередного ряда и, при необходимости, граничное значение $x_{1гр}$ для правой стороны крайнего правого квадрата ряда. После суммирования интегралов, вычисленных для всех рядов, получим приближенное значение двойного интеграла.

Программа 63/34. Приближенное вычисление двойного интеграла

Cx	ПС	ПП	56	ИПВ	ИП1	ИПА	+	П1	—
x < 0	14	ИП0	С/П	ПП	56	Вх	ИПС	+	ПС
ПП	52	ИП2	ИПА	2	×	—	ПП	55	ИП1
ИПА	+	П1	ПП	52	Вх	ПД	ИПС	ИПА	x^2
×	2	×	3	÷	ИП0	+	П0	ИПД	ПС
БП	04	ИП2	ИПА	+	П2	...	ИПС	XY	+
ПС	В/О								

Инструкция: заменить многоточие в программе операторами вычисления (при $x_1 = P1$, $x_2 = P2$) подынтегральной функции, $0 = P0$, $x_{10}^{(1)} = P1$, $x_{20}^{(1)} = P2$, $x_{1гр}^{(1)} = PB$, $h = PA$ В/О С/П $PX = I_1$, $x_{10}^{(2)} = P1$, $x_{20}^{(2)} = P2$, $x_{1гр} = PB$ В/О С/П $PX = I_2$ ... $x_{10}^{(n)} = P1$, $x_{20}^{n} = P2$, $x_{1гр}^{(n)} = PB$ В/О С/П $PX = I_n = I$ (в скобках указан номер ряда элементарных квадратов).

В качестве контрольного примера вычислим интеграл

$$I = \int_0^1 \int_2^3 \frac{dx_1\, dx_2}{\ln(x_1 + x_2)}.$$

Представив область интегрирования элементарным квадратом со стороной $2h = 1$, при $x_{10} = 0$, $x_{20} = 2{,}5$; $x_{1гр} = 1$, получим $I = 0{,}9332772$ (время счета около 45 с). Разбив область интегрирования на четыре квадрата со сторонами $2h = 0{,}5$, при $x_{10}^{(1)} = 0$, $x_{20}^{(1)} = 2{,}25$; $x_{1гр}^{(1)} = I$ получим $I_1 = 5{,}0501972$ (время счета около 90 с) и при $x_{10}^{(2)} = 0$, $x_{20}^{(2)} = 2{,}75$ получим $I_2 = I = 0{,}93422185$.

Элементарный тройной интеграл для куба со стороной $2h$ вычисляют [6] по формуле

$$\int_{-h}^{h}\int_{-h}^{h}\int_{-h}^{h} f(x_1, x_2, x_3)\, dx_1, dx_2, dx_3 \approx 4h^3 (f_{-1,0,0} + f_{0,1,0} + f_{0,0,1} + f_{0,-1,0} + f_{0,0,-1} + f_{1,0,0})/3,$$

где $f_{i,j,k}$ — значения подынтегральной функции в центрах граней элементарного куба.

При вычислениях по этой формуле трехмерная область интегрирования разбивается прямоугольной сеткой с шагом $2h$ на параллельные ряды кубов. Программу целесообразно составлять для суммирования кубов отдельных рядов с заданием координаты центра левой грани левого куба ряда и центра правой грани правого куба ряда. После каждого выполнения программы объем кубов ряда суммируется с ранее вычисленной суммой объемов рядов и после суммирования объемов всех кубов получают приближенное значение тройного интеграла.

Программа 64/34. Приближенное вычисление тройного интеграла

Cx	ПС	ПП	73	ИПВ	ИП1	ИПА	+	П1	—
$x<0$	14	ИП0	С/П	ПП	69	ИП3	ИПА	+	П3
ИП2	ИПА	—	ПП	72	ИП3	ИПА	—	П3	ИП2
ИПА	—	ПП	72	ИП3	ИПА	—	П3	ПП	69
ИП3	ИПА	+	П3	ИП1	ИПА	+	П1	ПП	73
Bx	ПД	ИПС	ИПА	↑	x^2	×	×	4	×
3	÷	ИП0	+	П0	ИПД	ПС	БП	04	ИП2
ИПА	+	П2	...	ИПС	XY	+	ПС	В/О	

Инструкция: заменить многоточие операторами вычисления (при $x_i = Pi$) подынтегральной функции, $0 = P0$, $h = PA$, $x_{10}^{(1)} = P1$, $x_{20}^{(1)} = P2$, $x_{30}^{(1)} = P3$, $x_{1гр}^{(1)} = PB$ В/О С/П $PX = I_1$, $x_{10}^{(2)} = P1$, $x_{20}^{(2)} = P2$, $x_{30}^{(2)} = P3$, $x_{1гр}^{(2)} = PB$ В/О С/П ... $x_{10}^{(n)} = P1$, $x_{20}^{(n)} = P2$, $x_{30}^{(n)} = P3$, $x_{1гр}^{(n)} = PB$ В/О С/П $PX = I_n = I$.

В качестве контрольного примера вычислим интеграл

$$I = \int_0^1 \int_1^2 \int_2^3 \frac{dx_1\, dx_2\, dx_3}{\ln(x_1 + x_2 + x_3)}.$$

Представив область интегрирования элементарным кубом с $h = 0{,}5$, при $x_{10} = 0$; $x_{20} = 1{,}5$; $x_{30} = 2{,}5$; $x_{1гр} = 1$ по программе 64/34 получим $I = 0{,}6713412$ (время счета около 1 мин). Разбив пространство интегрирования на восемь элементарных кубов с $h = 0{,}25$, при $x_{10}^{(1)} = 0$, $x_{20}^{(1)} = 1{,}25$; $x_{30}^{(1)} = 2{,}25$; $x_{1гр}^{(1)} = 1$ получим $I_1 = 0{,}18161187$ (время счета около 2 мин), при $x_{10}^{(2)} = 0$; $x_{20}^{(2)} = 1{,}25$; $x_{30}^{(2)} = 2{,}75$ получим $I_2 = 0{,}34863694$, при $x_{10}^{(3)} = 0$, $x_{20}^{(3)} = 1{,}75$; $x_{30}^{(3)} = 2{,}25$ получим $I_3 = 0{,}51566201$, при $x_{10}^{(4)} = 0$, $x_{20}^{(4)} = 1{,}75$; $x_{30}^{(4)} = 2{,}75$ получим $I_4 = I = 0{,}67154489$.

2. ИНТЕГРИРОВАНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА

Решение обыкновенных дифференциальных уравнений первого порядка $dy/dx = f(x, y)$ с начальными условиями x_0 и $y_0 = y(x_0)$ при использовании программируемых микрокалькуляторов удобно находить методами численного интегрирования, основанными на последовательном вычислении значений решения y_{i+1} с шагом интегрирования $h = x_{i+1} - x_i$ по соответствующему разностному уравнению [1, 3, 6].

Методы численного интегрирования делят на одношаговые и многошаговые. Из первых чаще всего используют методы Рунге—Кутта различного порядка с последовательным вычислением приближенных значений

$$y_{i+1} = y_i + \sum_{j=1}^{q} p_j g_j(h), \quad i = 0, 1, 2, \ldots, \tag{6.6}$$

где $g_j(h) = hf(x_i + \alpha_j h,\ y_i + \beta_j h)$; p_j, α_j, β_j — коэффициенты. Порядком называют наибольший порядок равных нулю производных погрешностей на одном шаге вычислений.

Методом Рунге—Кутта первого порядка является классический метод Эйлера с вычислениями по формуле

$$y_{i+1} = y_i + hf(x_i,\ y_i),\ i = 0,\ 1,\ 2,\ \ldots, \qquad (6.7)$$

отличающейся большой методической погрешностью, пропорциональной h^2. Поэтому обычно используют методы Рунге—Кутта более высокого порядка с большим уменьшением погрешности при уменьшении шага.

К одношаговым методам второго порядка относится модифицированный метод Эйлера с вычислениями по формуле

$$y_{i+1} = y_i + hf(x_i + h/2,\ y_i + hf(x_i,\ y_i)/2),\ i = 0,\ 1,\ 2,\ \ldots,$$

реализованный следующей базовой программой.

Программа 65/34. Решение уравнения $y' = f(x, y)$ модифицированным методом Эйлера

```
П4 ПП 09 ПП 09  +  С/П БП 00 П9
... ИП7 × ↑ ИП4 + ИП7 ИП8 + П8
→ В/О
```

Инструкция: заменить многоточие операторами вычисления (при $x = P8$, $y = P9$) функции $f(x, y)$; $h/2 = P7$, $x_0 = P8$, $y_0 = P9$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Для проверки правильности ввода и выполнения этой и следующих программ можно воспользоваться данными табл. 15, где приведены результаты решения дифференциального уравнения с шагом $h = 0{,}2$ для указанных начальных условий, абсолютная погрешность результата при $x = 1$ и время выполнения программы.

В усовершенствованном методе Эйлера — Коши, также относящемся к методам Рунге — Кутта второго порядка, решение находят по формуле

$$y_{i+1} = y_i + h(f(x_i,\ y_i) + f(x_{i+1},\ y_i + hf(x_i,\ y_i))/2,\ i = 0,\ 1,\ 2, \ldots,$$

реализованной в следующей программе.

Программа 66/34. Решение уравнения $y' = f(x, y)$ усовершенствованным методом Эйлера—Коши

```
ПП 20  П4  +  П9 ИП7 ИП8 +  П8 ПП
20  +  ИП4 +  2  ÷  П9 С/П БП 00
... ИП7 ×  ИП9 В/О
```

Инструкция: заменить многоточие операторами вычисления (при $x = P8$, $y = P9$) функции $f(x, y)$; $h = P7$, $x_0 = P8$, $y_0 = P9$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

15. Решение дифференциального уравнения $y' = y - 4x + 3$ в интервале [0,1] с шагом $h = 0{,}2$ при начальных условиях $x_0 = 0$, $y_0 = 3$

Способ решения	y_i при переменной x_i						$\Delta(1)$	t
	0	0,2	0,4	0,6	0,8	1,0		
Точное решение $y = 2e^x + 4x + 1$	3	4,2428056	5,5836494	7,0442376	8,6510818	10,436564	0	—
Метод Эйлера	3	4,2	5,48	6,856	8,3472	9,97664	0,459924	—
Программа 65/34	3	4,24	5,5768	7,031696	8,6306691	10,405416	0,031148	15 с
Программа 66/34	3	4,24	5,5768	7,031696	8,630669	10,405416	0,031148	17 с
Программа 67/34	3	4,2426666	5,5833101	7,043616	8,6500696	10,435018	0,001546	25 с
Программа 68/34	3	4,2428	5,583636	7,0442129	8,6510416	10,436503	0,000061	37 с
Программа 69/34	3	4,2436254	5,5812967	7,0376641	8,6385031	10,437102	0,020912	16 с
Программа 70/34	3	4,2428532	5,583767	7,0444536	8,6514341	10,436459	—0,000538	22 с
Программа 71/34	3	—	—	—	—	10,434929	0,001635	5 мин

Результаты вычислений по этой и предыдущей программам, приведенные в табл. 15, практически совпадают, но для других уравнений они могут быть различными. Более точные результаты при соответственно больших затратах времени обеспечивают методы Рунге—Кутта третьего порядка, например, с вычислениями [1] по формуле

$$y_{i+1} = y_i + (g_1 + 4g_2 + g_3)/3, \quad i = 0, 1, 2, \ldots,$$

где $g_1 = hf(x_i, y_i)/2$; $g_2 = hf(x_i + h/2, y_i + g_1)/2$; $g_3 = hf(x_i + h\ y_i - 2g_1 + 4g_2)/2$.

Автоматизацию вычислений по этой формуле обеспечивает следующая программа.

Программа 67/34. Решение уравнения $y' = f(x, y)$ методом Рунге—Кутта третьего порядка

ПП	37	П4	ИП9	П5	+	П9	ПП	33	4
×	П6	ИП4	2	×	—	ИП5	+	П9	ПП
33	ИП4	+	ИП6	+	3	÷	ИП5	+	П9
С/П	БП	00	ИП7	ИП8	+	П8	...	ИП7	×
В/О									

Инструкция: заменить многоточие операторами вычисления (при $x = P8$, $y = P9$) функции $f(x, y)$; $h/2 = P7$, $x_0 = P8$, $y_0 = P9$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Для решения дифференциальных уравнений с помощью универсальных ЭВМ обычно используют метод Рунге—Кутта четвертого порядка с вычислениями по формуле

$$y_{i+1} = y_i + (g_1 + 2g_2 + 2g_3 + g_4)/3,\ i = 0,\ 1,\ 2,\ \ldots,$$

где $g_1 = hf(x_i,\ y_i)/2$; $g_2 = hf(x_i + h/2;\ y_i + g_1)/2$; $g_3 = hf(x_i + h/2;\ y_i + g_2)/2$, $g_4 = hf(x_i + h;\ y_i + 2g_3)/2$.

Эта формула достаточно просто реализуется и на входном языке ЯМК34.

Программа 68/34. Решение уравнения $y' = f(x, y)$ методом Рунге—Кутта четвертого порядка

Сх	П4	ПП	31	ПП	27	+	П4	ПП	31
+	П4	ХУ	ИП9	+	П9	ПП	27	3	÷
ИП6	+	П6	П9	С/П	БП	00	ИП7	ИП8	+
П8	...	ИП7	×	↑	ИП6	+	П9	ХУ	↑
↑	ИП4	+	П4	В/О					

Инструкция: заменить многоточие операторами вычисления (при $x = P8$, $y = P9$) функции $f(x, y)$; $h/2 = P7$, $x_0 = P8$, $y_0 = P9 = P6$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Из многошаговых (конечно-разностных) методов численного интегрирования часто используют методы Адамса с формулами прогноза и коррекции

$$f_{п\,i+1} = \sum_{s=0}^{m} \gamma_s \Delta^s f_{i-s};\quad y_{i+1} = y_i + h \sum_{s=0}^{m} \bar{\gamma}_s \Delta^s f_{i+1-s},\ i = 0,\ 1,\ 2,\ \ldots,$$

где $f_k = f(x_k,\ y_k)$; $\Delta^s f_k = \Delta^{s-1} f_k - \Delta^{s-1} f_{k-1}$; $\gamma_s = 1;\ 1/2;\ 5/12;\ 3/8;\ \ldots$ $\ldots,\ \bar{\gamma}_s = 1;\ -1/2;\ -1/12;\ -1/24;\ \ldots$ [1]. При $m = 1$ формулы прогноза и коррекции

$$f_{п\,i+1} = 3f_i - f_{i-1},\ y_{i+1} = y_i + h(f_i + f_{i+1})/2$$

реализуются следующей программой.

Программа 69/34. Решение уравнения $y' = f(x, y)$ методом Адамса первого порядка

ПП	23	3	×	ХУ	—	ИП9	П5	+	П9
ИП7	ИП8	+	П8	ПП	23	+	ИП5	+	П9
С/П	БП	00	...	ИП7	×	2	÷	ИП4	ХУ
П4	В/О								

Инструкция: заменить многоточие операторами вычисления (при $x = P8$, $y = P9$) функции $f(x, y)$; $h = P7$, $x_0 = P8$, $y_0 = P9$, $hf_{-1}/2 = hf(x_{-1},\ y_{-1})/2 = P4$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Для проверки программы можно воспользоваться данными табл. 15 при $hf_{-1}/2 = 0{,}56374614$.

При $m = 2$ формулы прогноза и коррекции

$$f_{\text{п}\,i+1} = 23f_i - 16f_{i-1} + 5f_{i-2}, \quad y_{i+1} = y_i + h(5f_{i+1} + 8f_i - 8f_{i-1})/12$$

реализуются следующей программой.

Программа 70/34. Решение уравнения $y' = f(x, y)$ методом Адамса второго порядка

ПП	41	ИП5	5	×	ИП4	П5	1	6	×
—	XY	П4	2	3	×	+	ИП9	П6	+
П9	ИП7	ИП8	+	П8	ПП	41	5	×	ИП4
8	×	+	ИП5	—	ИП6	+	П9	С/П	БП
00	...	ИП7	×	1	2	÷	В/О		

Инструкция: заменить многоточие операторами вычисления (при $x = P8$, $y = P9$) функции $f(x, y)$; $h = P7$, $x_0 = P8$, $y_0 = P9$, $hf_{-1}/12 = P4$, $hf_{-2}/12 = P5$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Для проверки программы можно воспользоваться данными табл. 15 при $hf_{-1}/12 = 0{,}09395769$ и $hf_{-2}/12 = 0{,}089010665$, из которых следует, что метод Адамса второго порядка обеспечивает бóльшую точность решения, чем метод Рунге—Кутта третьего порядка. Однако использовать многошаговые методы целесообразно только при заданных начальных условиях, иначе их приходится находить с высокой точностью одношаговыми методами высокого порядка.

Если после введения в программу операторов вычисления функции $f(x, y)$ программная память оказывается незаполненной, приведенные программы решения дифференциальных уравнений можно модифицировать для большего удобства их выполнения. Например, обычно нет необходимости в регистрации результата, высвечиваемого после каждого шага, и в программу можно ввести счетчик шагов с прекращением вычислений лишь после выполнения заданного числа шагов. Если содержимое этого счетчика восстанавливается пользователем, то при выделении для счетчика регистра *О* следует перед оператором С/П поместить оператор цикла L0 00. Для автоматического восстановления содержимого регистра *О* следует ввести в начале программы операторы ИПД П0 (при занесении содержимого счетчика в регистр *Д* перед началом вычислений) и изменить адрес перехода в операторе цикла на 02. Можно также сократить программу на один шаг, заменив оператор безусловного перехода БП 00 оператором косвенного безусловного перехода КБПN при $0 = PN$ или, при автоматическом восстановлении содержимого счетчика, $02 = PN$. Однако при включении счетчика шагов в базовую программу 65/34 следует учитывать, что перед каждым ее пуском в регистре *X* должно находиться значение $y(x_0 + ih)$.

Существенным требованием является обеспечение заданной предельной точности результата вычислений на каждом шаге. Можно реализовать эту задачу повторными вычислениями с уменьшенным шагом и коррекцией результата.

Точность решения дифференциального уравнения одношаговым методом можно оценить, сравнивая результат выполнения одного шага с результатом вычислений после двух, в два раза меньших, шагов. Если разность результатов больше заданной погрешности ε, то вычисления повторяют с меньшим шагом. По разности $|y_{i+1}^{(2)} - y_{i+1}^{(1)}|$ можно судить и о возможности увеличения шага и, следовательно, уменьшения времени счета. По значениям $y_{i+1}^{(2)}$ и $y_{i+1}^{(1)}$ для отличающихся вдвое шагов оценка главного члена погрешности

$$y_{i+1}^{(1)} - y(x+h) = Kf^{(s)}(\theta)\,h^s,$$

где $y(x+h)$ — точное решение; K — зависящий от выбранного метода интегрирования коэффициент пропорциональности; аргумент производной $f^{(s)}$ находится в пределах $x_i \leqslant \theta \leqslant x_i + h$.

Полагая, что при выбранном h погрешность изменяется медленно, примем при двух половинных шагах

$$y_{i+1} - y(x+h) = 2Kf^{(s)}(\theta)\,(h/2)^s = Kf^{(s)}(\theta)\,h^s/2^{s-1},$$

откуда $Kf^{(s)}(\theta)h^s = (y_{i+1}^{(1)} - y_{i+1}^{(2)})/(1-1/2^{s-1}) = 2^{s-1}(y_{i+1}^{(1)} - y_{i+1}^{(2)})/(2^{s-1}-1)$ и решение целесообразно уточнять по формуле

$$y_{i+1} = y_{i+1}^{(2)} + (y_{i+1}^{(2)} - y_{i+1}^{(1)})/(2^{s-1}-1),$$

обеспечивающей устранение главного члена погрешности.

При выборе допустимого значения ε следует учитывать накопление погрешностей в процессе интегрирования и при изменении h соответственно изменять ε.

Реализация рассмотренной коррекции при использовании модифицированного метода Эйлера с автоматическим выбором шага на входном языке ЯМК34 занимает 75 шагов и, следовательно, пригодна лишь для простейших дифференциальных уравнений. Поэтому воспользуемся более сложным алгоритмом коррекции, реализованным в следующей программе с меньшим числом базовых операторов.

Программа 71/34. Решение дифференциального уравнения $y' = f(x, y)$ комбинированным методом с коррекцией и автоматическим выбором шага при заданных значениях $x_{зад}$ и предельной погрешности на шаге

П0	ИП8	—	П1	ИП1	2	÷	П1	ИП9	П5
ИП8	ПП	49	П6	ПП	45	+	ПП	45	ИП6
+	ИП9	—	ИП5	—	3	÷	↑	x^2	ИП7
—	$x<0$	04	XY	ИП5	+	П9	ИП4	П8	ИП0
—	$x=0$	08	ИП9	С/П	П5	ИП4	ИП1	+	П4
...	ИП1	×	↑	ИП9	+	В/О			

Инструкция: заменить в программе многоточие операторами вычисления (при $x = P4$, $y = P5$) функции $f(x, y)$; $\varepsilon^2 = P7$, $x_0 = P8$, $y_0 = P9$, $\dot{x}_{зад} = PX$ В/О С/П $PX = y(x_{зад})$, $P1 = h_i/2$.

Оценка полной погрешности результата определяется по вычисленному значению $h_i/2$ числом шагов $n \approx 2(x_{зад} - x_0)/h_i$ и заданным

значением ε. Так, после решения по этой программе уравнения из табл. 15 при $\varepsilon = 0{,}001$ по значению $P1 = 0{,}0625$ находим $n = 2/h_i \approx 16$ и оценку полной погрешности $n\varepsilon = 1{,}6 \cdot 10^{-2}$, значительно превосходящую истинное значение погрешности. Из сравнения данных табл. 15 следует, что модифицировав, например, программу 68/34 добавлением счетчика числа шагов интегрирования, можно получить более точный результат с меньшими затратами времени, чем при коррекции формул низкого порядка, реализованной в программе 71/34.

Методика численного интегрирования дифференциальных уравнений с частными производными [6, 20] близка к методике вычисления кратных интегралов и сводится к изменению на каждом шаге координат пространства переменных по определенным шаблонам, выбираемым для определенного типа уравнения. Эта методика сравнительно просто реализуется на входных языках микрокалькуляторов, но небольшая емкость памяти последних ограничивает возможность применения достаточно точных методов интегрирования.

3. РЕШЕНИЕ СИСТЕМ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Решение системы дифференциальных уравнений первого порядка

$$dy_k/dx = f_k \equiv f_k(x,\ y_1,\ \ldots,\ y_n),\ k = 1,\ 2,\ \ldots,\ n \qquad (6.8)$$

сводится к вычислению на каждой итерации значений $y_{k,\,i+1}$ всех искомых переменных одним из рассмотренных методов. Выбор метода, предельные число и сложность уравнений зависят от емкости памяти микрокалькулятора используемого типа. На входном языке ЯМК34 решение систем, содержащих до четырех, пяти и даже шести несложных уравнений, можно реализовать модифицированным методом Эйлера по образцу следующей программы.

Программа 72/34. Решение системы из четырех уравнений $y_k' = f_k$ модифицированным методом Эйлера

ИП4	П8	ИП3	П7	ИП2	ПП	39	+	П0	ИП9
2	×	П9	ИП4	ИП8	П4	XY	П8	ИП3	ИП7
П3	XY	П7	ИП2	ИП6	П2	XY	ПП	39	2
÷	П9	+	П0	ИП5	П1	С/П	БП	00	П6
...	ИП9	×	ИП4	+	П4	...	ИП9	×	ИП3
+	П3	...	ИП9	×	ИП2	+	П2	...	ИП9
×	ИП1	+	П5	ИП0	ИП9	В/О			

Инструкция: заменить многоточия операторами вычисления (при $x = P0$, $y_1 = P5$, $y_2 = P6$, $y_3 = P7$, $y_4 = P8$) соответственно функций f_4, f_3, f_2 и f_1; $h/2 = P9$, $x_0 = P0$, $y_{10} = P1 = P5$, $y_{20} = P2$, $y_{30} = P3$, $y_{40} = P4$ В/О С/П $PX = P1 = y_{11}$, $P2 = y_{21}$, $P3 = y_{31}$, $P4 = y_{41}$ ... С/П $PX = y_{1,\,i+1}$, $P2 = y_{2,\,i+1}$, $P3 = y_{3,\,i+1}$, $P4 = y_{4,\,i+1}$.

При меньшем числе уравнений точность решения системы повышается, если использовать методы интегрирования более высокого порядка [6].

Программа 73/34. Решение системы из уравнений $y' = f(x, y, z)$ и $z' = \varphi(x, y, z)$ методом Рунге—Кутта четвертого порядка

ПП	52	ПП	*A*	ПП	48	Вх	+	П6	ПП
A	Вх	+	П5	ПП	52	Вх	+	П6	Вх
ИП0	+	П0	ПП	*A*	Вх	+	П5	Вх	ИП1
+	П1	ПП	48	П6	3	÷	П0	ПП	*A*
П5	3	÷	П1	П3	ИП2	П4	С/П	ИП7	ИП8
+	П8	...	ИП7	×	↑	ИП4	+	П0	→
ИП6	ХУ	+	П6	В/О	...	ИП7	×	↑	ИП0
П2	→	ИП3	+	П1	ХУ	ИП5	ХУ	+	П5
В/О									

Инструкция: заменить в программе символ *A* адресом начала второй подпрограммы — многоточия — операторами вычисления (при $x = P8$, $y = P1$, $z = P2$ и использовании регистров *9*, *A*, *B*, *C* и *Д*) функций φ и f соответственно; $h/2 = P7$, $x_0 = P8$, $y_0 = P1 = P3$, $z_0 = P2 = P4$, $3y_0 = P5$, $3z_0 = P6$ В/О С/П $PX = P2 = z\ (x_0 + h)$, $PY = P1 = y\ (x_0 + h)$... В/О С/П $PX = z\ (x_0 + ih)$, $PY = y\ (x_0 + ih)$.

В качестве контрольного примера рассмотрим решение системы уравнений $y' = z$, $z' = 1 - 3y - 4z$ с начальными условиями $x_0 = y_0 = 0$, $z_0 = 3$. Заменив многоточия в программе соответственно операторами вычисления функций 1 ИП1 3 × — ИП2 4 × — и ИП2, а символ *A* адресом 73, при исходных данных 0,025 = *P7*, 0 = *P1* = *P3* = *P5* = *P8*, 3 = *P2* = *P4*, 9 = *P6* получим (время счета около 70 с) значения $z\ (0{,}05) = 2{,}4916049$; $y\ (0{,}05) = 0{,}1369513$; $z\ (0, 1) = 2{,}0584397$; $y\ (0, 1) = 0{,}25041171$; $z\ (0{,}15) = 1{,}6898102$; $i\ (0{,}15) = 0{,}34386863$; $z\ (0{,}2) = 1{,}3765221$; $y\ (0{,}2) = 0{,}4203132$ с пятью верными цифрами.

Решение обыкновенного дифференциального уравнения *n*-го порядка

$$a_n d^n y/dx^n + a_{n-1} d^{n-1} y/dx^{n-1} + \cdots + a_1 dy/dx + a_0 y = b \quad (6.9)$$

или

$$d^n y/dx^n = f(x, y, y', y'', \ldots, y^{(n-1)})$$

подстановками $y_1 = y$, $y_2 = y_1'$, $y_3 = y_2'$, ..., $y_n = y_{n-1}'$ сводят к решению системы уравнений первого порядка

$$y_1' = y_2,\ y_2' = y_3,\ \ldots,\ y_{n-1}' = y_n,\ y_n = f(x, y_1, \ldots, y_n). \quad (6.10)$$

Следовательно, при программной реализации решения дифференциального уравнения *n*-го порядка как системы уравнений первого порядка достаточно обеспечить вычисление y_n', потому что определение правых частей остальных уравнений сводится к вызову переменных y_k из памяти.

В качестве примера рассмотрим решение уравнений четвертой степени

$$y^{IV} + 10y''' + 35y'' + 50y' + 24y = 1$$

с начальными условиями $x_0 = y(x_0) = 0$, $y'(x_0) = 2$, $y''(x_0) = -4$, $y'''(x_0) = 2$.

Представив это уравнение системой уравнений первого порядка $y_1' = y_2$, $y_2' = y_3$, $y_3' = y_4$, $y_4' = 1 - 24y_1 - 50y_2 - 35y_3 - 10y$ и заменив в программе 72/34 многоточия соответственно операторами 1 ИПА ИП5 × — ИПВ ИП6 × ИПС ИП7 × — ИПД ИП8 × — ... ИП8 ... ИП7 ... ИП6 ..., при исходных данных $h/2 = 0{,}005 = P9$, $x_0 = 0 = P0$, $y(0) = 0 = P1 = P5$, $y'(0) = 2 = P2$, $y''(0) = -4 = P3$, $y'''(0) = 2 = P4$, $24 = PA$, $50 = PB$, $35 = PC$, $10 = PД$ получим (время счета около 45 с) $y(0{,}01) = 0{,}0198$; $y(0{,}02) = 0{,}039202053$; ...; $y(0{,}1) = 0{,}18040512$. Точное значение $y(0{,}1) = 0{,}180489$ для заданного уравнения определяется его аналитическим решением $y = 1 - e^{-x} + 2e^{-2x} - 3e^{-3x} + e^{-4x}$.

При решении дифференциального уравнения n-го порядка по эквивалентной системе уравнений первого порядка для сокращения длины программы (соответствующего увеличению свободных ячеек программной памяти для записи операторов вычисления функции) и времени ее выполнения целесообразно воспользоваться следующим алгоритмом:

1. Вычислить для всех переменных

$$y_{k,\ i+1/2} = y_{ki} + hf_k(x_i,\ y_{1i},\ \ldots,\ y_{ni})/2;\ k = 1,\ 2,\ \ldots,\ n.$$

2. Вычислить согласно модифицированному методу Эйлера

$$y_{n,\ i+1} = y_{ni} + hf_n(x_i + h/2,\ y_{1,\ i+1/2},\ \ldots,\ y_{n,\ i+1/2}).$$

3. Значения остальных переменных вычислить в порядке убывания индексов согласно усовершенствованному методу Эйлера—Коши

$$y_{k,\ i+1} = y_{k,\ i+h/2} + hy_{k+1,\ i+1}/2,\ k = n-1,\ n-2,\ \ldots,\ 1.$$

Для решения уравнения четвертого порядка этот алгоритм реализуется следующей программой, которая может служить образцом для решения уравнений другого порядка.

Программа 74/34. Решение дифференциального уравнения четвертого порядка комбинированным методом второго порядка

ИП4	П5	ПП	38	ПП	14	ПП	38	+	ПП
14	С/П	БП	00	П4	ИП0	ИП9	+	П0	ИП5
ИП9	×	ИП3	+	П3	Вх	ИП9	×	ИП2	+
П2	Вх	ИП9	×	ИП1	+	П1	В/О	...	ИП9
×	↑	ИП5	+	В/О					

Инструкция: заменить многоточие операторами вычисления (при $x = P0$, $y_1 = P1$, $y_2 = P2$, $y_3 = P3$, $y_4 = P4$ и использовании регистров *6*, *7*, *8*, *A*, *B*, *C* и *Д*) функции $f(x, y_1, y_2, y_3, y_4)$ эквивалентной системы уравнений первого порядка; $h/2 = P9$, $x_0 = P0$, $y(0) = y_1 = P1$, $y'(0) = y_2 = P2$, $y''(0) = y_3 = P3$, $y'''(0) = y_4 = P4$ В/О С/П $PX = y(x_0 + h)$ С/П $PX = y(x_0 + 2h)$ ... С/П $PX = y(x_0 + ih)$.

При решении заданного выше уравнения четвертого порядка в этой программе следует заменить многоточие операторами 1 ИПА ИП1

× — ИПВ ИП2 × — ИПС ИП3 × — ИПД ИП4 × — и при исходных данных 0,005 = *P9*, 0 = *P0* = *P1*, 2 = *P2*, —4 = *P3*, 2 = *P4*, 24 = *PA*, 50 = *PB*, 35 = *PC*, 10 = *PД* получить (время счета около 35 с) y (0,01) = 0,0199, y (0,02) = 0,039401; ...; y (0,1) = = 0,18133588.

Следовательно, программа 75/34 обеспечила решение заданного уравнения с меньшей точностью, чем программа 72/34, хотя методические погрешности алгоритмов одинаковы и при решении других уравнений результат решения по программе 74/34 может оказаться точнее.

Для решения уравнений третьего или второго порядка программу 74/34 целесообразно сократить, устранив ненужные операторы, но в этом случае для повышения точности решения следует использовать методы численного интегрирования более высокого порядка, в частности, для решения уравнений второго порядка можно использовать метод Рунге—Кутта четвертого порядка [6].

Программа 75/34. Решение уравнения $y''= f(x, y, y')$ методом Рунге—Кутта четвертого порядка

ПП	44	ПП	40	П6	→	ИП5	+	П5	ПП
40	П6	Вх	ИП2	+	П2	→	→	ИП1	+
П1	→	ИП5	+	П5	ПП	44	ИП6	3	÷
П2	П4	ИП5	3	÷	П1	П3	С/П	БП	00
ИП9	ИП0	+	П0	...	ИП9	×	ИП2	ИП9	×
↑	ИП3	+	П1	→	↑	ИП5	+	П5	→
ХУ	↑	ИП4	+	П2	→	ИП6	ХУ	+	П6
Вх	+	В/О							

Инструкция: заменить многоточие операторами вычисления (при $x = P0$, $y = P1$, $y' = P2$ и использовании регистров *7*, *8*, *A*, *B*, *C* и *Д*) функции $f(x, y, y')$; $h/2 = P9$, $x_0 = P0$, $y(x_0) = P1 = P3$, $y'(x_0) = P2 = P4$, $3y(x_0) = P5$, $3y'(x_0) = P6$ В/О С/П $PX = P3 = y(x_0 + h)$, $PY = P4 = y'(x_0 + h)$ С/П $PX = y(x_0 + 2h)$, $PY = y'(x_0 + 2h)$ ... С/П $PX = y(x_0 + ih)$, $PY = y'(x_0 + ih)$.

Для решения по этой программе, например, уравнения $y''= 1 - 3y' - 4y$ при $x_0 = 0$, $y(x_0) = 0$, $y'(x_0) = 3$ заменить многоточие фрагментом 1 ИП1 3 × — ИП2 4 × —, ввести 0 = *P0* = *P1* = = *P3* = *P5*, 3 = *P2* = *P4*, 9 = *P6* и при $h/2 = 0{,}025 = P9$ зарегистрировать (время счета около 1 мин) y (0,05) = 0,1369513; y'(0,05) = = 2,4916049; y (0,1) = 0,25041171; y'(0,1) = 2,0584397; y (0,15) = = 0,34386863; y'(0,15) = 1,6898102; y (0,2) = 0,4203151; y'(0,2) = = 1,3765221.

Если при использовании приведенных программ ресурса памяти недостаточно для вычисления правых частей уравнений, то требования к ресурсу программной памяти можно уменьшить, сократив текст программы (например, изъяв операторы БП 00, как в программе 73/34). Однако более радикальный способ заключается в нормировании уравнений для уменьшения числа коэффициентов и операторов. Нормируя уравнения, в частности, переходом к аргументу $\gamma = x/h$ и шагу $h_\gamma = 1$,

можно существенно сократить текст программы и требования к программной и числовой памяти, так как нет необходимости хранить значения нормированного шага.

При решении дифференциальных уравнений приходится учитывать особенности их численного интегрирования [1] для устранения возможной неустойчивости процесса вычислений. Большинство практических задач моделирования физических объектов с сильными воздействиями сводится к решению дифференциальных уравнений второго порядка, при интегрировании которых может возникнуть численная неустойчивость вследствие накопления погрешностей (ее устраняют переходом к методам более высокого порядка) или свойств моделируемого физического объекта. При анализе процессов в таких объектах, моделируемых дифференциальными уравнениями второго порядка, часто используют представление решения на фазовой плоскости. В книге [13] приведены программы, облегчающие построение фазовых траекторий.

4. ЦИФРОВОЕ МОДЕЛИРОВАНИЕ

Основное преимущество применения ЭВМ и, в частности, программируемых микрокалькуляторов в инженерном проектировании заключается не в ускорении вычислений, а в связанной с ним возможности более точно моделировать свойства физических объектов и происходящие в них процессы. Основная задача математического моделирования заключается в установлении зависимости (2.5) для рассматриваемых свойств объектов. Часто эту задачу решают, описывая воздействия аналитическими или обобщенными функциями во всем заданном интервале изменения независимой переменной $t \geqslant t_0$. Однако подобное описание не всегда осуществимо и тогда возникает задача определения значений реакций y_i по последовательно задаваемым значениям воздействий $x_i = x(t_i)$. Для решения этой задачи предназначены аналоговые ЭВМ, но ее можно решать и на цифровых машинах (программы для вычисления значений реакции по вводимым текущим значениям воздействий называют цифровыми моделями).

Составление цифровой модели сводится к определению математической модели параметра a в соотношении (2.5) и ее реализации на входном языке ЭВМ. Если канал связи допустимо полагать безынерционным, то эта задача сводится к программированию решения линейных или нелинейных уравнений. Наибольшие трудности возникают при заметном влиянии накопления энергии и конечной скорости ее распространения на свойства моделируемого объекта, когда свойства объекта приходится моделировать дифференциальными уравнениями в частных производных по времени и пространственным координатам. Ограничимся моделированием объектов с сосредоточенными параметрами, свойства которых отображают неоднородным дифференциальным уравнением

$$a_n \frac{d^n y}{dt^n} + \cdots + a_1 \frac{dy}{dt} + a_0 y = b_m \frac{d^m x}{dt^m} + \cdots + b_1 \frac{dx}{dt} + b_0 x \quad (6.11)$$

с известными начальными условиями (задача Коши) и задаваемыми значениями воздействия x при $t \geqslant t_0$. Коэффициенты таких уравнений постоянны при слабых воздействиях, но зависят от переменных при нелинейности или параметричности моделируемого канала причинно-следственной связи. В последнем случае параметрическую зависимость коэффициентов от сторонних воздействий обычно отображают функциями независимой переменной t.

Решение относительно сложных по структуре неоднородных уравнений (6.11) с помощью микрокалькулятора с ограниченным ресурсом памяти приходится реализовать наиболее простыми разностными схемами. В простейших случаях для построения таких схем непрерывные воздействия и реакции на них моделируют ступенчатыми функциями, имеющими постоянное значение в интервале $h = t_i - t_{i-1}$, равном шагу интегрирования. Для этого уравнение (6.11) аппроксимируют заменой дифференциалов dt приращениями $\Delta t = h$ и дифференциалов переменных реакции и воздействия — конечными восходящими, нисходящими или центральными разностями [6]. Полученное таким образом аппроксимирующее разностное уравнение преобразуют в расчетную формулу

$$y_{i+1} = \sum_{j=0}^{n-1} A_j y_{i-j} + \sum_{j=0}^{m-1} B_j x_{i-j}, \quad i = 0, 1, 2, \ldots \qquad (6.12)$$

для вычисления значений реакции по заданным начальным условиям и текущим (задаваемым) значениям воздействия $x_i = x(t_0 + ih)$.

Линейное дифференциальное уравнение (6.11) часто удобно представлять изображением по Лапласу [6]

$$a_n p^n y + \cdots + a_1 p y + a_0 y = b_m p^m x + \cdots + b_1 p x + b_0 x \qquad (6.13)$$

или дробно-рациональной функцией комплексной частоты

$$F(p) = \frac{y(p)}{x(p)} = \frac{b_m p^m + \cdots + b_1 p + b_0}{a_n p^n + \cdots + a_1 p + a_0}, \qquad (6.14)$$

представляющей параметр a (передаточную функцию канала связи) в соотношении (2.5) в области оператора Лапласа p.

Для перехода от изображений Лапласа к расчетной формуле (6.12) достаточно заменить произведения степеней p на переменные y и x (соответствующие производным по времени) разностными отношениями и преобразовать полученное таким образом разностное уравнение.

Приближенно представляя функцию (6.14) степенным многочленом [7, 11]

$$y(p)/x(p) \approx c_0 + c_1 p + c_2 p^2 + \cdots + c_q p^q,$$

после его аппроксимации разностным уравнением можно составить расчетную формулу

$$y_i = \sum_{j=0}^{q-1} C_j x_{i-j}, \quad i = 0, 1, 2, \ldots, \qquad (6.15)$$

в правую часть которой входят только значения воздействий, в связи с чем эту формулу называют *нерекурсивной* в отличие от формулы (6.12).

При составлении разностных уравнений по представлениям дифференциальных уравнений (6.11) или (6.13) первую производную обычно аппроксимируют по формуле

$$\mu (y_{i+1} - y_{i-1})\, 2h + (1 - \mu)(y_{i+1} - y_i)/h \approx dy/dt,$$

причем выбор целого числа μ должен обеспечивать сходимость и устойчивость вычислительного процесса. Так, при $\mu = 4$ соответствующая расчетная формула не аппроксимирует дифференциальное уравнение [1] и вычисления не сходятся к искомому результату. Чаще всего принимают $\mu = 0$ или $\mu = 1$, причем в последнем случае порядок разностного уравнения увеличивается на единицу, что может привести к повышению точности результата.

Аппроксимацию производных высших порядков согласуют с аппроксимациями низших производных и физическими условиями задачи. Так, при $\mu = 0$ или $\mu = 1$ вторую производную обычно аппроксимируют разностным отношением

$$\Delta^2 y_{i+1}/h^2 = (y_{i+1} - 2y_i + y_{i-1})/h^2 \approx d^2y/dt^2,$$

а производные высших порядков — восходящими разностными отношениями.

Проверка условий сходимости и устойчивости вычислений по разностным схемам достаточно громоздка [1, 3], но во многих случаях ее удается упростить, сравнивая начальный участок решения по составленной расчетной формуле для стандартного воздействия (например, единичного скачка) с известным аналитическим решением соответствующего дифференциального уравнения при выбранном воздействии. Иногда практически достаточно сравнить начальные части решений, полученных по составленной формуле для нескольких значений шага интегрирования h, с целью проверки сходимости при $h \to 0$. Сходимость вычислений (и точность решения) обычно улучшается при уменьшении шага h, но при этом увеличивается время счета, что заставляет выбирать компромиссное значение шага. Однако оно не может быть большим времени, в течение которого заметно изменяется состояние моделируемого объекта, так как это неизбежно приведет к ошибкам.

Предельная сложность линейных уравнений (6.11) при их решении по разностным схемам в основном ограничена емкостью числовой памяти микрокалькулятора, где необходимо хранить не менее $2(m + n)$ коэффициентов и значений переменных при реализации формулы (6.12) и не менее $2q$ при использовании формулы (6.15). В частности, на входном языке ЯМК34 при 14 регистрах числовой памяти возможна реализация этих формул при различных сочетаниях чисел m и n, удовлетворяющих условию $m + n \leqslant 7$. Например, при $n = 4$, $m = 3$ формула (6.12) реализуется следующей программой.

Программа 76/34. Решение линейного разностного уравнения при $n = 4$, $m = 3$

ПД	ИП3	ИПС	×	ИП2	ИПВ	ПС	×	+	ИП1		
ИПА	ПВ	×	+	ИП0	ИП9	ПА	×	+	ИП8		
ИП6	×	+	ИП7	П8	ИП5	×	+	ИПД	П7		
ИП4	×	+	П9	С/П	БП	00					

Инструкция: $A_0 = P0$, $A_1 = P1$, $A_2 = P2$, $A_3 = P3$, $B_0 = P4$, $B_1 = P5$, $B_2 = P6$, $x_{-1} = P7$, $x_{-2} = P8$, $y_0 = P9$, $y_{-1} = PA$, $y_{-2} = PB$, $y_{-3} = PC$, $x_0 = PX$ В/О С/П $PX = y_1$ ... $x_i = PX$ С/П $PX = y_{i+1}$ ...

По образцу этой программы несложно составить варианты для других соотношений между n и m при $n + m \leqslant 7$. Программу 76/34

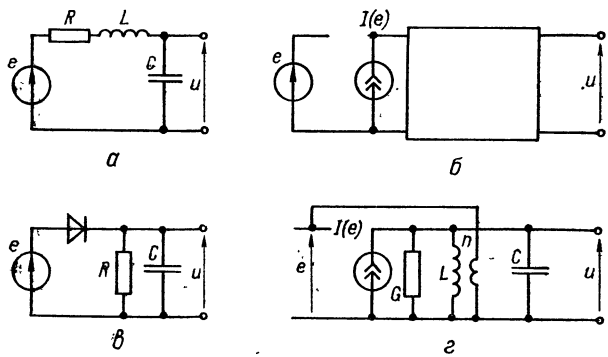


Рис. 36

можно использовать и при $m + n < 7$, очищая регистры памяти для хранения отсутствующих коэффициентов, но для уменьшения времени счета в этом случае целесообразно сократить длину программы, изъяв ненужные операторы.

При решении разностных уравнений для стандартных воздействий целесообразно автоматизировать и вычисление их очередных значений. Например, при воздействии единичным скачком $x_i = 1$, $t_i \geqslant 0$ ($y_{-3} = y_{-2} = y_{-1} = y_0 = x_{-2} = x_{-1} = 0$) достаточно ввести в программу 76/34 между операторами С/П и БП оператор набора цифры 1. В этом случае при повторных пусках программы нажатиями только клавиши С/П в регистр X будет автоматически заноситься очередное значение $x_i = 1$. Аналогично можно ввести между операторами С/П и БП и фрагменты для вычисления значений других стандартных воздействий [13].

В качестве примера составим программу вычисления напряжения $u(t)$ на выходе несложной цепи (рис. 36,*а*) при воздействии напряжением $e(t)$. По матрице слабосигнальных операторных проводимостей цепи

$$Y = \begin{bmatrix} 1/r & -1/r & 0 \\ -1/r & 1/r + 1/pL & -1/pL \\ 0 & -1/pL & pC + 1/pL \end{bmatrix}$$

определим передаточную функцию (коэффициент передачи напряжения)

$$u/e = \Delta_{13}/\Delta_{11} = 1/(LCp^2 + rCp + 1),$$

соответствующую в операторной области параметру a в соотношении (2.5), и составим разностное уравнение

$$LC\,(u_{i+1} - 2u_i + u_{i-1})/h^2 + rC\,(u_{i+1} - u_i)/2h + u_i = e_i.$$

Преобразовав это уравнение в расчетную формулу

$$u_{i+1} = A_0 u_i + A_1 u_{i-1} + B_0 e_i,$$

реализуем ее программой

ИП0 × ИП2 ИП4 × + ИП3 П4 ИП1 ×
+ П3 С/П БП 00

с инструкцией: $B_0 = P0$, $A_0 = P1$, $A_1 = P2$, $u_0 = P3$, $u_{-1} = P4$, $e_0 = PX$ В/О С/П $PX = u_1$ ... $e_i = PX$ С/П $PX = u_{i+1}$ ..., где $A_0 = (2 - h^2/LC)/(1 + rh/2L)$, $A_1 = (hr/2L - 1)/(1 + rh/2L)$, $B_0 = h^2/LC\,(1 + rh/2L)$, $h = \Delta t$.

Для проверки сходимости составленного разностного уравнения при $L = 0{,}1$ Гн; $C = 10^{-6}$ Ф; $r = 160$ Ом примем $h = 0{,}1$ мс и вычислим $B_0 = 0{,}92592592$; $A_0 = 1{,}7592592$; $A_1 = -0{,}85185185$. Выбрав в качестве пробного воздействие единичным скачком ($e_i = 1$, $i \geqslant 0$), получим решение $u\,(t)$ по составленной программе, достаточно точно совпадающее с аналитическим решением, которое при $r < \sqrt{2/C}$ определяется выражением

$$u\,(t) = 1 - \omega_0\, e^{-\alpha t} \sin\,(nt + \beta)/\alpha,$$

где $\omega_0 = 1/\sqrt{LC}$; $n = \sqrt{\omega_0^2 - \alpha^2}$; $\beta = \operatorname{arctg}\,(\beta/\alpha)$; $\alpha = r/2L$.

Полученное совпадение результатов является практическим основанием для использования составленных расчетной формулы и программы при воздействиях любой другой формы, хотя в этих случаях желательно проверять сходимость вычислительного процесса при выбранном значении шага h.

Расчетные формулы не обязательно (а для нелинейных и параметрических каналов связи и нецелесообразно) составлять в форме выражений (6.12) или (6.15) с предварительным вычислением коэффициентов. Выбор оптимального выражения расчетных формул зависит от вида исходного уравнения и особенностей задачи. При моделировании нелинейных и параметрических каналов причинно-следственных связей целесообразно, когда это возможно, разбивать их на участки с линейными инерционными и нелинейными безынерционными свойствами. Передаточные функции линейных участков удобно представлять в операторной области дробно-рациональными функциями (6.14) с последующим переходом к разностной форме и учитывать дополнительно нелинейные зависимости между воздействием и реакциями на остальных участках.

Эта задача достаточно проста в тех случаях, когда нелинейные свойства безынерционных участков не зависят от реакции и определя-

ются только воздействием. Пусть, например, усилитель при сильных сигналах моделируется эквивалентной схемой, содержащей нелинейный участок канала передачи воздействия, отображенный зависимостью тока $I(e) = S(e)e$, где $S(e)$ — крутизна передаточной характеристики, и линейный участок, условно обозначенный прямоугольником (рис. 36,*б*). В этом случае связь между реакцией $u(t)$ и током $I(e)$, являющимся переменной воздействия для линейного участка, моделируется дробно-рациональной функцией оператора p, а нелинейные свойства предыдущего участка моделируются аналитическим выражением или различного рода аппроксимациями. Сшивая фрагменты вычисления реакций на каждом участке, получают программу для вычисления напряжения $u(t)$ при больших уровнях $e(t)$, когда проявляются нелинейные свойства усилителя.

Значительно бо́льшие трудности возникают, когда характеристики нелинейных участков канала причинно-следственной связи зависят от реакции на их выходе. Простейшим примером может служить диодный детектор, схема которого показана на рис. 36,*в*, с зависимостью тока и проводимости $g_д$ диода от разности напряжений $E = e - u$. Свойства такого детектора при слабых воздействиях (малых уровнях напряжения e) можно приближенно описать операторным уравнением

$$u = (e - u)\, g_д/(pC + 1/R),$$

а нелинейную зависимость проводимости диода от приложенного к нему напряжения $E = e - u$ аппроксимировать, например, кусочно-линейной функцией

$$g_д = \begin{cases} g_{д1} \text{ при } E \leqslant 0; \\ g_{д2} \text{ при } 0 < E \leqslant E_п; \\ g_{д3} \text{ при } E_п < E. \end{cases}$$

Записав операторное уравнение детектора $Cpu + u/R = (e - u)\, g_д$ и перейдя к разностному уравнению $C(u_{i+1} - u_i)/h + u_i/R = (e_i - u_i)\, g_д$, составляем расчетную формулу $u_{i+1} = u_i + ((e_i - u_i)\, g_д - u_i/R)\, h/C$, реализуемую следующей программой.

Программа 77/34. Цифровая модель диодного детектирования

ИП9	—	П0	$x < 0$	08	ИП4	БП	16
ИП3	—	$x < 0$	15	ИП5	БП	16	ИП6
ИП0	×	ИП7	ИП9	×	—	ИП8	×
ИП9	+	П9	С/П	БП	00		

Инструкция: $E_п = P3$, $g_{д1} = P4$, $g_{д2} = P5$, $g_{д3} = P6$; $1/R = P7$, $h/C = P8$, $u_0 = P9$, $e_0 = PX$ В/О С/П $PX = u_1 \ldots e_i = PX$ С/П $PX = u_{i+1}$.

Проверим сходимость вычислительного процесса по этой программе для нескольких значений шага интегрирования, убедившись, что при его изменениях в пределах от 0,1 до 0,5 мкс при воздействии единичным скачком погрешность решения не превышает нескольких процентов.

Разностные схемы применимы и для моделирования свойств физических объектов с инерционными каналами, образующими петли обратной связи, например, в автоколебательных системах, где реальное воздействие определяется реакцией на выходе активного компонента, преобразующего энергию источника мощности в энергию автоколебаний. В качестве простейшего примера рассмотрим автогенератор электрических колебаний, свойства которого приближенно отображены эквивалентной схемой, показанной на рис. 36,*е* при аппроксимации тока активного компонента функцией

$$I = I_{\max}(0{,}5 + (\operatorname{arctg}(10e))/\pi).$$

Составив для выходной цепи операторное уравнение $u = 1/(pC + G + 1/pL)$ или $LCp^2u + LCpu + u = LpI$, перейдем к разностному уравнению $LC(u_{i+1} - 2u_i + u_{i-1})/h^2 + LC(u_i - u_{i-1})/h + u_i = L(I_i - I_{i-1})/h$ и расчетной формуле $u_{i+1} = h(I_i - I_{i-1})/C + (2 - h^2/LC - hG/C)u_i - (1 - hG/C)u_{i-1}$. Учитывая соотношение $e = nu$, где n — коэффициент трансформации, а обратная связь обеспечивается сохранением результата вычислений в регистре X перед очередным выполнением программы, составим программу с реализацией заданной аппроксимации тока активного компонента.

Программа 78/34. Цифровая модель автоколебаний

ИП7	ИП8	—	ИП0	×	arctg	π	÷	2	1/*x*
+	ИП1	×	ИП5	П6	XY	П5	ИП6	—	ИП2
×	ИП8	ИП4	×	—	ИП7	П8	ИП3	×	+
П7	С/П	БП	00						

Инструкция: $10n = P0$, $I_{\max} = P1$, $h/C = P2$, $(2 - h^2/LC - hG/C) = P3$, $1 - hG/C = P4$, $I_0 = P5$, $I_{-1} = P6$, $u_0 = P7$, $u_{-1} = P8$ В/О С/П $PX = u_1$ ... С/П $PX = u_i$ С/П $PX = u_{i+1}$ ...

При $n = 0{,}002$; $I_{\max} = 0{,}01$ А; $C = 10^{-8}$ Ф; $L = 10^{-3}$ Гн; $G = 10^{-4}$ См $= 1/10$ кОм, приняв $h = 10^{-6}$ с и вычислив исходные данные $100 = P2$, $1{,}89 = P3$; $0{,}99 = P4$, по программе 78/34 получим ($u_0 = 0{,}0001$ В; $u_{-1} = I_0 = I_{-1} = 0$) начальные значения $u_i = 0{,}5002$; 0,9456; 1,2919; 1,506; ... Продолжение вычислений для нескольких периодов автоколебаний свидетельствует о достаточно мягком установлении квазигармонического режима автоколебаний.

Рассмотренный подход применим не только для электрических, но и для любых других воздействий, связанных с искомой реакцией аналогичными математическими моделями. В некоторых случаях воздействие на входе моделируемого канала представляет собой последовательность импульсов малой длительности, следующих через равные промежутки времени. Кроме того, на входе некоторых цифровых устройств обработки сигналов (цифровых фильтров) непрерывное воздействие преобразуется в последовательность дискретных значений, выбираемых через равные промежутки времени (период дискретизации) из исходного воздействия.

Подобные воздействия отображают решетчатыми функциями $x_i = x(t_i)$, $x(t \neq t_i) = 0$, $t_i = t_0 + ih$, где h — период дискретизации.

Хотя и при ступенчатой аппроксимации непрерывных сигналов для численного интегрирования значения x_i выбираются через равные промежутки времени, принципиальной особенностью решетчатой аппроксимации является ее равенство нулю в промежутках между моментами t_i отсчетов. Однако при $h \to 0$ ступенчатая и решетчатая аппроксимации непрерывных воздействий совпадают и, следовательно, при достаточно малых значениях шага интегрирования h непрерывные воздействия также аппроксимируются решетчатыми функциями. Различие между этими представлениями проявляется лишь при моделировании свойств инерционных каналов, зависящих от значений воздействий как в данный, так и в предыдущие моменты времени.

При решетчатых воздействиях разностные уравнения, аппроксимирующие линейные неоднородные дифференциальные уравнения (6.11), а также рекурсивные и нерекурсивные расчетные формулы, составляют различными способами. В книге [11] для этой цели использовано билинейное преобразование дробно-рациональных передаточных функций (6.14) подстановкой $p = (z - 1)/(z + 1)$ с последующим переходом к разностным уравнениям.

Билинейное преобразование при численном интегрировании исходных дифференциальных уравнений (6.11) приводит к достаточно точным решениям, когда воздействие действительно является решетчатой функцией (например, в цифровых фильтрах [13]). Однако при замене непрерывного воздействия решетчатой функцией погрешность результата решения разностного уравнения определяется не только погрешностью аппроксимации, но и нелинейной зависимостью частоты при билинейном преобразовании, которую приходится дополнительно корректировать [11].

При решетчатом воздействии для составления нерекурсивной формулы численного интегрирования можно непосредственно применить z-преобразование, основанное на переходе к комплексной переменной z в бесконечной сумме, полученной при делении числителя на знаменатель функции (6.14). Однако в этом случае возникает дополнительная погрешность ограничения частного конечным числом слагаемых.

Рассмотрим «стандартное z-преобразование», обеспечивающее точное совпадение дискретных значений реакции на единичный импульс с отсчетом в точках дискретизации импульсной характеристики моделируемого канала связи при непрерывном воздействии. При таком преобразовании передаточную функцию (6. 14) раскладывают на простые дроби (предполагается, что ее знаменатель не имеет кратных корней):

$$F(p) = \sum_{i=1}^{n} A_i/(p - p_i),$$

где p_i — корни знаменателя (полюсы) функции $F(p)$.

После этого каждый член разложения преобразуют по формуле $A_i/(1 - z^{-1} e^{p_i h})$, где h — период дискретизации. Для преобразованных пар членов разложения в простые дроби, соответствующих парам комплексно-сопряженных полюсов $(A_i p + B_i)/(p^2 + 2\sigma p + \sigma^2 + \Omega^2)$

преобразование удобно выполнять по формуле $(A_i - (A_i \cos \Omega h + (A_i\sigma - B_i) \sin(\Omega h)/\Omega) z^{-1} e^{-\sigma h})/(1 - 2z^{-1} e^{-\sigma h} \cos \Omega h + z^{-2} e^{-2\sigma h})$.

Сумму преобразованных членов представляют дробно-рациональной функцией переменной z, представляющей собой z-преобразование передаточной функции с дискретизованными переменными

$$F(z) = \frac{y(z)}{x(z)} = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2} + \cdots + B_m z^{-m}}{1 + A_1 z^{-1} + A_2 z^{-2} + \cdots + A_n z^{-n}}.$$

Так как умножение переменной на множитель z^{-k} соответствует задержке оригинала переменной на k интервалов дискретизации, то, переписав правое равенство

$$(1 + A_1 z^{-1} + \cdots + A_n z^{-n})\, y(z) = (B_0 + B_1 z^{-1} + \cdots + B_m z^{-m})\, x(z),$$

получим разностное уравнение

$$y_i + A_1 y_{i-1} + \cdots + A_n y^{-n} = B_0 x_i + B_1 x_{i-1} + \cdots + B_m x_{i-m},$$

которое легко преобразовать в рекурсивную формулу (6.12).

Многие операции этого алгоритма можно автоматизировать с помощью программ, описанных в пособии [13]. Пусть, например, свойства исследуемого канала связи моделированы дифференциальным уравнением

$$y''' + 4y'' + 30y' + 52y = x$$

или, после преобразования Лапласа,

$$F(p) = y(p)/x(p) = 1/(p + 2)(p^2 + 2p + 26).$$

Разложив эту функцию в простую дробь $F(p) = (1/26)(p + 2) + (p/26)/(p^2 + 2p + 26)$ и приняв $h = 0{,}1$, преобразуем члены разложения $(1/25)/(p + 2) - (1/(1 - z^{-1}e^{-0,2}) = 1/(1 - 0{,}81873072z^{-1})$, $(p/26)/(p^2 + 2p + (1^2 + 5^2) - (1/26 - (\cos 0{,}5/26 + \sin 0{,}5/5 \cdot 26) \times \times e^{-0,1}z^{-2}\cos 0{,}5 + z^{-2}e^{-0,2}) = (0{,}038461538 - 0{,}033878078z^{-1})/(1 - 1{,}5881392z^{-1} + 0{,}81873072z^{-2})$.

Сложив преобразованные члены, получим передаточную функцию

$$F(z) = \frac{0{,}0042854469z^{-1} + 0{,}00375262z^{-2}}{1 - 2{,}4068699z^{-1} + 2{,}1189892z^{-2} - 0{,}67032007z^{-3}}$$

и после перехода к разностному уравнению — расчетную формулу

$$y_i = 2{,}4068699y_{i-1} - 2{,}1189892y_{i-2} + 0{,}67032007y_{i-3} + + 0{,}0042854469x_{i-1} + 0{,}00375262x_{i-2},$$

легко программируемую на любом входном языке.

Разложив знаменатель $F(z)$ на множители, можно записать

$$F(z) = \frac{1}{1 - 0{,}81873072z^{-1}} \cdot \frac{0{,}0042854469z^{-1} + 0{,}00375262z^{-2}}{1 - 1{,}5881392z^{-1} + 0{,}81873072z^{-2}}$$

и представить моделируемый канал связи двумя участками с передаточными функциями, равными полученным множителям низшего порядка. Следовательно, можно представить моделируемый канал участком (звеном) первого порядка с реакцией $y_i = 0{,}81873072y_{i-1} +$

$+ x_i$ и звеном второго порядка с реакцией $y_{i+1} = 1{,}5881392y_i - 0{,}81773072y_{i-1} + 0{,}0042854469y_{i-2} + 0{,}00375262y_{i-3}$. В этом случае, программно реализовав каждую из этих формул отдельным фрагментом, можно составить цифровую модель канала из двух последовательно выполняемых фрагментов. Аналогично составляются цифровые модели при разбиении передаточной функции на большее число множителей.

Рассмотренный метод применим к передаточным функциям со степенью знаменателя, большей степени числителя, и обеспечивает инвариантность преобразования импульсной характеристики в точках дискретизации. Однако он не гарантирует точного совпадения в этих точках непрерывного и дискретного представлений реакции на произвольное воздействие, причем возможное различие этих представлений возрастает при расширении интервала дискретизации.

5. ВЫЧИСЛЕНИЕ СПЕЦИАЛЬНЫХ ФУНКЦИЙ

Любые вычисления сводятся к выполнению последовательности арифметических операций, но во многих случаях причинно-следственные отношения (2.5) отображаются функциональными зависимостями, которые нельзя представить в конечном виде с помощью только арифметических операторов. Наиболее часто встречающиеся из таких зависимостей обозначают определенными символами и называют специальными функциями. К ним относятся и такие элементарные трансцендентные функции как тригонометрические, логарифмические, гиперболические и обратные им.

Во входных языках программируемых микрокалькуляторов предусмотрены операторы автоматического вычисления всех или большинства элементарных функций. Другие специальные функции табулированы и их значения между узлами табличной модели несложно найти методами интерполирования. Однако для этого пользователь должен располагать таблицей нужной функции с требуемым числом верных цифр, причем обращение к таблице нежелательно, когда вычисление специальной функции является частью более сложной задачи, автоматизируемой с помощью микрокалькулятора. Поэтому в библиотеке пользователя должны содержаться программы вычисления нужных ему специальных функций.

Некоторые элементарные функции, например, гиперболические и обратные им, точно выражаются через другие элементарные функции (см. программы 138/34 и 182/21 в приложении). Однако большинство специальных функций нельзя выразить в конечном виде через элементарные функции и для их вычисления используют аппроксимирующие формулы или итерационные процессы, основанные на приближенном представлении функций бесконечными суммами или произведениями, ортогональными многочленами, цепными или рациональными дробями [1, 4, 6, 11, 13, 20]. Выбор метода вычисления зависит от свойств функции, заданного интервала изменения аргумента, требуемой точности результата, особенностей входного языка и допустимых затрат времени.

При составлении прикладных программ, используемых в качестве подпрограмм или фрагментов более сложных программ, для приближенного вычисления специальных функций в ограниченном интервале аргумента обычно используют несложные аппроксимирующие выражения, реализуемые небольшим числом операторов с минимальными затратами времени. При составлении универсальных программ для вычисления специальных функций с высокой точностью во всем интервале изменения аргумента использование эмпирических аппроксимирующих выражений, подобных формуле (4.3), является исключением и приходится прибегать к более общим методам и их комбинациям для различных интервалов аргумента.

Большинство специальных функций порождается дифференциальными или интегральными уравнениями и для их вычисления непосредственно применимы методы численного интегрирования. Примером могут служить нормальные эллиптические интегралы Лежандра (неполные) первого рода

$$F(k, \varphi) = \int_0^{\varphi} d\varphi / \sqrt{1 - k^2 \sin^2 \varphi}$$

и второго рода

$$E(k, \varphi) = \int_0^{\varphi} \sqrt{1 - k^2 \sin^2 \varphi}\, d\varphi,$$

которые при верхнем пределе интегрирования $\varphi = \pi/2$ называют полными.

Воспользовавшись формулой Симпсона с учетом нулевого значения нижнего предела интегрирования, составим следующую программу.

Программа 79/34. Вычисление эллиптических интегралов Лежандра первого рода $F(k, \varphi)$ и второго рода $E(k, \varphi)$

П0	1	П6	ИП8	П3	ИП0	÷	П2	ПП	36
1	ПП	25	4	ПП	25	2	БП	11	ИП6
3	÷	ИП2	×	С/П	×	ИП6	+	П6	КИП0
ИП0	$x \neq 0$	19	ИП2	×	П3	ИП3	sin	ИП7	×
x^2	1	XY	—	$\sqrt{}$	$1/x$	В/О			

Инструкция: установить переключатель Р—Г в положение Р; для вычисления интеграла первого рода ($k = P7$, $\varphi = P8$) четное число $n = PX$ В/О С/П $PX = F(k, \varphi)$; для вычисления интеграла второго рода заменить в программе оператор $1/x$ по адресу 45 оператором НОП.

По этой программе при $n = 10$ время вычисления полного интеграла $F(0{,}5;\ \pi/2) = 1{,}6857501$ составляет около 100 с, при $n = 20$ время вычисления $F(0{,}5;\ \pi/2) = 1{,}6857502$ составляет около 3,5 мин. Следовательно, при $n = 10$ и заданном значении аргумента результат вычислен с семью верными цифрами.

Основной недостаток численного интегрирования заключается в значительном увеличении затрат времени при повышении точности результата вычислений. Поэтому для вычисления специальных функций часто используют их разложение в степенной ряд Тейлора или порождаемые им функциональные ряды. В этом случае минимальная методическая погрешность достигается при суммировании членов ряда в цикле, охватывающем вычисление очередного члена ряда и его сложение с суммой предыдущих членов, с прекращением вычислений, когда очередной член ряда не изменяет вычисленной микрокалькулятором суммы предыдущих членов (критерий максимальной точности).

Увеличение аргумента приводит к возрастанию числа членов ряда (особенно при его медленной сходимости), которые необходимо суммировать для обеспечения минимальной методической погрешности и, следовательно, к увеличению времени счета и операционных погрешностей. Кроме того, некоторые аппроксимирующие ряды сходятся немонотонно (при определенных значениях аргумента члены ряда немонотонно убывают по модулю) и необходимо тщательное исследование условий сходимости.

В связи с этими особенностями аппроксимаций степенными рядами, для вычисления специальных функций, значения которых конечны при стремлении аргумента к бесконечности, в области больших значений аргумента используют асимптотическое разложение

$$F(x) \sim a_0 + a_1/x + a_2/x^2 + a_3/x^3 + \cdots + a_n/x^n + \ldots$$

Члены такого разложения, строго говоря, образуют расходящийся ряд, сумма $S_n(x)$ ограниченного числа n членов которого удовлетворяет условию

$$\lim_{x \to \infty} |F(x) - S_n(x)| = 0.$$

При фиксированном и достаточно большом значении аргумента x погрешность асимптотического разложения с увеличением n вначале уменьшается, но затем начинает возрастать. Поэтому необходимо исследовать зависимость погрешности асимптотического разложения (соответствие которого функции обозначают символом $\sim$) от числа членов.

Корректируя коэффициенты усеченного степенного ряда или асимптотического разложения, можно уменьшить число членов, требуемое для обеспечения заданной точности аппроксимации. Эту коррекцию следует выбирать таким образом, чтобы упростить программную реализацию вычислений и уменьшить число регистров памяти для хранения исходных данных.

Универсальные программы, в которых реализованы различные методы вычисления специальных функций, целесообразно составлять так, чтобы обеспечить автоматический выбор метода вычисления в зависимости от значений аргумента. Для примера рассмотрим методику составления программ вычисления нескольких наиболее употребительных в инженерной практике специальных функций.

Вычисление интегральных функций, в том числе интегрального синуса (6.4), с высокой точностью обеспечивается методами численно-

го интегрирования. Для этого можно воспользоваться базовой программой 58/34, но для вычисления результата с заданной точностью целесообразно упростить автоматизацию выбора числа n разбиений интервала интегрирования в зависимости от значения аргумента. Установив, например, прямыми расчетами число n, обеспечивающее получение результата не менее чем с пятью верными цифрами при различных значениях аргумента, автоматизируем выбор n для заданного значения аргумента интегрального синуса по следующему условию: если $x < 1$, то принять $n = 4$,иначе принять $n = 4\,\mathrm{E}\,(x)$, где $\mathrm{E}\,(x)$ — целая часть значения аргумента.

Программа 80/34. Вычисление интегрального синуса по формуле Симпсона с автоматическим выбором числа n разбиений

П9	1	—	$x<0$	08	4	БП	14	ИП9	ПД
КИПД	ИПД	4	×	П0	1	П6	ИП9	П8	ИП0
÷	П7	ПП	50	1	ПП	39	4	ПП	39
2	БП	25	ИП6	3	÷	ИП7	×	С/П	×
ИП6	+	П6	КИП0	ИП0	$x\neq 0$	33	ИП7	×	П8
ИП8	sin	ИП8	÷	В/О					

Инструкция: установить переключатель Р—Г в положение Р, $x = PX$ В/О С/П $PX = \mathrm{Si}\,(x)$.

По этой программе получим Si (1) = 0,9460869 (время счета около 35 с), Si (2) = 1,6054182 (время счета около 70 с), Si (10) = 1,658349 (время счета около 5 мин), Si (20) = 1,5482414 (время счета около 10 мин).

Так как затраты времени при вычислениях по этой программе значительно возрастают при увеличении аргумента, попытаемся составить универсальную программу с помощью усеченных рядов. Интегральный синус аппроксимируют усечением бесконечного ряда [6]

$$\mathrm{Si}\,(x) = \sum_{n=0}^{\infty} (-1)^n x^{2n+1}/(2n+1)!\,(2n+1),$$

для суммирования членов которого выберем следующий алгоритм:

1. Принять $i = 0$, $a_0 = S_0 = x$, $k = 1$.
2. Вычислить $a_{i+1} = a_i(-x^2)/(k+1)(k+2)^2$, $S_{i+1} = S_i + a_{i+1}$, $k = k + 2$.
3. Если $S_{i+1} - S_i = 0$, то перейти к п. 4, иначе к п. 2.
4. Принять $\mathrm{Si}\,(x) = S_{i+1}$.

Разместив значения S_i, x^2 и k соответственно в регистрах *2*, *3* и *4*, реализуем этот алгоритм на входном языке ЯМК34 программой

П2	x^2	П3	ИП2	1	П4	→	КИП4	→	ИП4
÷	КИП4	→	ИП4	÷	ИП3	/—/	×	↑	ИП4
÷	ИП2	+	П2	Вх	—	$x=0$	06	ИП2	С/П

с инструкцией: $x = PX$ В/О С/П $PX = \mathrm{Si}\,(x)$.

По этой программе получим Si (0,1) = 0,099944467 (время счета около 20 с), Si (1) = 0,94608314 (время счета около 40 с), Si (10) = = 1,658364 (время счета около 2 мин).

Для исследования свойств аппроксимирующего ряда при различных значениях аргумента введем в составленную программу оператор С/П между операторами ÷ и ИП2 с адресами 20 и 21. Выполнив вычисления, устанавливаем, что при $x \leqslant 4$ члены ряда монотонно убывают по модулю и максимален первый член, но при $x = 8$ и $x = 9$ члены ряда вначале возрастают по модулю и лишь затем начинают убывать. Так как максимальные члены в рассмотренных случаях содержат соответственно 6 и 5 разрядов после запятой, то абсолютная операционная погрешность будет достигать соответственно значений 10^{-6} и 10^{-5}. При $x > 9$ эта погрешность, несмотря на выбор критерия максимальной точности для прекращения вычислений, будет еще больше.

Так как $\mathrm{Si}(\infty) = \pi/2$, то при больших значениях аргумента целесообразно воспользоваться асимптотическим разложением

$$\mathrm{Si}(x) \sim \frac{\pi}{2} - \frac{\cos x}{x} \sum_{n=0}^{\infty} (-1)^n \frac{2n!}{x^{2n}} - \frac{\sin x}{x} \sum_{n=0}^{\infty} (-1)^n \frac{(2n+1)!}{x^{2n+1}} =$$

$$= \pi/2 + (\cos x/x)(1 - 2!/x^2 + 4!/x^4 - 6!/x^6 + 8!/x^8 - \ldots) -$$

$$- (\sin x/x^2)(1 - 3!/x^2 + 5!/x^4 - 7!/x^6 + 9!/x^8 - \ldots).$$

Вычислив члены рядов в скобках при $x = 8$, убеждаемся, что максимальная точность достигается при сохранении только убывающих по модулю первых пяти членов в первых скобках и четырех во вторых скобках. Составив программу для вычисления $\mathrm{Si}(x)$ при сохранении этих членов разложения и подобрав по результатам вычислений коэффициенты последних сохраняемых членов рядов, составим расчетную формулу

$$\mathrm{Si}(x) = (((21/x^2 - 1)\,120/x^2 + 6)\,x^2 - 1)(\sin x)/x^2 - ((((38/x^2 -$$

$$- 1)\,30/x^2 + 1)\,24/x^2 - 2)/x^2 + 1)(\cos x)/x + \pi/2.$$

Реализовав вычисления по этой формуле при хранении значений $1/x^2$ в операционном стеке и совместив эту реализацию с составленной ранее программой степенной аппроксимации при $x < 8$, получим следующую универсальную программу.

Программа 81/34. Вычисление интегрального синуса суммированием усеченных рядов*

П2	x^2	П3	ИП2	ИП2	8	—	$x<0$	37	→
1	П4	→	КИП4	→	ИП4	÷	КИП4	→	ИП4
÷	ИП3	/—/	×	↑	ИП4	÷	ИП2	+	П2
Вх	—	$x=0$	12	ИП2	БП	94	ИП3	$1/x$	↑
↑	↑	3	8	×	1	—	×	3	0
×	1	+	×	2	4	×	2	—	×
1	+	ИП2	÷	ИП2	cos	×	П9	→	2
1	×	1	—	×	1	2	0	×	6
+	×	1	—	×	ИП2	sin	×	ИП9	—
π	2	÷	+	С/П	БП	00			

* В приложении приведен вариант этой программы 136/34.

Инструкция: установить переключатель Р—Г в положение Р, $x = PX$ (В/О) С/П $PX = \mathrm{Si}\,(x)$. . . . . .

По этой программе получим Si (1) = 0,94608314 (время счета около 40 с), Si (2) = 1,605413 (время счета около 50 с), Si (10) = 1,6581546 (время счета около 23 с), Si (20) = 1,5482416 (время счета около 23 с), Si (7,9) = 1,56167 (время счета около 1 мин 45 с), Si (8) = 1,574188 (время счета около 23 с). Сравнение с табличными данными [6] свидетельствует, что эти результаты содержат не менее шести верных значащих цифр, а время счета меньше (и при $x < 8$ постоянно), чем при вычислениях методом численного интегрирования по программе 80/34.

В инженерных приложениях часто встречается интеграл вероятности (3.9), при вычислении которого по программе 6/34 время счета быстро растет при увеличении аргумента. Непосредственное численное интегрирование при $x \geqslant 3$ в этом случае также связано со значительными затратами времени. Поэтому интеграл (3.9), имеющий значение $\Phi(\infty) = 1$, при значениях аргумента $x \geqslant 3$ целесообразно вычислять с помощью разложения

$$1 - \Phi(x) \sim ((((3{,}7/x^2 - 1)\,15/x^2 + 3)/x^2 - 1)/x^2 + 1)\sqrt{2e^{x^2}/\pi x^2},$$

обеспечивающего погрешность не более $5 \cdot 10^{-7}$.

Для составления универсальной программы совместим с помощью условного оператора программу 6/34 и фрагмент, реализующий приведенную формулу при $x \geqslant 3$.

Программа 82/34. Вычисление интеграла вероятности $\Phi(x)$ суммированием рядов

П0	П1	x^2	П2	9	—	$x<0$	38	Сх	П4
КИП4	ИП0	ИП1	/—/	ИП2	×	ИП4	2	×	÷
П1	Вх	1	+	÷	+	П0	—	$x=0$	10
ИП0	2	π	÷	$\sqrt{}$	×	БП	77	ИП2	$1/x$
↑	↑	↑	3	,	7	×	1	—	×
1	5	×	3	+	×	1	—	×	1
+	XY	↑	$1/x$	e^x	÷	2	×	π	÷
$\sqrt{}$	×	↑	↑	1	XY	—	С/П	БП	00

Инструкция: $x = PX$ (В/О) С/П $PX = \Phi(x)$, $PY = 1 - \Phi(x)$. При $x \geqslant 5{,}5$ значения $\Phi(x)$ по этой программе округляются до единицы и о величине $\Phi(x)$ лучше судить по содержимому регистра Y.

По этой программе получим $\Phi(0{,}1) = 0{,}079655672$ (время счета около 23 с), $\Phi(2{,}9) = 0{,}99626842$ (время счета около 2 мин), $\Phi(3) = 0{,}9973001$ (время счета около 17 с), $\Phi(5{,}5) = 1$ (время счета около 17 с), $1 - \Phi(x) = 3{,}7977891 \cdot 10^{-8}$. Сравнение с табличными данными (данные справочника [6] следует удвоить) свидетельствует о высокой точности вычислений.

Достаточно широко в инженерных приложениях используется гамма-функция, определяемая для $\mathrm{Re}\,x > 0$ интегралом

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1}\, dt$$

или, для любых вещественных и комплексных значений аргумента, бесконечным произведением

$$\Gamma(x) = \lim_{n\to\infty} \frac{n!\, n^{x-1}}{x(x+1)(x+2)\ldots(x+n-1)},$$

причем $\Gamma(x+1) = x\,\Gamma(x)$, $\Gamma(n) = (n-1)!$ при целом положительном n, $\Gamma(x)\Gamma(1-x) = \pi/\sin \pi x$.

Для вычисления гамма-функции положительного вещественного аргумента удобно использовать формулу Стирлинга

$$\Gamma(x) = \sqrt{2\pi/x}\, e^{-x} x^x H(x),$$

где $H(x) = 1 + 1/12x + 1/288x^2 - 139/51840x^3 - 571/2488320x^4 + \ldots$ является разложением в степенной ряд функции e^γ, $\gamma = B_1/1 \cdot 2x - B_2/(3 \cdot 4)x^3 + B_3/(5 \cdot 6)x^5 - B_4/(7 \cdot 8)x^7 + \ldots$ и числа Бернулли $B_1 = 1/6$, $B_2 = 1/30$, $B_3 = 1/42$, $B_4 = 1/30$, ...

Для программной реализации вычисления гамма-функции целесообразно выполнить 9-кратное приведение по формуле $\Gamma(x+1) = x\,\Gamma(x)$ с последующим использованием формулы Стирлинга, в которой достаточно ограничиться первым членом ряда $H(x)$. Полученная таким образом формула

$$\Gamma(x) = \frac{\sqrt{2\pi/(x+10)}\, e^{(x+10)(\ln(x+10)-1)+1/12(x+10)}}{x(x+1)(x+2)\ldots(x+9)}$$

реализуется следующей программой для вещественного аргумента.

Программа 83/34. Вычисление гамма-функции вещественного аргумента

П1	9	П0	ИП1	ИП1	1	+	×	Вх	L0
05	1	+	П1	↑	ln	1	—	×	ИП1
1	2	×	1/x	+	e^x	XY	÷	2	π
×	ИП1	÷	√	×	С/П	БП	00		

Инструкция: $x = PX$ (В/О) С/П $PX = \Gamma(x)$.

По этой программе время вычисления $\Gamma(0{,}5) = 1{,}7724568$ (точное значение 1,7724538) составляет около 25 с. Программа обеспечивает вычисление результата не менее чем с пятью верными цифрами и для устранения сомнительных цифр может быть дополнена (как и другие программы с такой же точностью результата) фрагментом ↑ ВП 3 + Вх — ..., записываемым перед оператором С/П. В этом случае будут высвечиваться только пять верных цифр результата.

Многие инженерные задачи связаны с использованием цилиндрических функций, например, функций Бесселя первого и второго рода

порядка n, представляющих частные решения дифференциального уравнения

$$d^2y/dx^2 + (1/x)dy/dx + (1 - n^2)/x^2)y = 0.$$

Чаще всего приходится вычислять функции Бесселя первого рода порядка n, аппроксимируемые степенным рядом

$$J_n(x) = \sum_{k=0}^{\infty} (-1)^k (x/2)^{n+2k}/k!\ \Gamma(n + k + 1)$$

или, при целых значениях n,

$$J_n(x) = (1/n!)(x/2)^n (1 - (x/2)^2/1!(n + 1) + \\ + (x/2)^4/2!(n + 1)(n + 2) - \ldots .$$

При вычислении функций Бесселя высших порядков удобно использовать рекуррентное соотношение

$$J_{n+1}(x) = (2n/x) J_n(x) - J_{n-1}(x).$$

Для вычисления функций Бесселя первого рода целого порядка можно воспользоваться рекуррентными соотношениями

$$J_n(x) = \begin{cases} (2/\pi) \int_0^{\pi/2} \cos(x \sin\varphi) \cos n\varphi\, d\varphi & \text{для четных } n, \\ (2/\pi) \int_0^{\pi/2} \sin(x \sin\varphi) \sin n\varphi\, d\varphi & \text{для нечетных } n, \end{cases}$$

которые преобразуемы в расчетные соотношения для программирования

$$J_n(x) = \begin{cases} (1/m) \sum_{k=0}^{m-1} \cos n\theta_k \cos(x \sin\theta_k) & \text{для четных } n; \\ (1/m) \sum_{k=0}^{m-1} \sin n\theta_k \sin(x \sin\theta_k) & \text{для нечетных } n, \end{cases}$$

где $\theta_k = \pi(k - 1/4)/m$; m — число членов ряда.

Программа 84/34. Вычисление функций Бесселя первого рода целочисленного порядка

ИП2	П4	ИП3	—	$x < 0$	08	ИП3	П4	КИП4	ИП2
π	×	cos	П5	Cx	П8	ИП4	1	—	П7
4	$1/x$	—	ИП4	÷	π	×	П6	sin	ИП3
×	ИП6	ИП2	×	ИП5	$x \geqslant 0$	43	→	cos	XY
cos	БП	47	→	sin	XY	sin	×	ИП8	+
П8	ИП7	$x = 0$	17	ИП8	ИП4	÷	С/П	БП	00

Инструкция: установить переключатель Р—Г в положение Р, $n = P2$, $x = P3$ (В/О) С/П $PX = J_n(x)$.

В этой программе автоматически выбирается большее из чисел n и x, округляется до целого числа и принимается в качестве большего

на единицу числа k, что обеспечивает точность вычислений около пяти цифр. Время счета зависит от порядка и значения аргумента. Например, время вычисления $J_2(5) = 0{,}046565068$ составляет около 80 с.

Для составления программы вычисления функций Бесселя мнимого аргумента (модифицированных функций Бесселя) следует воспользоваться разложением в степенной ряд

$$I_n(x) = (1/n!)(x/2)^n(1 + (x/2)^2/1!(n+1) + \\ + (x/2)^4/2!(n+1)(n+2) + \ldots),$$

для малых значений аргумента и асимптотическим представлением

$$I_n(x) \sim (e^x/\sqrt{2\pi x})(1 - (4n^2-1)/1!8x + \\ + (4n^2-1)(4n^2-3^2)/2!(8x)^2 - \ldots$$

для больших значений аргумента.

При составлении программ для вычисления многих других специальных функций целесообразно использовать соотношения, следующие из определения этих функций. Примером могут служить приведенные в приложении программы для вычисления коэффициентов ортогональных многочленов.

Таким образом, пользователь программируемого микрокалькулятора с библиотекой программ для вычисления специальных функций освобождается от необходимости обращения к табличным моделям этих функций и их интерполирования.

Глава 7

СТАТИСТИЧЕСКИЕ РАСЧЕТЫ

1. ОЦЕНКА ХАРАКТЕРИСТИК ОДНОМЕРНЫХ СЛУЧАЙНЫХ ВЕЛИЧИН

Простейшая задача статистической обработки одномерных массивов экспериментально полученных случайных значений x_i физической величины заключается в оценке среднего значения выборки из n случайных чисел

$$\tilde{m}_1 = (1/n)\sum_{i=1}^{n} x_i \qquad (7.1)$$

и дисперсии выборки

$$\tilde{\sigma}^2 = (1/(n-1))\sum_{i=1}^{n-1}(x_i - \tilde{m}_1)^{2*}. \qquad (7.2)$$

* Формула (7.2) справедлива для наиболее часто встречающегося нормального распределения случайных чисел с неизвестным средним. При других законах распределения элементов обрабатываемого массива эта оценка может оказаться смещенной.

При использовании микрокалькуляторов, емкость памяти которых недостаточна для хранения всего обрабатываемого массива, вычисления по этим формулам приходится совмещать с вводом исходных данных. Непосредственные вычисления по формуле (7.2) связаны с необходимостью дважды вводить числа исследуемого массива для вычисления среднего и дисперсии. Повторного ввода массива можно избежать, заменив формулу (7.2) выражением

$$\tilde{\sigma}^2 = (1/(n-1))\left(\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2/n\right). \qquad (7.3)$$

В этом случае, организовав при вводе элементов x_i выборки вычисление текущих сумм $\sum x_i$ и $\sum x_i^2$, после окончания вычислений

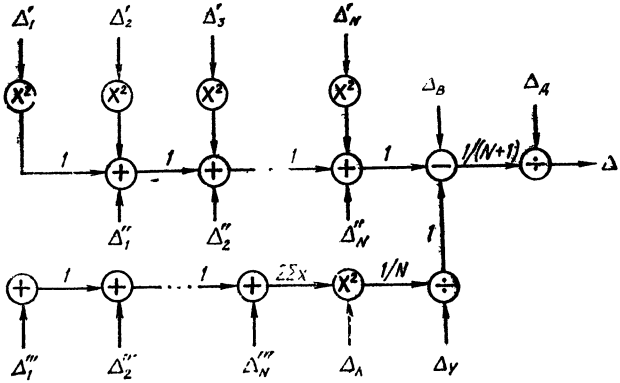


Рис. 37

можно получить оценки среднего и дисперсии одновременно [11, 13]. Однако при этом возможна большая операционная погрешность результата, возрастающая при уменьшении отношения $\tilde{\sigma}^2/\tilde{m}_1$. Для оценки этой ошибки составим граф накопления абсолютных погрешностей при вычислениях по формуле (7.3), согласно которому (рис. 37) операционная составляющая погрешности оценки дисперсии

$$\Delta\sigma^2 = \Delta_д + (\Delta + \Delta_у + \sum_{i=1}^{n} \Delta_i' + \sum_{i=1}^{n-1} \Delta_i'' + (\Delta_к + 2\left(\sum_{i=1}^{n} x_i\right) \times$$
$$\times \sum_{i=1}^{n} \Delta_i'''))/n/(n-1). \qquad (7.4)$$

Воспользовавшись для оценки абсолютных погрешностей результатов операций соотношением (2.8), при разрядности $r = 8$ получим $\Delta_д = \tilde{\sigma}^2 10^{-7}$; $\Delta_в = (n-1)\tilde{\sigma}^2 10^{-7}$; $\Delta_у = n\tilde{m}_1^2 10^{-7}$; $\Delta_к = \tilde{m}_1^2 10^{-7}$; $\Delta_i' = (x_i^2/(n+1))\,10^{-7}$; $\sum \Delta_i' = (n\tilde{m}_1 + (n-1)\tilde{\sigma}^2)\,10^{-7}$. Погрешности операций суммирования оценим согласно формуле (2.12а) величинами $\sum \Delta_i'' = (n\tilde{m}_i^2 + (n-1)\tilde{\sigma}^2)\,(n10^{-7}/3{,}6)$, $\sum \Delta_i''' = 10^{-7}n/3{,}6$.

Подставив эти оценки в формулу (7.4) и приняв $n \gg 1$, получим оценку относительной погрешности вычисления дисперсии

$$\delta\tilde{\sigma}^2 = \Delta/\tilde{\sigma}^2 \approx 10^{-7}(3 + 2\tilde{m}_1^2/\tilde{\sigma}^2 + \tilde{m}_1^2/n^2\tilde{\sigma}^2 + \\ + n(3\tilde{m}_1^2/\tilde{\sigma}^2 + 1)/3{,}6) \approx (3\tilde{m}_1^2/\tilde{\sigma}^2 + 1)\, n10^{-7}/3{,}6.$$

При $\tilde{m}_1^2/\tilde{\sigma}^2 = 10^3$ и $n = 20$ по этой оценке относительная погрешность вычисления дисперсии $\delta\tilde{\sigma}^2 \approx 1{,}67$ и неверной может оказаться даже первая цифра результата. Погрешность оценки дисперсии существенно уменьшается при вычитании из вводимых значений x_i числа x_0, близкого к искомому среднему. В этом случае

$$\tilde{m}_1 = (1/n)\sum_{i=1}^{n}(x_i - x_0) + x_0; \quad \tilde{\sigma}^2 = (1/(n-1))\Big(\sum_{i=1}^{n}(x_i - x_0) - \\ - \Big(\sum_{i=1}^{n}(x_i - x_0)^2\Big)/n\Big)$$

с относительной погрешностью

$$\delta\tilde{\sigma}^2 \approx (3(\tilde{m}_1 - x_0)^2/\tilde{\sigma}^2 + 1)\, n10^{-7}/3{,}6.$$

Этот прием особенно удобен, когда повторяется содержимое нескольких первых разрядов десятичных представлений чисел x_i. Выбрав соответствующее значение x_0, можно ограничиться вводом содержимого $x_i - x_0$ только «флюктуирующих» разрядов, что сократит время ввода и обработки исследуемого массива данных.

Например, для оценки характеристик массива (с объемом выборки $n = 10$) $x_i = 1235{,}44$; 1232,38; 1233,55; 1231,36; 1237,54; 1233,79; 1235,54; 1236,18; 1232,95; 1231,48 следует принять $x_0 = 1230$ и обрабатывать массив $x_i - x_0 = 5{,}44$; 2,38; 3,55; 1,36; 7,54; 3,79; 5,54; 6,18; 2,95; 1,48, содержащий вместе с x_0 только 34 цифры вместо 60 для исходного массива. По преобразованному массиву получим

$$\tilde{m}_1 = (1/10)\sum_{i=1}^{10}(x_i - 1230) + 1230 = 1234{,}021;$$

$$\tilde{\sigma}^2 = (1/9)\sum_{i=1}^{10}(x_i - 1230)^2 - 40{,}21^2/10 = 4{,}335365$$

с относительной погрешностью оценки дисперсии $\delta\tilde{\sigma}^2 = (3 \times \\ \times 4{,}021^2/4{,}335365 + 1)\, 10^{-6}/3{,}6 \approx 1{,}3 \cdot 10^{-6}$ вместо $\delta\tilde{\sigma}^2 = (3 \times \\ \times 1234{,}021^2/4{,}335365 + 1)\, 10^{-6}/3{,}6 \approx 0{,}293$ при непосредственной обработке исходного массива*.

Подобный прием обеспечивает также уменьшение погрешности оценки среднего $\delta\tilde{m}_1 = (n10^{-7}/3{,}6)(\tilde{m}_1 - x_0) + 2 \cdot 10^{-7}$ вместо $\delta\tilde{m}_1 \approx \\ \approx n10^{-7}/3{,}6$ при непосредственной обработке исходного массива. Для рассматриваемого массива получим $2 \cdot 10^{-7}$ вместо $3 \cdot 10^{-7}$.

* В рассматриваемом случае $\tilde{\sigma}^2 = (15228114 - 1{,}5228078 \cdot 10^3/10)/9 = 4$ с относительной погрешностью $\delta\tilde{\sigma}^2 \approx 0{,}08$ относительно оценки дисперсии для преобразованного массива.

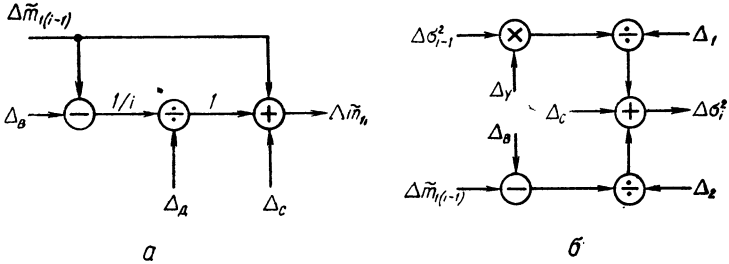
Если этот прием применить не удается, то для уменьшения погрешности оценок следует вычислять текущие оценки среднего и дисперсии после ввода очередного значения x_i. Для этого следует принять $\tilde{m}_{10} = \tilde{\sigma}_0^2 = 0$ и при $i \geqslant 1$ вычислять текущие оценки по формулам

$$\tilde{m}_{1i} = \tilde{m}_{1(i-1)} + (x_i - m_{1(i-1)})/i; \qquad (7.5)$$

$$\tilde{\sigma}_i^2 = (i-2)\tilde{\sigma}_{i-1}^2/(i-1) + (x_i - \tilde{m}_{1(i-1)})^2/i. \qquad (7.6)$$

Мажоритарная оценка операционной погрешности $\tilde{m}_{1i}$ в этом случае согласно графу накопления погрешностей (рис. 38, *а*) составляет

$$\Delta\tilde{m}_{1i} = \Delta_c + \Delta_д + \Delta\tilde{m}_{1(i-1)} + (\Delta_в + \Delta\tilde{m}_{1(i-1)})/i.$$



а *б*

Рис. 38

Приняв $\Delta_c = \Delta_д = \tilde{m}_{1i}10^{-7}$ и при $i \gg 1$ полагая $\Delta\tilde{m}_{1i} \approx \Delta_c + \Delta_д + \Delta\tilde{m}_{1(i-1)}$, получим, что обработка очередного значения x_i приводит к приращению абсолютной погрешности среднего на величину $\Delta\tilde{m}_{1i} - \Delta\tilde{m}_{1(i-1)} \approx 2 \cdot 10^{-7}\tilde{m}_{1i}$ и, следовательно, $\Delta\tilde{m}_{1i} \approx 2i\ 10^{-7}\tilde{m}_{1i}$ или $\delta\tilde{m}_{1i} \approx 2i\ 10^{-7}$.

По графу накопления абсолютных погрешностей вычисления дисперсии (рис. 38, *б*) находим

$$\Delta\sigma_i^2 = \Delta_c + \Delta_1 + \Delta_2 + (\Delta_в + 2(\tilde{m}_{i-1} - x_i)(\Delta_в + \Delta\tilde{m}_{1(i-1)}/(i-1)) + (\Delta_y + (i-2)\Delta\tilde{b}_{i-1}^2)/(i-1),$$

откуда при $i \gg 1$ на каждом цикле обработки получаем приращение $\Delta\tilde{\sigma}_i^2 - \Delta\tilde{\sigma}_{i-1}^2 \approx 3 \cdot 10^{-7}\tilde{\sigma}_i^2$. Следовательно, относительная погрешность $\delta\tilde{\sigma}_i^2 \approx 10^{-7}i$ оказывается практически независимой от отношения среднего и дисперсии.

Вычисления по формулам (7.5) и (7.6) легко реализовать на входном языке программируемого микрокалькулятора любого типа.

Программа 85/34. Вычисление текущих оценок среднего $\tilde{m}_{1i}$ и дисперсии $\tilde{\sigma}_i^2$ выборки случайных чисел

П7	x^2	П8	1	П4	ИП7	С/П	ИП7	—	П5
КИП4	→	ИП4	÷	ИП7	+	П7	ИП5	x^2	ИП4
÷	ИП4	2	—	ИП8	×	ИП4	1	—	÷
+	П8	ИП7	БП	06					

Инструкция: $x_i = PX$ (В/О) С/П $PX = \tilde{m}_{1i}$, $PY = \tilde{\sigma}_i^2$, $P4 = i$.

Проверку программы обеспечивает вычисление оценок приведенного выше массива с объемом выборки $n = 10$.

Вычисляемые оценки среднего и дисперсии также являются случайными величинами и поэтому статистическую обработку часто дополняют анализом возможных погрешностей этих оценок. С этой целью по закону распределения анализируемой случайной величины, размеру обрабатываемой выборки и требуемой достоверности результатов определяют доверительный интервал или область, в которых с заданной (доверительной) вероятностью $P_{\text{дов}}$ находится истинное значение оцениваемого параметра.

Чаще всего возникает необходимость в определении доверительного интервала оценки среднего значения, для чего находят *квантиль* $t\,(n,\ P_{\text{дов}})$ распределения выборки*, согласно которому доверительный интервал

$$(\tilde{m}_1 - t\sigma/\sqrt{n}) < m_1 < (\tilde{m}_1 + t\sigma/\sqrt{n}). \qquad (7.7)$$

Для оценки среднего значения нормально распределенной случайной величины, дисперсия которой известна, используют квантили $t\,(p)$ нормального распределения. Если дисперсия неизвестна, то в формулу (7.7) вместо значения σ подставляют оценку $\tilde{\sigma}$, а в качестве коэффициентов используют квантили нормального распределения Стьюдента [6].

В случае, когда при определении квантилей $t\,(p)$ нормального распределения допустима относительная погрешность до 10 %, то вместо решения уравнения с интегралом вероятности

$$P = \sqrt{2/\pi} \int_0^{t(p)} e^{-x^2/2}\,dx$$

относительно искомого верхнего предела интегрирования или обращения к таблицам квантилей целесообразно воспользоваться аппроксимацией

$$t\,(p) = \tau\,(a + b\tau)/(1 + c\tau),$$

где $\tau = \sqrt{\ln (2/(1-P))^2} - \sqrt{\ln 4}$, $a = 1{,}47$; $b = 0{,}437$; $c = 0{,}482$.

При программировании аппроксимации квантилей по этой формуле вычисления целесообразно реализовать двумя этапами, вначале вычисляя τ, а затем $t\,(p)$.

---

* Квантилем t называют значение аргумента x функции распределения, которому с заданной вероятностью P соответствует выполнение условия $x < t$, т. е. выполняется равенство $\int_0^t p\,(x)dx = P$, где $p\,(x)$ — функция плотности вероятности (при симметричных распределениях часто нижний предел интегрирования принимают равным нулю, а значение интеграла удваивают при двусторонних оценках).

Программа 86/34. Вычисление квантилей нормального распределения $t(p)$ с погрешностью менее 10 %

↑ 1 — 2 XY ÷ x^2 ln $\sqrt{}$ 4
ln $\sqrt{}$ — П8 ИП3 × ИП2 + ИП4 ИП8
× 1 + ÷ ИП8 × С/П БП 00

Инструкция: 1,47 = *P2*; 0,437 = *P3*; 0,482 = *P4*; *P* = *PX* (В/О) С/П *PX* = $t(p)$.

При $P = 0{,}5$ по этой программе получим $t(p) = 0{,}6646245 \approx$ $\approx 0{,}66$ (время счета около 12 с).

Можно подобрать аппроксимирующее выражение и для вычисления квантилей распределения Стьюдента по двухпараметрической формуле, но в этом случае потребуется около девяти коэффициентов или сложный алгоритм вычислений. Поэтому целесообразно воспользоваться комбинированным методом, при котором аппроксимируется только зависимость $t(n)$, где n — число степеней свободы распределения Стьюдента, равное n — 1 при вычислении среднего значения выборки из n чисел. Достаточную точность (не менее трех верных цифр при $n \geqslant 7$) обеспечивает простая аппроксимирующая формула

$$t(n, P) = t(\infty, P)/\sqrt{1 - \alpha/n + \beta/n^2}, \qquad (7.8)$$

где $t(n, P)$ — квантиль нормального распределения, а коэффициенты α и β выбираются по заданному значению P.

Этот метод обеспечивает вычисление квантилей $t(n, P)$ для всех практически употребляемых значений $P = P_{\text{дов}}$. Значения всех трех параметров аппроксимирующей формулы (7.8) для значений вероятности P, обычно употребляемых в качестве доверительных, даны в табл. 16. Для промежуточных значений вероятности P эти параметры несложно найти интерполированием.

16. Коэффициенты формулы (7.8) для аппроксимации квантилей распределения Стьюдента

P	0,8	0,9	0,95	0,98	0,99	0,999
$t(\infty, P)$	1,282	1,645	1,960	2,326	2,576	3,291
α	1,282	1,851	2,387	3,212	3,779	5,933
β	0,134	0,928	1,260	2,935	3,867	10,81

Во многих практических задачах обработку данных целесообразно продолжать лишь до тех пор, пока не достигнута требуемая точность оценки параметра. В этом случае программу вычисления текущих оценок $\tilde{m}_{1i}$ и $\tilde{\sigma}_i^2$ (программа 85/34) целесообразно, как в следующей программе, дополнить блоком определения ширины доверительного интервала и ее сравнения с заданной для выбранной доверительной вероятности $P_{\text{дов}}$.

Программа 87/34. Вычисление оценок среднего значения и дисперсии случайной величины для заданного доверительного интервала среднего

П0	1	П1	ИП0	С/П	ИП0	—	↑	ИП1	П3
1	+	П1	÷	×	Вх	XY	ИП1	2	—
ИП3	÷	ИПД	×	+	ПД	XY	ИП0	+	П0
XY	ИПВ	ИП3	÷	ИПА	—	ИП3	+	÷	$x \geqslant 0$
03	$\sqrt{}$	ИПС	×	П9	ИП8	—	$x < 0$	03	Сх
БП	04								

Инструкция: по выбранному значению доверительной вероятности из табл. 16 определить и ввести $\alpha = PA$, $\beta = PB$, $t(\infty, P) = PC$, требуемую относительную погрешность $\delta = \Delta\tilde{m}_1/\tilde{m}_1 = P8$, $x_1^2 = PД$, $x_1 = PX$ В/О С/П $PX = x_1$, $x_2 = PX$ С/П $PX = \tilde{m}_{12}$, $x_3 = PX$ С/П $PX = \tilde{m}_{13}$, ..., $x_{i+1} = PX$ С/П $PX = 0$, $P0 = m_{1i}$, РД $= \tilde{\sigma}_i^2$, $P9 = t\tilde{\sigma}_i/\tilde{m}_{1i}\sqrt{n}$ — относительную ширину достигнутого доверительного интервала.

Первые 30 операторов этой программы предназначены для вычисления текущих оценок $\tilde{m}_{1i}$ и $\tilde{\sigma}_i^2$, следующий блок обеспечивает вычисление относительной ширины доверительного интервала по формуле (7.7), представленной выражением

$$|\tilde{m}_1 - m_1|/\tilde{m}_1 \leqslant t\tilde{\sigma}/\tilde{m}_1\sqrt{n} = (t_\infty/\tilde{m}_1)(\tilde{\sigma}^2/(n - \alpha + \beta/n))^{1/2}.$$

Перед оператором извлечения корня в программе предусмотрена проверка выполнения условия $x \geqslant 0$ для устранения переполнения при отрицательном значении подкоренного выражения, которое может появиться при $n < 3$, когда погрешность аппроксимации еще велика. Следующий условный оператор сравнивает достигнутую относительную ширину доверительного интервала с допустимой и при достижении требуемой точности оценивания содержимое регистра X стирается.

Таким образом, высвечивание нуля после обработки очередного отсчета означает, что требуемая точность достигнута. При обработке массивов с нулевым средним значением такой способ индикации достигнутой точности может привести к недоразумениям при высвечивании $m_{1i} = 0$. В этих случаях, встречающихся сравнительно редко, операторы программы $x < 0$ 03 Сх БП 04 достаточно заменить операторами $\sqrt{}$ БП 03 и тогда о достигнутой точности будет свидетельствовать высвечивание на индикаторе символа ЕГГОГ.

Пусть требуется определить $\tilde{m}_1$ с предельной погрешностью 1 % при доверительной вероятности 0,95. Согласно табл. 16 введем 0,01 = $= P8$; 2,387 = PA; 1,26 = PB; 1,96 = PC. Обработав массив 1,05; 1,08; 1,03; 1,06; 0,98; 1,01; 1,03; 0,99; 1,03; 1,02; 1,05; 1,02; 1,04; 1,03; 1,06; 1,02; 1,02; 1,01; 1,04; 1,03; 1,04; 1,02; 1,05, получим на индикаторе нуль. Следовательно, требуемая точность достигнута, и по содержанию регистров *0*, *Д* и *9* находим $\tilde{m}_1 = 1{,}0308696$; $\tilde{\sigma}^2 = 5{,}0830028 \times$ $\times 10^{-4}$; $\Delta\tilde{m}_1/m_1 \leqslant 0{,}00968$.

При построении доверительного интервала оценки дисперсии рассматривают случайную величину $S^2 = (n-1)\tilde{\sigma}^2/\sigma^2$, характеризуемую при нормальном распределении выборки «распределением χ^2» с $n-1$ степенями свободы

$$\tilde{\sigma}^2(n-1)/\chi_1^2 < \sigma^2 < \tilde{\sigma}^2(n-1)/\chi_2^2,$$

где χ_1^2, χ_2^2 — квантили χ^2-распределения, соответствующие значениям $\alpha = P_{\text{дов}}/2$ и $\beta = (1-P_{\text{дов}})/2$ и определяемые из соотношений

$$\int_0^{\chi_1^2} p(u)\,du = \alpha, \quad \int_0^{\chi_2^2} p(u)\,du = \beta,$$

в которых $p(u) = u^{(n-3)/2}/(e^{u/3}2^{(n-1)/2}\Gamma((n-1)/2))$ называют χ^2-распределением.

Получающийся несимметричный доверительный интервал иногда заменяют симметричным*, а вместо дисперсии рассматривают среднеквадратичное отклонение $\tilde{\sigma}(1-q) < \sigma < \tilde{\sigma}(1+q)$, определяя коэффициенты по таблицам. Для автоматизации вычислений в этом случае можно воспользоваться аппроксимацией вида $q(n) = 1/(a + b/n + cn)^{1/2}$, где коэффициенты зависят от $P_{\text{дов}}$ (например, при $P_{\text{дов}} = 0{,}95$ и $P_{\text{дов}} = 0{,}99$ соответственно $a = -2{,}4$; $b = -3{,}2$; $c = 0{,}518$ и $a = -3{,}761$; $b = 22{,}78$; $c = 0{,}293$). Результаты вычислений при такой аппроксимации для $n \geqslant 19$ отличаются от табличных значений менее чем на $2 \cdot 10^{-3}$. Программа вычисления доверительного интервала при использовании подобной аппроксимации аналогична программе 86/34.

Иногда возникает необходимость в оценке центральных моментов более высокого порядка. Текущие значения моментов третьего и четвертого порядков можно вычислить по программам [11, 13], реализующим формулы

$$\tilde{M}_{3i} = (i-1)(\tilde{M}_{3(i-1)}/i + (i+1)\xi^3) - 3\xi\tilde{M}_{2i};$$
$$\tilde{M}_{4i} = (i-1)(\tilde{M}_{4(i-1)}/i + (i^2+i+1)\xi^4) - 4\xi(\tilde{M}_{3i} + 1{,}5\xi\tilde{M}_{2i}),$$

где $\xi = (x_i - \tilde{m}_{1(i-1)})/i$.

2. ДИСПЕРСИОННЫЙ И РЕГРЕССИОННЫЙ АНАЛИЗЫ

Во многих инженерных задачах возникает необходимость в проверке гипотезы о равенстве средних значений выборок, полученных в различных условиях. В таких случаях обычно прибегают к *дисперсионному* анализу. При однофакторном дисперсионном анализе весь статистический материал рассматривают как одну выборку объемом n, разби-

---

* Симметричный интервал всегда шире несимметричного, а при близости $P_{\text{дов}}$ к единице такая замена может оказаться невозможной. Однако при умеренных значениях $P_{\text{дов}}$ и $n \gg 1$ различие между симметричным и несимметричным интервалами незначительно.

тую на t групп объемом n_k каждая по некоторому критерию (например, при анализе погрешностей измерений, выполненных различными методами). Для каждой из групп вычисляют среднее $\bar{x}_k = \sum_{i=1}^{n_k} x_i/n_k = S_k/n_k$ и сумму квадратов отклонений $Q_k = \sum_{i=1}^{n_k} (x_i - x_k)^2$ и для совокупности всех групп находят

$$\bar{x} = \sum_{k=1}^{t} n_k x_k/n, \quad Q_A = \sum_{k=1}^{t} n_k (x_k - \bar{x})^2, \quad Q_B = \sum_{k=1}^{t} Q_k$$

и оценивают межгрупповую $\tilde{\sigma}_A^2 = Q_A/(t-1)$ и внутригрупповую $\tilde{\sigma}_B^2 = Q_B/(n-t)$ дисперсии.

Отношение полученных значений $F = \tilde{\sigma}_A^2/\tilde{\sigma}_B^2$ сравнивают с «пороговым», зависящим от величин t, n и заданного уровня значимости α, определяющего вероятность отклонения правильной гипотезы. Если полученное значение F больше порогового, то с вероятностью ошибки α средние значения групп следует признать различными и, следовательно, фактор, по которому классифицировались группы, влияющим на значение среднего m_1.

Рассмотренную процедуру называют проверкой по одностороннему F-критерию. В этом критерии порогом являются квантили F-распределения Фишера (называемого также распределением Снедекора, v^2-распределением или ω^2-распределением), используемые при проверке гипотезы о равенстве дисперсий двух выборок с равными средними. Необходимые значения $F_{1-\alpha}(f_1, f_2)$, где f_1, f_2— числа степеней свободы F-распределения (при дисперсионном анализе $f_1 = t - 1$, $f_2 = n - t$), можно найти, например, в работе [6].

При составлении программы дисперсионного анализа целесообразно преобразовать расчетные выражения так, чтобы не приходилось дважды вводить исходные данные (при вычислении суммы квадратов отклонений по разностям $x_i - \bar{x}_k$ и операционная погрешность не оказалась чрезмерной. Для этого, как и при вычислении оценок $\tilde{m}_1$ и $\tilde{\sigma}^2$, следует перейти к текущим значениям, последовательно вычисляя для каждой группы (приняв $x_0 = Q_1 = 0$) значения $\bar{x}_i = \bar{x}_{i-1} + (x_i - \bar{x}_{i-1})/i$, $Q_i = Q_{i-1} + (x_i - \bar{x}_{i-1})(i-1)/i$, $i = 1, 2, \ldots, k$ и, после ввода очередной группы ($n^0 = 0$, $\bar{x}^0 = 0$, $Q_A^0 = 0$, $Q_n^0 = 0$, где верхний индекс относится к числу обработанных групп), $n^j = n^{j-1} + k$, $\bar{x}^j = \bar{x}^{j-1} + (\bar{x}_k - \bar{x}^{j-1})/n^j$, $Q_A^j = Q_A^{j-1} + n^{j-1}(\bar{x}_k - \bar{x}^{j-1})^2/n^j$, $Q_B^j = Q_B^{j-1} + Q_k$, $j = 1, 2, \ldots, t$.

Включив в цикл «обработки» группы цикл вычислений $\tilde{\sigma}_A^{2j} = Q_A^j/(j-1)$, $\tilde{\sigma}_B^{2j} = Q_B^j/(n^j - j)$ и $F^j = \tilde{\sigma}_A^{2j}/\tilde{\sigma}_B^{2j}$, получим искомое значение $F = F^t$ после ввода и «обработки» всех групп.

Для составления программы дисперсионного анализа на входном языке ЯМК34 распределим регистры памяти следующим образом: $\bar{x}_i = P1$, $Q_i = P2$, $n^i = P6$, $\bar{x}^j = P7$, $Q_A^j = P8$, $Q_B^j = P9$, $\tilde{\sigma}_A^{2j} = PA$,

$\tilde{\sigma}_B^{2l} = PB$, $F^l = РД$, $i = P4$, $j = P5$ (с учетом увеличения на единицу содержимого регистров *4* и *5* при каждом обращении к ним с помощью операторов косвенной адресации). Для идентификации ввода первой группы данных используем очистку регистра *5* перед началом вычислений.

Примем, что при вводе первого отсчета каждой группы должно заноситься $x_1 = P1$ и $0 = P2$, а перед первым пуском программы регистры *6*, ..., *9* очищаются. В регистр *4* можно заносить число обработанных отсчетов или номер следующего отсчета и соответственно составить связанную с этим регистром часть программы. Если обработку каждой группы начинать с нажатия клавиш В/О и С/П, то регистр *5* можно очищать «вручную», а остальные регистры могут очищаться автоматически, как это предусмотрено в следующей программе.

Программа 88/34. Дисперсионный анализ

П0	ИП5	$x \neq 0$	50	ИП6	ИП1	ИП7	—	ПС	ИП4
ИП6	+	П6	÷	ИП4	×	×	Вх	ИП7	+
П7	ХУ	ИПС	×	ИП8	+	П8	ИП5	1	—
$x \neq 0$	34	÷	ПА	ИП2	ИП9	+	П9	ИП6	ИП5
—	÷	ПВ	ИПА	ХУ	÷	ПД	КИП5	БП	56
П6	П7	П8	П9	1	П5	Сх	П2	1	П4
ИП0	П1	С/П	П0	ИП1	—	ПС	x^2	ИП4	×
КИП4	ХУ	ИП4	÷	ИП2	+	П2	ИПС	ИП4	÷
ИП1	+	П1	ИП0	БП	62				

Инструкция: $0 = P5$, $x_{11} = PX$ В/О С/П $x_{21} = PX$ С/П ... $x_{k1} = PX$ С/П $x_{12} = PX$ В/О С/П $x_{22} = PX$ С/П ... $x_{k2} = PX$ / $x_{13} = PX$ В/О С/П $x_{23} = PX$ С/П ... $x_{ki} = PX$ С/П $РД = F$, $P5 = t + 1$, $P6 = n$, $P7 = \bar{x}$, $P8 = Q_A$, $P9 = Q_B$, $PA = \sigma_A^2$, $PB = \sigma_B^2$. Время обработки одного отсчета около 7с.

Контрольный пример: после обработки трех групп данных $x_{i1} = 19, 45, 26, 23, 36, 23, 26, 33, 22$; $x_{i2} = 40, 29, 26, 15, 24, 26, 36, 27, 28, 19$ и $x_{i3} = 32, 26, 30, 17, 23, 24, 29, 20$ получим $F = 0{,}3773066$, $\sigma_A^2 = 19{,}001385$; $\sigma_B^2 = 50{,}360995$.

При решении многих задач существенное значение приобретает оценка корреляционных соотношений между различными случайными процессами или различными сечениями одного и того же процесса. Коэффициент корреляции между парами значений x_i и y_i случайных величин или выборок случайного процесса

$$\tilde{r}_{xy} = \left(\sum_{i=1}^{n} (y_i - \tilde{m}_1(y))(x_i - \tilde{m}_1(x))\right)/(n-1)\tilde{\sigma}_x\tilde{\sigma}_y$$

можно вычислить по экспериментальным данным с помощью следующей программы.

Программа 89/21. Вычисление коэффициента корреляции

P2	×	ПП	ВП	F3	P4	F2	↑	ПП	ВП	F5	↑
←	+	P5	С/П	ПП	8	P6	←	ПП	8	↑	F6
×	√	P7	F3	↑	F4	×	↑	F8	÷	↑	F5
—	↑	F7	—	С/П	←	+	P3	XY	x^2	XY	←
+	←	В/О	←	x^2	↑	F8	÷	↑	←	—	В/О

Инструкция: $0 = P5 = C1 = C2 = C3 = C4 = C5 = C6$, $n = P8$, $y_i = PY$, $x_i = PX$ В/О С/П (до ввода всех n пар значений) С/П $PX = -\tilde{r}_{xy}$.

При изучении статистической взаимосвязи между случайными процессами широко используют регрессионный анализ, обеспечивающий установление функциональной связи между средними значениями исследуемых процессов. Простейшей является линейная регрессия, при которой условные средние связаны линейными уравнениями

$$m_y(x) = m_y(0) + \beta_{y/x}x, \quad m_x(y) = m_x(0) + \beta_{x/y}y.$$

Коэффициенты линейной регрессии связаны с коэффициентом корреляции соотношениями $\beta_{y/x} = r_{xy}\sigma_y/\sigma_x$ и $\beta_{x/y} = r_{xy}\sigma_x/\sigma_y$, а условные средние определяют по безусловным средним согласно формулам

$$m_y(0) = m_{1y} - \beta_{y/x}m_{1x} \text{ и } m_x(0) = m_{1x} - \beta_{x/y}m_{1y}.$$

Коэффициенты линейной регрессии, определяемые уравнением прямой, аппроксимирующей массив данных x_i, y_i, можно вычислить с помощью следующей программы.

Программа 90/21. Вычисление коэффициентов линейной регрессии

P2	F5	+	P5	F2	×	↑	F6	+	P6	F2	↑
F7	+	P7	F2	×	↑	F8	+	P8	F4	1	+
P4	С/П	F8	↑	F7	÷	↑	F5	×	P2	F4	×
↑	F7	—	P3	F2	↑	F6	—	↑	F3	÷	P2
↑	P4	×	↑	F5	XY	—	↑	F7	÷	P3	С/П

Инструкция: $0 = P4 = P5 = P6 = P7 = P8$, $y_i = PY$, $x_i = PX$ В/О С/П $PX = i$ (повторять ввод y_i и x_i до $PX = n$) С/П $PX = P3 = \beta_{y/x}$, $P2 = m_y(0)$, $P4 = n + 1$, $P5 = \Sigma y_i$, $P6 = \Sigma x_i y_i$, $P7 = \Sigma x_i$, $P8 = \Sigma x_i^2$. Для вычисления $\beta_{x/y}$ и $m_x(0)$ следует повторить вычисления, поменяв местами y_i и x_i при вводе.

На входном языке ЯМК34 можно запрограммировать одновременное вычисление коэффициента корреляции, коэффициенты регрессии и условные средние значения.

Программа 91/34. Вычисление взаимных характеристик случайных величин

ИПД	$x=0$	49	→	П2	x^2	П3	ХУ	П4	x^2
ПД	ИП2	ИП4	×	П1	ИП0	1	+	П0	С/П
↑	ИП2	+	П2	→	↑	x^2	ИП3	+	П3
→	ХУ	↑	ИП4	+	П4	→	↑	x^2	ИП5
+	П5	→	×	ИП1	+	П1	БП	15	1
3	П0	ПП	70	ИП4	ИП2	П4	ХУ	П2	ИП5
ИП3	П5	ХУ	П3	ПП	70	ИП8	ИПВ	×	С/П
ИП3	ИП4	×	ИП1	ИП2	×	—	ИПД	ИП3	×
ИП2	x^2	—	П6	÷	КП0	ИП1	ИПД	×	ИП2
ИП4	×	—	ИП6	÷	КП0	В/О			

Инструкция: $0 = РД$, $y_1 = PY$, $x_1 = PX$ В/О С/П $PX = 1$, $y_2 = PY$, $x_2 = PX$ С/П $PX = 2 \ldots y_n = PY$, $x_n = PX$ С/П $PX = n$ В/О С/П $PX = \tilde{r}_{xy}$, $PC = \tilde{m}_{1y}(0)$, $PB = \beta_{y/x}$, $PA = \tilde{m}_{1x}(0)$, $P9 = \beta_{x/y}$; значения $\tilde{m}_{1x}$ и $\tilde{m}_{1y}$ вычисляют по содержимому регистров $P2 = \Sigma x_i$, $P4 = \Sigma x_i$, $РД = n$, а значения дисперсий σ_x^2 и σ_y^2 по содержимому регистров $P3 = \Sigma y_i^2$ и $P5 = \Sigma x_i^2$.

Во многих случаях представляет интерес определение корреляционной функции для двух выборок случайного процесса, смещенных на определенное число s отсчетов. Эмпирическое значение нормированной корреляционной функции $\tilde{r}(s\Delta x)$ по выборке случайной функции $y(x)$ определяют согласно формуле

$$\tilde{r}(s\Delta x) = \frac{\sum_{i=1}^{n-s} y_i y_{i-s} - (n-s)\left(\sum_{i-1}^{n} y_i^2\right)/n^2}{\left[\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2/n\right](n-s-1)/(n-1)}.$$

Емкость памяти микрокалькуляторов с входными языками ЯМК21 и ЯМК34 обеспечивает вычисление этой корреляционной функции при смещении не более чем на пять или шесть отсчетов, причем на входном языке ЯМК21 вычисление и нормирование такой функции приходится реализовать при помощи пакета из двух программ [11]. На микрокалькуляторе с входным языком ЯМК34 эти операции выполняются с помощью одной программы.

Программа 92/34. Вычисление автокорреляционной функции

ИП7	$x=0$	63	КП0	ИП0	$x=0$	00	С/П	↑	↑
ИПД	×	ИП6	+	П6	→	ИПС	ПД	×	ИП5
+	П5	→	ИПВ	ПС	×	ИП4	+	П4	→
ИПА	ПВ	×	ИП3	+	П3	→	ИП9	ПА	×
ИП2	+	П2	→	ИП0	П9	×	ИП1	+	П1
→	x^2	ИП7	+	П7	→	ИП8	+	П8	—
П0	БП	07	ИП8	x^2	ИПА	÷	П9	—	ИПА
1	—	÷	ПД	7	П0	КИП0	ИПА	ИП0	—
ИПА	÷	ИП9	×	—	ИПД	÷	ИПА	ИП0	—
1	—	÷	С/П	БП	76				

Инструкция: $0 = P7$, $14 = O$ В/О С/П $x_1 = PX$ С/П $PX = 0$, $x_2 = PX$ С/П ... $x_n = PX$ С/П $PX = x_{n-1}$, $n = PA$ В/О С/П $PX = \tilde{r}(6\Delta x)$ С/П $PX = \tilde{r}(5\Delta x)$ С/П $PX = \tilde{r}(4\Delta x)$... С/П $PX = r(\Delta x)$, $РД = \sigma^2$, $PA = n$, $P8 = \Sigma x_i$. Значение среднего несложно вычислить по содержимому регистров *8* и *A*.

3. АППРОКСИМАЦИЯ РЕЗУЛЬТАТОВ ЭКСПЕРИМЕНТА

Численные результаты экспериментального определения значений физической величины y или зависимости $y(x)$ являются случайными в связи со случайностью изменения физических величин (при влиянии бесконечного числа сторонних воздействий) и погрешностей измерения или наблюдения величин y и x. Поэтому прежде всего необходимо установить доверительный интервал результатов эксперимента, в которых с определенной доверительной вероятностью находится истинное (среднее при малых погрешностях измерений) значение физической величины. При многократных повторениях эксперимента среднее значение и доверительный интервал для каждого численного результата находят статистическими методами. Если погрешность измерительных приборов значительно превосходит случайные изменения физических величин, то доверительные интервалы можно установить после однократного выполнения эксперимента по известным погрешностям измерительных приборов.

Обычно результаты эксперимента представляют табличными моделями $y_i = y(x_i)$, $i = 1, 2, \ldots, n$ или их графическими эквивалентами, причем переменная x_i может отображать как величину воздействия в причинно-следственном отношении (2.5), так и координату отсчета (например, время отсчета) измеряемой величины. После установления доверительных интервалов для значений y_i и x_i обычно возникает задача определения с требуемой вероятностью значений $y(x)$ между узлами табличной модели. Если средние значения x_i и y_i точно не известны, то решение этой задачи методами интерполирования не имеет смысла в связи с неопределенностью узлов интерполяции. Поэтому для приближения экспериментальных значений y_i в заданном интервале аргумента x обычно используют аналитические функции $f(x)$, значения $f(x_i)$ которых должны находиться внутри доверительных интервалов значений y_i. Задача аппроксимации аналитическими выражениями возникает и в тех случаях, когда результаты эксперимента предполагается использовать для расчета.

В качестве аппроксимирующих целесообразно использовать наиболее простые функции, значения $f(x_i)$ которых лежат в доверительных интервалах. Явно монотонные табличные зависимости $y(x_i)$ целесообразно аппроксимировать одной из функций, указанных в табл. 10, выбрав ее в соответствии с методикой, описанной в гл. 5. Выбор коэффициентов аппроксимирующей функции можно автоматизировать, воспользовавшись приведенными в гл. 5 программами, основанными на минимизации максимального отклонения аппроксимирующей функции в узлах табличной модели при соблюдении условия

$$\max |y_i - f(x_i)| \leqslant \varepsilon_i,$$

где ε_i — половина ширины симметричного доверительного интервала отсчетов y_i табличной модели.

Иногда для равномерного в среднем приближении табличной модели разбивают n ее отсчетов y_i на $m+1$ групп ($m+1 \leqslant n$) с номерами k и примерно равными суммами отсчетов каждой группы. После этого $m+1$ коэффициентов аппроксимирующей функции (например, степенного многочлена m-й степени) находят по решению системы уравнений

$$\sum_k (y_{i(k)} - f(x_{i(k)})) = 0, \quad k = 1, 2, \ldots, m+1.$$

Например, для равномерного в среднем приближения табличной модели линейной функцией $f(x) = a_0 + a_1 x$ отсчеты y_i разбивают на две группы с примерно равными суммами и находят коэффициенты a_0 и a_1 по решению системы уравнений

$$\sum_{i(1)=1}^{p} (y_{i(1)} - a_0 - a_1 x_{i(1)}) = 0;$$
$$\sum_{i(2)=1}^{n-p} (y_{i(2)} - a_0 - a_1 x_{i(2)}) = 0.$$

Для удобства решения такой системы суммы разностей предварительно заменяют разностями сумм, составляя матричное уравнение

$$\begin{bmatrix} p & \sum\limits_{i(1)=1}^{p} x_{i(1)} \\ n-p & \sum\limits_{i(2)=1}^{n-p} x_{i(2)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum\limits_{i(1)=1}^{p} y_{i(1)} \\ \sum\limits_{i(2)=1}^{n-p} y_{i(2)} \end{bmatrix}.$$

При реализации этого метода целесообразно разбить программу на три части — блок очистки используемых для вычисления регистров (например, регистры *1*, *2*, *4*, *5*, *С*, *Д*), блок формирования системы уравнений по вводимой табличной модели и блок решения этой системы относительно искомых коэффициентов. При этом в блоке формирования системы из двух уравнений необходимо предусмотреть автоматическое распределение данных с примерно равными суммами y_i.

Программа 93/34. Вычисление коэффициентов линейной функции, равномерно в среднем приближающей табличную модель

$x=0$	39	П1	П2	П4	П5	ПС	ПД	С/П	ПА
XY	ПВ	КИП4	ИПС	ИПД	—	$x \geqslant 0$	29	ИПВ	ИПД
+	ПД	ИПА	ИП2	+	П2	КИП5	БП	08	ИПВ
ИПС	+	ПС	ИПА	ИП1	+	П1	БП	08	ИП5
ИП1	×	ИП4	ИП5	—	П3	ИП2	×	—	П6
ИП5	ИПС	×	ИП3	ИПД	×	—	ИП6	÷	ИП1
ИПД	×	ИП2	ИПС	×	—	ИП6	÷	С/П	

Инструкция: $0 = PX$ В/О С/П $PX = 0$, $y_1 = PY$, $x_1 = PX$ С/П $y_2 = PY$, $x_2 = PX$ С/П ... $y_n = PY$, $x_n = PX$ С/П $1 = PX$ В/О С/П $PX = a_0$, $PY = a_1$.

Для табличной модели $y(0) = 60$, $y(2) = 80$, $y(4) = 100$, $y(6) = 115$, $y(8) = 125$, $y(10) = 135$, $y(12) = 140$, $y(14) = 145$ по этой программе (время обработки одного узла около 7 с, время вычисления коэффициентов около 13 с) получим $a_0 = 78{,}75$; $a_1 = 5$ или $f(x) = 68{,}75 + 6{,}25x$ (прямая *1* на рис. 39).

Более часто для аппроксимации табличных моделей используют метод наименьших квадратов, при котором мерой приближения табличной модели аппроксимирующей функцией $f(x)$ является сумма квадратов разностей

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

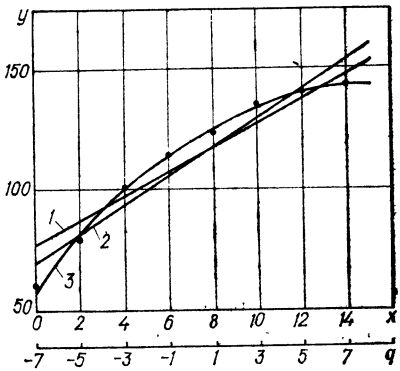


Рис. 39

Аппроксимирующую функцию в этом методе выбирают так, чтобы сумма S была минимальной, что соответствует (при попадании значений функции в доверительные интервалы) наиболее вероятным значениям аппроксимируемой физической зависимости.

Вычисление коэффициентов аппроксимирующей функции упрощается, если эта функция линейна относительно коэффициентов или

$$f(x) = f(x, a_0, a_1, \ldots, a_m) = \varphi_0(x)\, a_0 + \ldots + \varphi_m(x)\, a_m,$$

где $m < n$, а $\varphi(x)$ являются функциями только аргумента x.

Приравнивая нулю производные такой функции по коэффициентам, получают систему уравнений

$$2 \sum_{i=1}^{n} (y(x_i) - \varphi_0(x_i)\, a_0 - \ldots - \varphi_m(x_i)\, a_m)(-\varphi_r(x_i)\, a_r) = 0,$$

$$r = 1, 2, \ldots, m + 1,$$

решением которой и являются искомые коэффициенты.

Обозначив для простоты записи $\sum_{i=1}^{n} y(x_i)\varphi_j(x_i) = (y\varphi_j)$, $\sum_{i=1}^{n} \varphi_j^2(x_i) = (\varphi_j\varphi_i)$, $\sum_{i=1}^{n} \varphi_r(x_i)\varphi_j(x_i) = (\varphi_r\varphi_j)$, представим рассматриваемую систему уравнений матричным уравнением

$$\begin{bmatrix} (\varphi_0\varphi_0) & (\varphi_0\varphi_1) & \ldots & (\varphi_0\varphi_m) \\ (\varphi_1\varphi_0) & (\varphi_1\varphi_1) & \ldots & (\varphi_1\varphi_m) \\ \ldots & \ldots & \ldots & \ldots \\ (\varphi_m\varphi_0) & (\varphi_m\varphi_1) & \ldots & (\varphi_m\varphi_m) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_m \end{bmatrix} = \begin{bmatrix} (y\varphi_0) \\ (y\varphi_1) \\ \ldots \\ (y\varphi_m) \end{bmatrix}, \qquad (7.9)$$

решение которого в виде искомых значений a_r обеспечивает составление аппроксимирующей табличную модель функции по методу наименьших квадратов. Например, при выборе в качестве аппроксимирующей линейной функции $f(x) = a_0 + a_1 x$ получим матричное уравнение

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix},$$

решение которого дает коэффициенты функции.

Программную реализацию построения аппроксимирующей функции по методу наименьших квадратов также целесообразно составлять из трех блоков, подобно программе 93/34, но в этом случае целесообразно использовать симметрию квадратной матрицы относительно главной диагонали, распределив данные в памяти $x_i = PA$, $x_i = PB$, $\Sigma x_i = P1$, $\Sigma x_i^2 = P2$, $\Sigma y_i = P3$, $\Sigma x_i y_i = P4$, $n = P5$.

Программа 94/34. Вычисление коэффициентов линейной функции, аппроксимирующей табличную модель по методу наименьших квадратов,

$x=0$	32	П1	П2	П3	П4	П5	С/П	ПА	XY
ПВ	ИП3	+	П3	КИП5	ИПВ	ИПА	×	ИП4	+
П4	ИПА	ИП1	+	П1	ИПА	x^2	ИП2	+	П2
БП	07	ИП5	ИП2	×	ИП1	x^2	—	П6	ИП5
ИП4	×	ИП1	ИП3	×	—	ИП6	÷	ИП2	ИП3
×	ИП1	ИП4	×	—	ИП6	÷	С/П		

Инструкция к этой программе совпадает с инструкцией к программе 93/34. Для табличной модели, приведенной в контрольном примере к программе 93/34, по этой программе получим коэффициенты функции $f(x) = 70{,}4166666 + 6{,}0119047x$. Сравнение этой функции (прямая *2* на рис. 39) с функцией, вычисленной по предыдущей программе, достаточно наглядно свидетельствует о преимуществах метода наименьших квадратов.

При использовании метода наименьших квадратов число коэффициентов аппроксимирующей функции и, следовательно, точность аппроксимации ограничены емкостью запоминающих устройств микрокалькулятора, необходимой для решения уравнения требуемого порядка. Это ограничение удается смягчить, выбирая начало отсчета аргумента в средине его табличного интервала $x_ц = (x_п - x_0)/2$ и соответственно изменяя значения аргумента в узлах табличной модели $x_{пi} = x_i - x_0$. В этом случае в связи с парной симметрией узлов относительно начала отсчета аргумента значения $x_п = -x_п$, и матричное уравнение (7.9) вырождается в уравнение с нулевыми элементами квадратной матрицы

$$\begin{bmatrix} (\varphi_0\varphi_0) & 0 & (\varphi_0\varphi_2) & \dots \\ 0 & (\varphi_1\varphi_1) & 0 & \dots \\ (\varphi_2\varphi_0) & 0 & (\varphi_2\varphi_2) & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} a_{п0} \\ a_{п1} \\ a_{п2} \\ \dots \end{bmatrix} = \begin{bmatrix} (y\varphi_0) \\ (y\varphi_1) \\ (y\varphi_2) \\ \dots \end{bmatrix} \qquad (7.10)$$

и требования к емкости запоминающих устройств микрокалькулятора соответственно снижаются. Если аппроксимирующая функция является степенным многочленом, то вычисленные по уравнению (7.10) коэффициенты $a_{пr}$ для начала отсчета в средине интервала аргумента можно преобразовать с помощью программы 16/34 для другого выбора (например, исходного) значений узлов табличной модели.

При аппроксимации табличной модели с парной симметрией узлов линейной функцией $f(x_п) = a_{п0} + a_{п1}x_п$ уравнение (7.10) принимает вид уравнения второго порядка

$$\begin{bmatrix} n & 0 \\ 0 & \sum x^2{}_{пi} \end{bmatrix} \begin{bmatrix} a_{п0} \\ a_{п1} \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_{пi} \end{bmatrix}$$

с решением $a_{п0} = \Sigma\, y_i/n$, $a_{п1} = \Sigma\, x_{пi}y_i/\Sigma x^2_{пi}$.

Программа 95/34. Вычисление по методу наименьших квадратов коэффициентов линейной функции, аппроксимирующей табличную модель с парной симметрией узлов

$x=0$	25	П1	П2	П3	С/П	ПА	ХУ	ПВ	ИП2
+	П2	ИПВ	ИПА	×	ИП3	+	П3	ИПА	x^2
ИП1	+	П1	БП	05	ИП3	ИП1	÷	ИП2	ИП0
÷	С/П								

Инструкция: $n = P0$, $0 = PX$ В/О С/П $y_1 = PY$, $x_{п1} = PX$ С/П $y_2 = PY$, $x_{п2} = PX$ С/П ... $y_n = PY$, $x_{пn} = PX$ С/П $1 = PX$ В/О С/П $PX = a_{п0}$, $PY = a_{п1}$.

При использовании этой программы, например, для табличной модели из контрольного примера к программе 93/34 следует определить $x_ц = (14 - 0)/2 = 7$ и смещенные узлы с отсчетами $y\,(-7) = 60$, $y\,(-5) = 80$, $y\,(-3) = 100$, $y\,(-1) = 115$, $y\,(1) = 125$, $y\,(3) = 135$, $y\,(5) = 140$, $y\,(7) = 145$. Для этих данных и $n = 8$ по программе 95/34 получим коэффициенты функции $f(x_п) = 112{,}5 + 6{,}0119047x_п$, практически совпадающие (с учетом изменения шкалы аргумента) с коэффициентами функции, вычисленной по предыдущей программе (кривая *2* на рис. 39).

Коэффициенты аппроксимирующего многочлена второй степени $f(x_п) = a_{п0} + a_{п1}x_п + a_{п2}x_п^2$ при парной симметрии узлов определяются по методу наименьших квадратов из уравнения

$$\begin{bmatrix} n & 0 & \sum x^2_{пi} \\ 0 & \sum x^2_{пi} & 0 \\ \sum x^2_{пi} & 0 & \sum x^4_{пi} \end{bmatrix} \begin{bmatrix} a_{п0} \\ a_{п1} \\ a_{п2} \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_{пi} \\ \sum y_i x^2_{пi} \end{bmatrix}$$

с решением $a_{п0} = (\sum y_i \sum x^4_{пi} - \sum y_i x^2_{пi} \sum x^2_{пi}/\xi$, $a_{п1} = \sum y_i x_{пi}/\sum x^2_{пi}$, $a_{п2} = (n \sum y_i x^2_{пi} - \sum y_i \sum x^2_{пi})/\xi$, где $\xi = n \sum x^2_{пi} - (\sum x^2_{пi})^2$.

Программа 96/34. Вычисление по методу наименьших квадратов коэффициентов многочлена второй степени, аппроксимирующего табличную модель с парной симметрией узлов

$x=0$	40	П1	П2	П3	П4	П5	С/П	ПА	XY
ПВ	ИП3	+	П3	ИПА	x^2	ИП1	+	П1	ИПА
x^2	x^2	ИП2	+	П2	ИПА	ИПВ	×	ИП4	+
П4	ИПА	x^2	ИПВ	×	ИП5	+	П5	БП	07
ИП0	ИП2	×	ИП1	x^2	—	П6	ИП0	ИП5	×
ИП1	ИП3	×	—	ИП6	÷	ПД	ИП4	ИП1	÷
ИП3	ИП2	×	ИП5	ИП1	×	—	ИП6	÷	С/П

Инструкция: n = *P0*, 0 = *PX* В/О С/П y_1= *PY*, $x_{п1}$ = *PX* С/П y_2= *PY*, $x_{п2}$ = *PX* С/П ... y_n= *PY*, $x_{пn}$= *PX* С/П *1* = *PX* В/О С/П $PX = a_{п0}$, $PY = a_{п1}$, $РД = a_{п2}$.

Для табличной модели, по которой проверялась предыдущая программа, по этой программе получим коэффициенты многочлена $f(x_п) = 120{,}3125 - 6{,}0119047x_п - 0{,}3720238x_п^2$ (кривая *3* на рис. 39).

Если табулированная зависимость явно немонотонна, используют аппроксимирующие многочлены более высокой степени. Степенной многочлен третьей степени $f(x_п) = a_{п0} + a_{п1}x_п + a_{п2}x_п^2 + a_{п3}x_п^3$ определяют из уравнения

$$\begin{bmatrix} n & 0 & \sum x_{пi}^2 & 0 \\ 0 & \sum x_{пi}^2 & 0 & \sum x_{пi}^4 \\ \sum x_{пi}^2 & 0 & \sum x_{пi}^4 & 0 \\ 0 & \sum x_{пi}^4 & 0 & \sum x_{пi}^6 \end{bmatrix} \begin{bmatrix} a_{п0} \\ a_{п1} \\ a_{п2} \\ a_{п3} \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum y_i x_{пi} \\ \sum y_i x_{пi}^2 \\ \sum y_i x_{пi}^4 \end{bmatrix},$$

решение которого: $a_{п0} = (\sum y_i \sum x_{пi}^4 - \sum y_i x_{пi}^4 \sum x_{пi}^4)/\xi_1$, $a_{п1} = (\sum y_i x_{пi} \sum x_{пi}^6 - \sum y_i x_{пi}^4 \sum x_{пi}^4)/\xi_2$, $a_{п2} = (\sum y_i x_{пi} \sum x_{пi}^6 - \sum y_i \sum x_{пi}^2)/\xi_1$, $a_{п3} = (\sum y_i x_{пi}^4 \sum x_{пi}^4 - \sum y_i x_{пi} \sum x_{пi}^4)/\xi_2$, где $\xi_1 = n \sum x_{пi}^4 - (\sum x_{пi}^2)^2$, $\xi_2 = \sum x_{пi}^2 \sum x_{пi}^6 - (\sum x_{пi}^4)^2$.

Программа 97/34. Вычисление коэффициентов многочлена третьей степени, аппроксимирующего табличную модель с парной симметрией узлов по методу наименьших квадратов

ПА	ИП4	+	П4	ИП9	ИПА	×	ИП5	+	П5
ИП9	x^2	ИПА	×	ИП6	+	П6	ИП9	x^2	x^2
ИПА	×	ИП7	+	П7	ИП9	x^2	ИП1	+	П1
ИП9	x^2	x^2	ИП2	+	П2	ИП9	x^2	x^2	x^2
ИП3	+	П3	С/П	ИП0	ИП2	×	ИП1	x^2	—
П8	ИП1	ИП3	×	ИП2	x^2	—	ПВ	ИП0	ИП6
×	ИП4	ИП1	×	—	ИП8	÷	ПС	ИП7	ИП1
×	ИП5	ИП2	×	—	ИПВ	÷	ПД	ИП5	ИП3
×	ИП7	ИП2	×	—	ИПВ	÷	ИП4	ИП3	×
ИП6	ИП1	×	—	ИП8	÷	С/П			

Инструкция: $n = P0$, $0 = P1 = P2 = \ldots = P6 = P7$, $x_{п1} = P9$, $y_1 = PX$ В/О С/П $x_{п2} = P9$, $y_2 = PX$ В/О С/П ... $x_{пn} = P9$, $y_n = PX$ В/О С/П ($PX = \sum x_{пi}^6$) С/П $PX = a_{п0}$, $PY = a_{п1}$, $PC = a_{п2}$, $PД = a_{п3}$. (Время обработки одного отсчета около 15 с, время вычислений коэффициентов решением системы уравнений около 20 с).

Для табличной модели, показанной на рис. 39 при парной симметрии узлов, по этой программе получим результаты, практически совпадающие на графике со значениями табулированной функциональной зависимости в узлах.

При необходимости аппроксимации табличной модели по методу наименьших квадратов многочленами степени $m > 3$ можно составить пакет программ для раздельного формирования системы уравнений и вычисления коэффициентов по этой системе. В некоторых случаях аргументом табличной модели являются номера узлов $i = 1, 2, \ldots, n$. В этом случае можно упростить ввод данных, автоматизировав формирование узлов i с помощью оператора косвенного вызова из инкриминируемого (с увеличивающимся на единицу содержимым) регистра *4*, *5* или *6*. Например, программу 94/34 в этом случае можно модифицировать следующим образом.

Программа 98/34. Вычисление коэффициентов линейной функции, аппроксимирующей табличную модель со значениями узлов, равными их номерам

$x = 0$	31	П1	П2	П3	П4	П5	С/П	ПА	КИП5
ИП5	ИП1	+	П1	ИП5	x^2	ИП2	+	П2	ИПА
ИП3	+	П3	ИПА	ИП5	×	ИП4	+	П4	БП
07	ИП5	ИП2	×	ИП1	x^2	—	П6	ИП3	ИП2
×	ИП4	ИП1	×	—	ИП6	÷	ИП2	ИП3	×
ИП1	ИП4	×	÷	ИП6	—	С/П			

Инструкция: $0 = PX$ В/О С/П $PX = 0$, $y_1 = PX$ С/П $y_2 = PX$ С/П ... $y_n = PX$ С/П $1 = PX$ В/О С/П $PX = a_0$, $PY = a_1$.

Автоматическое вычисление значений аргумента в узлах табличной модели можно использовать и при парной или непарной симметрии узлов, если имеется резерв памяти микрокалькулятора. Например, организовав хранение шага модели в регистре *4* и формирование значений узлов в регистре *5*, программу 94/34 можно модифицировать.

Программа 99/34. Вычисление коэффициентов линейной функции, аппроксимирующей по методу наименьших квадратов табличную модель с парной симметрией узлов при автоматическом вычислении значений узлов

$x = 0$	27	П1	П2	П3	С/П	ПА	ИП2	+	П2
ИП5	ИПА	×	ИП3	+	П3	ИП5	x^2	ИП1	+
П1	ИП4	ИП5	+	П5	БП	05	ИП3	ИП1	÷
ИП2	ИП0	÷	С/П						

Инструкция: $n = P0$, $(x_1 - x_n)/2 = P5$, $h = P4$, $0 = PX$ В/О С/П $y_1 = PX$ С/П $y_2 = PX$ С/П ... $y_n = PX$ С/П $1 = PX$ В/О С/П $PX = a_0$, $PY = a_1$.

Например, для табличной модели, приведенной в контрольном примере к программе 93/34, при использовании этой программы следует выполнить $8 = P0$, $2 = P4$, $-7 = P5$, $0 = PX$ В/О С/П $60 = PX$ С/П $80 = PX$ С/П ... . Полученные результаты будут совпадать с результатами вычислений по программе 94/34, но ввод данных существенно упрощается, что значительно облегчает вычисления при больших массивах исходных данных.

При большом числе узлов и многоэкстремальной табличной модели исследуемой экспериментально функциональной зависимости аппроксимирующая функция должна быть многочленом высокой степени или другой функцией с большим числом коэффициентов. В этом случае, вычисление этой аппроксимирующей функции становится чрезмерно громоздкой. Кроме того, часто требуется лишь очистить результате эксперимента от случайных погрешностей. В таких случаях прибегают к методам последовательного сглаживания табличной модели по нескольким точкам. Такие методы относительно просто реализуются не только на ЭВМ высокой производительности, но и с помощью программируемых микрокалькуляторов.

При линейном сглаживании по трем очередным значениям x_i табличной модели с постоянным шагом используют формулу

$$\bar{x}_i = (x_{i-1} + x_i + x_{i+1})/3,$$

причем для крайних узлов таблицы используют соотношения

$$\bar{x}_1 = (5x_1 + 2x_2 - x_3)/6,\ \bar{x}_n = (5x_n + 2x_{n-1} + x_{n-2})/6.$$

Эти формулы несложно реализовать программно, но при линейном сглаживании более точные результаты можно получить по очередным пяти точкам согласно формуле

$$\bar{x}_i = (x_{i-2} + x_{i-1} + x_i + x_{i+1} + x_{i+2})/5,$$

причем четыре крайние точки сглаживаются по формулам

$$\bar{x}_1 = (3x_1 + 2x_2 + x_3 - x_4)/5,\ \bar{x}_2 = (4x_1 + 3x_2 + 2x_3 + x_4)/10,$$
$$\bar{x}_{n-1} = (x_{n-3} + 2x_{n-2} + 3x_{n-1} + 4x_n)/10,\ \bar{x}_n = (3x_n + 2x_{n-1} + x_{n-2} - x_{n-3})/5.$$

Программа 100/34. Линейное сглаживание по пяти точкам

П4	Сх	С/П	П3	Сх	С/П	П2	Сх	С/П	П1
ПП	38	ПП	55	С/П	ИП4	П5	ИП3	П4	+
ИП2	П3	+	ИП1	П2	+	XY	П1	+	5
÷	БП	14	ПП	55	ПП	38	С/П	ИП1	4
×	ИП2	3	×	+	ИП3	2	×	+	ИП4
+	1	0	÷	В/О	ИП1	3	×	ИП2	2
×	+	ИП3	+	ИП4	—	5	÷	В/О	

Инструкция: $x_1 = PX$ В/О С/П $PX = 0$, $x_2 = PX$ С/П $PX = 0$, $x_3 = PX$ С/П $PX = 0$, $x_4 = PX$ С/П $PX = \bar{x}_1$, $PY = x_2$, $x_5 = PX$ С/П $PX = \bar{x}_3$, ... $x_i = PX$ С/П $PX = \bar{x}_{i-2}$, ... $x_{n-1} = PX$ С/П

$PX = \bar{x}_{n-3}$, $x_n = PX$ С/П $PX = \bar{x}_{n-2}$ БП 33 С/П $PX = \bar{x}_{n-1}$, $PY = \bar{x}_n$.

Контрольный пример: для последовательности $x_i = 12{,}1;\ 12{,}3;\ 11{,}9;\ 12;\ 12{,}2;\ 11{,}8;\ 12{,}1;\ 11{,}7;\ 12;\ 12{,}4;\ 11{,}6;\ 12{,}1;\ 11{,}9;\ 11{,}6;\ 12$ после сглаживания по этой программе (время обработки одного отсчета в средине последовательности около 7 с) получим $\bar{x}_i = 12;\ 12{,}04;\ 12{,}1;\ 12{,}04;\ 12;\ 11{,}96;\ 11{,}96;\ 12;\ 11{,}96;\ 11{,}96;\ 12;\ 11{,}92;\ 11{,}84;\ 11{,}87;\ 11{,}8$.

Подобную программу можно составить для нелинейного сглаживания последовательности случайных чисел по семи текущим отсчетам согласно формуле

$$\bar{x}_i = (7x_i + 6(x_{i+1} + x_{i-1}) + 3(x_{i+2} + x_{i-2}) - 2(x_{i+3} + x_{i-3}))/21$$

с вычислением крайних сглаженных значений по формулам:

$$\bar{x}_1 = (39x_1 + 8x_2 - 4(x_3 + x_4 - x_5) + x_6 - 2x_7)/42;$$
$$\bar{x}_2 = (8x_1 + 19x_2 + 16x_3 + 6x_4 - 4x_5 - 7x_6 + 4x_7)/42;$$
$$\bar{x}_3 = (-4x_1 + 16x_2 + 19x_3 + 12x_4 + 2x_5 - 4x_6 + x_7)/42;$$
$$\bar{x}_{n-2} = (x_{n-6} - 4x_{n-5} + 2x_{n-4} + 12x_{n-3} + 19x_{n-2} + 16x_{n-1} - 4x_n)/42;$$
$$\bar{x}_{n-1} = (4x_{n-6} - 7x_{n-5} - 4x_{n-4} + 6x_{n-3} + 16x_{n-2} + 19x_{n-1} + 8x_n)/42;$$
$$\bar{x}_n = (-2x_{n-6} + 4x_{n-5} + x_{n-4} - 4x_{n-3} - 4x_{n-2} + 8x_{n-1} + 39x_n)/42.$$

При необходимости сглаживание можно повторять два или более раз, пока не будут устранены «шумы» в виде случайных изменений сглаживаемой величины. В некоторых случаях по табличным моделям с постоянным шагом приходится вычислять производную исследуемой зависимости, что целесообразно совместить с ее сглаживанием для получения более точного значения производной. В этих случаях вычисление производной табличной модели целесообразно выполнять по ее сглаженным отсчетам согласно формулам для пяти точек

$$x_i' = ((x_{i+1} - x_{i-1})\,2/3 - (x_{i+2} - x_{i-2})/12)/h,\ i \geqslant 2$$

или семи точек

$$x_i' = ((x_{i+1} - x_{i-1})\,58 + (x_{i+2} - x_{i-2})\,67 - (x_{i+3} - x_{i-3})\,22)/252h,$$
$$i \geqslant 3,$$

где h — шаг табличной модели.

4. МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ СОБЫТИЙ

Случайное событие при заданной вероятности его исходов удобно моделировать с помощью программы автоматических вычислений, не содержащей оператора остановки С/П и обеспечивающей хранение в регистре X символов (цифр) возможных исходов в течение отрезков времени, пропорциональных заданным вероятностям исходов. После каждого пуска такой программы нажатиям клавиш В/О и С/П и ее аварийной остановки в случайный момент времени с соответствующей вероятностью будет высвечиваться символ одного из исходов случайного события.

Событие с двумя равновероятными исходами, обозначенными символами 0 и 1, проще всего моделировать по программе XY БП 00 (на входном языке ЯМК21 адрес 00 заменяется указателем этого адреса РО) или XY XY В/О с предварительным вводом чисел 0 и 1 в регистры X и Y. Случайное событие с тремя (1, 2, 3) или четырьмя (1, 2, 3, 4) равновероятными исходами на входном языке ЯМК34 удобно моделировать по программе → БП 00 или → → В/О при предварительном занесении в регистры операционного стека соответственно чисел 0, 1, 2 и 3 (при высвечивании нуля испытание повторяют) или 1, 2, 3 и 4. По этим же программам можно моделировать и случайное событие с вероятностью p (1) = 1 — p (0) = 0; 0,25; 0,5; 0,75; 1 одного из двух исходов, занося предварительно в регистры операционного стека числа 0 и 1 в нужном соотношении.

Если число равновероятных исходов случайного события $m < 15$, то символы исходов следует занести в m регистров памяти и использовать программу с последовательным вызовом содержимого этих регистров в регистр X. Для обеспечения одинакового времени хранения в регистре X символов всех исходов перед оператором безусловного перехода к началу программы вводится оператор набора (вызова) цифр, не используемых для обозначения исходов (при высвечивании этих цифр испытание повторяют). Например, при $m = 10$ следует занести цифры 0, 1, ..., 9 в регистры с номерами 0, 1, ..., 9 (в случайном порядке) и моделировать случайное событие с 10 равновероятными исходами при помощи программы

ИПО ИП1 ИП2 ... ИП9 ИПА БП 00

с 77 = PA (при высвечивании 77 испытание повторяют).

Эту программу удобно использовать для формирования n-разрядных чисел, равномерно распределенных в интервале $(0,10^n)$ с шагом $1/10^n$ и соответствующих равновероятным 10^n— 1 исходам случайного события. В этом случае (например, при составлении таблицы случайных чисел) повторяют серию из n испытаний, рассматривая результат каждого испытания в серии как содержимое очередного разряда формируемого случайного числа. Для преобразования этих чисел в интервал (0,1) достаточно разделить их на 10^n.

Случайное событие с небольшим числом неравновероятных исходов также можно моделировать рассмотренным способом, например, заполнив программную память в нужном соотношении кодами операторов вызова из памяти или набора символов исходов. Однако реализация таких методов связана со значительными затратами времени и практически неприменима, когда требуется совместить в одной программе формирование и использование модели случайного события.

Случайные события или дискретные случайные величины моделируют на универсальных ЭВМ с помощью детерминированного (программируемого) процесса формирования последовательности некоррелированных (псевдослучайных или квазислучайных) чисел. Нарушение корреляции между числами такой последовательности обычно обеспечивают преобразованием числа z_{i-1} из интервала $(0, b)$ в больший интервал $(0, ab)$ и последующим отбрасыванием содержимого старших

разрядов для получения числа z_i в исходном интервале $(0, b)$. Если этот процесс приводит к получению всех дискретных чисел из интервала $(0, b)$, то они будут равномерно распределены в интервале. Для получения псевдослучайных чисел с другими законами распределения аналитически преобразуют псевдослучайные числа с равномерным распределением.

Рассмотренное преобразование часто реализуют по рекуррентной формуле вычетов

$$z_i = az_{i-1} \pmod b, \tag{7.11}$$

означающей, что при делении положительных целых чисел z_i и az_{i-1} на целое число b получаются одинаковые остатки.

При $a < b$ эту формулу можно представить выражением

$$z_i = az_{i-1} - bE(az_{i-1}/b), \tag{7.12}$$

где $E(\ )$ — целая часть содержимого скобок.

Вычисления по формуле (7.12) несложно автоматизировать с помощью программ на входных языках ЯМК21 [11] и ЯМК34, причем в последнем случае для выделения целой части числа целесообразно использовать оператор косвенного вызова из памяти.

Программа 101/34. Генерирование псевдослучайных чисел с равномерным распределением в интервале $(0, b)$

```
ИП7 ИПА × П8 ИПВ ÷ 1 + П9 КИП9
ИП8 ИП9 1 — ИПВ × — П7 С/П БП
00
```

Инструкция: $a = PA$, $b = PB$, $z_0 = P7$ В/О С/П $PX = z_1$ С/П $PX = z_2$ ... С/П $PX = z_n$.

Для получения по этой программе массива из $n = b - 1$ целых псевдослучайных чисел необходимо тщательно подобрать коэффициенты a и b из множества простых или нечетных (кроме кратных 5) чисел. Для небольших массивов с $n = 10; 30; 100$ псевдослучайных чисел можно принять соответственно $a = 7$, $b = 11$, $a = 17$, $b = 31$; $a = 31$, $b = 61$. При неудачном выборе коэффициентов образуются независимые циклы с периодом $s < n$, коэффициент корреляции становится равным единице, а распределение генерируемых чисел отличается от равномерного. Так, при $a = 3$, $b = 11$, $z_0 = 1$ и $z_0 = 2$ по программе 101/34 получим соответственно $z_i = 3, 9, 5, 4, 1, \ldots$ и $z_i = 6, 7, 10, 8, 2, \ldots$ с $s = 5$ и неравномерным распределением.

Трудности выбора коэффициентов возрастают при генерировании больших массивов псевдослучайных чисел, но достаточно хорошие результаты обеспечивает выбор по методу Коробова [9] коэффициента b из множества простых чисел 2027, 5087, 10079, ..., z_0 из интервала $(0, b)$ и $a = b - 3^c \approx b/2$, c — положительное целое число. При выборе $a = 1298$, $b = 2027$, $z_0 = 1013$ по программе 101/34 получим $z_i = 1378, 830, 1003, 560, 1214, 793, 1625, 1170, 437, \ldots$

Для определения периода s некоррелированной последовательности чисел, генерируемых по программе 101/34, следует перед оператором С/П ввести фрагмент ИП6 — $x \neq 0$ 24 L0, 00, занести z_0 в регистры

6 и *7*, а в регистр *0* — число *n*. В этом случае выполнение программы прекратится при генерировании числа z_0, а период *s* определится разностью числа *n* и содержимого регистра *0*. Подобная проверка показывает, что при $a = 1298$ и $b = 2027$ по программе 101/34 генерируются все $n = s = 2026$ чисел.

В инженерных задачах часто требуются псевдослучайные числа с равномерным распределением в интервале (0,1). Для приведения в этот интервал результатов выполнения программы 101/34 достаточно перед оператором С/П ввести фрагмент ИПВ ÷.

Формулу (7.12) можно упростить делением правой и левой частей на число *b*, но в этом случае возрастает вероятность возникновения замкнутых циклов с периодом $s < n$. При моделировании выборок небольшого объема линейную функцию az_{i-1} в формуле (7.12) можно заменить другой линейной или нелинейной функцией $f(x_{i-1})$, значения которой больше значений аргумента. В этом случае псевдослучайные числа с равномерным распределением в интервале (0,1) можно генерировать по рекурсивной формуле

$$x_i = f(x_{i-1}) - E(f(x_{i-1})) = Д(f(x_{i-1})), \qquad (7.13)$$

где *Д* — символ десятичной дробной части содержимого скобок.

Закон распределения псевдослучайных чисел, генерируемых по этой формуле, зависит от выбора функции $f(x_{i-1})$ и способа округления результатов операций микрокалькулятором. Для сокращения времени счета целесообразно выбирать функции, наиболее просто реализуемые на входном языке, например $11x_{i-1} + \pi$ или $e^{x_{i-1}+\pi}$ [19].

Программа 102/34. Генерирование псевдослучайных чисел с равномерным распределением в интервале (0,1)

ИП9 π + e^x П9 КИП9 ↪ ИП9 — П9
С/П БП 00

Инструкция: x_0= *P9* В/О С/П $РХ = x_1$ С/П $РХ = x_2$ ... С/П $РХ = x$ (время счета около 5 с.)

При x_0= 0,5 по этой программе получим x_i = 0,152545; 0,954155; 0,084227; 0,174197; 0,544131; ...

Подобные программы удобно использовать и в качестве фрагментов более сложных программ. Так, для моделирования случайного события с двумя исходами 1 и 0 при заданных вероятностях исходов *p* (1) и *p* (0) = 1 — *p* (1) достаточно разбить интервал (0,1) на части, пропорциональные заданным вероятностям и реализовать высвечивание символа 1 или 0 в зависимости от попадания очередного числа x_i в соответствующую часть интервала.

Программа 103/34. Моделирование случайного события с заданной вероятностью *p* (1) = 1 — *p* (0) исходов *y* = 1 и *y* = 0

ИП9 π + e^x П9 КИП9 — ИП9 — П9
ИП8 — $x<0$ 17 1 БП 18 Сх С/П БП
00

Инструкция: p (1) = *P8*, x_0= *P9* В/О С/П $PX = y_1$ С/П $PX =$ $= y_2$ ... С/П $PX = y_n$ (время счета около 7 с).

Контрольный пример: при $x_0 = p$ (1) = 0,5 получим y_i = 0 1 0 0 1 1 0 0 1 1 ...

Для моделирования события с k равновероятными исходами достаточно разбить интервал (0,1) на k равных частей и организовать высвечивание символа исхода, соответствующего попаданию псевдослучайного числа в частный интервал.

Программа 104/34. Моделирование случайного события с k равновероятными исходами

ИП9 π + e^x П9 КИП9 → ИП9 — П9
ИП8 × 1 + П7 КИП7 ИП7 С/П БП 00

Инструкция: k = *P8*, x_0= *P9* В/О С/П $PX = y_1$ С/П $PX = y_2$ С/П $PX = y_3$ ... С/П $PX = y_n$ (время счета около 7 с).

Контрольный пример: при $k = 6$, x_0= 0,5 получим y_i = 1 6 1 2 4 6 3 ...

Подобным образом моделируются случайные события с k неравновероятными исходами при разбиении интервала (0,1) на k отрезков, пропорциональных вероятностям исходов.

Псевдослучайные числа с равномерным распределением могут быть аналитически преобразованы в псевдослучайные числа с другими законами распределения. Так, для генерирования псевдослучайных чисел с нормальным распределением при нулевом среднем значении и заданной дисперсии σ^2 используют преобразование

$$q_i = r_i \sin 2\pi x_{i-1},$$

где r_i — псевдослучайные числа с распределением по закону Рэлея,

$$r_i = \sqrt{2\sigma^2 \ln (1/x_i)};$$

x_i, x_{i-1} — псевдослучайные числа с равномерным распределением в интервале (0,1).

При жестких требованиях к коэффициенту корреляции для генерирования псевдослучайных чисел следует использовать формулу (7.12) с выбором коэффициентов по методу Коробова, но в большинстве случаев достаточно хорошие результаты обеспечивает использование формулы (7.13), реализованной в следующей программе.

Программа 105/34. Генерирование псевдослучайных чисел q_i с нормальным распределением, r_i с распределением Рэлея и x_i с равномерным распределением в интервале (0,1)

ИП9 П6 π + e^x П9 КИП9 → ИП9 —
П9 1/x ln ИПД × 2 × $\sqrt{}$ П8 ИП6
2 × π × sin × С/П БП 00

Инструкция: установить переключатель Р—Г в положение Р, σ^2= *РД*, x_0= *P9* В/О С/П $PX = q_1$, *P8* = r_1, *P9* = x_1 ... С/П $PX = q_n$, *P8* = r_n, *P9* = x_n (время счета около 11 с).

Контрольный пример: при $\sigma^2 = 1$, $x_0 = 0{,}5$ получим $q_i = 0$; 0,25070047; — 0,63195364; 0,94384051; 0,90845124; ..., r_i= 1,9392244; 0,30636285; 2,2245177; 1,8695284; 1,1032364; ..., x_i= 0,152545; 0,954155; 0,084227; 0,174197; 0,544131; ...

При моделировании выборок случайных чисел небольшого объема числовые характеристики (прежде всего среднее и дисперсия) генерируемых псевдослучайных чисел могут значительно отличаться от требуемых для равномерного распределения. Между тем, во многих задачах требуются достаточно представительные выборки небольшого объема. Если требования к ранговой корреляции не слишком жесткие, то в этих случаях можно формировать псевдослучайные числа, независимо изменяя содержимое их разрядов по детерминированному закону. Для этого пригодны различные алгоритмы, один из которых реализован следующей программой:

ИПС	1	+	ПС	1	0	—	$x=0$	31	ПС
ИПА	5	+	ПА	ИПВ	1	+	ПВ	ИПД	1
+	ПД	1	0	—	$x=0$	31	ИПС	1	+
ПС	ИПА	3	+	ПА	1	0	—	$x\geqslant 0$	41
ПА	ИПВ	7	+	ПВ	1	0	—	$x\geqslant 0$	51
ПВ	КИПВ	1	0	×	КИПА	+	1	0	×
КИПС	+	С/П	БП	00					

Пользование этой программой описывается следующей инструкцией: занести в регистры *0*, ..., *9* в случайном порядке цифры от 0 до 9, ввести произвольные однозначные числа в регистры *А*, *В*, *С* и *Д*, выполнить (В/О) С/П $PX = z_i$ (время счета около 40 с).

Контрольный пример: при $8 = P0$, $2 = P1$, $5 = P2$, $7 = P3$, $6 = P4$, $1 = P5$, $3 = P6$, $0 = P7$, $9 = P8$, $4 = P9$, $1 = PA$, $2 = PB$, $3 = PC$, $4 = PД$ получим z_i= 466, 301, 783, 870, 39, 644, 508, 482, 375, 737, ...

Нетрудно проверить, что для любой последовательности из $n > 10$ чисел, генерируемых по этой программе (при полном периоде неповторяющихся чисел $n = 999$) среднее и дисперсия моделируемых выборок случайных чисел удовлетворяют равномерному распределению. Однако при относительно больших n может проявляться ранговая корреляция между генерируемыми числами. В тех случаях, когда такая корреляция недопустима, можно генерировать содержимое каждого разряда по квазислучайному закону, реализованному в ранее рассмотренных программах. Так, для генерирования целых чисел в интервале (0,1000) можно использовать следующую программу:

Сх	П7	ПП	11	ПП	11	ПП	11	С/П	БП
00	ИП9	π	+	e^x	П9	КИП9	→	ИП9	—
П9	1	0	×	1	+	П8	КИП8	ИП8	1
—	ИП7	1	0	×	П7	+	В/О		

Пользование этой программой описывается простой инструкцией: x_0= *P9* (В/О) С/П $PX = z_i$ (время счета около 30 с).

В подобных программах содержимое каждого разряда формируется по квазислучайному закону и представительные выборки моделируются лишь при достаточно большой последовательности генерируемых квазислучайных чисел. Так, при $x_0 = 0{,}5$ по этой программе получим $z_i = 190, 158, 449, 871, 591, 50, 636, 991, 806, 854, 659, 145, 395, 500, 718, \ldots$ Среднее и дисперсия генерируемых псевдослучайных чисел приближаются к требуемым для равномерного распределения ($m_1 = 500$, $\sigma^2 = 83333{,}3$) лишь при достаточно больших последовательностях — для первых 10 членов $\tilde{m}_1 = 552{,}65$; $\tilde{\sigma}^2 = 104261{,}6$, для первых 20 членов $\tilde{m}_1 = 551{,}35$; $\tilde{\sigma}^2 = 82857{,}3$.

Программная реализация моделирования случайных чисел на входном языке ЯМК21 рассмотрена в книгах [11, 19], но следует учитывать, что реализация некоторых алгоритмов для микрокалькуляторов первых выпусков с этим входным языком требует дополнительного исследования. Так, при округлении по дополнению, реализованному в таких микрокалькуляторах, выделение дробной части десятичного представления числа согласно формуле (7.13) часто удается выполнить лишь при введении поправки $4/9 = 0{,}44444444$. Для микрокалькуляторов с входным языком ЯМК21 более поздних выпусков такая поправка не нужна.

5. СТАТИСТИЧЕСКИЕ ИСПЫТАНИЯ

В инженерной практике применяются численные методы, называемые методами статистических испытаний (методами Монте-Карло) и основанные на оценке свойств моделируемого объекта при описании его параметров выборками случайных или квазислучайных чисел. Погрешности такой оценки обычно обратно пропорциональны квадратному корню из числа испытаний, равного объему выборки. Следовательно, для уменьшения погрешности в 10 раз, что соответствует одной цифре десятичного представления, число испытаний необходимо увеличить в 100 раз. Практически для получения результата с точностью 5...10 % требуется несколько сотен или тысяч, а иногда и десятков тысяч испытаний. Поэтому применение программируемых микрокалькуляторов с малыми емкостью памяти и особенно быстродействием возможно лишь для весьма грубой оценки результата статистических испытаний на простейших моделях.

Для моделирования случайной величины по ее выборке необходимо определить числовые характеристики (моменты) этой величины и закон ее распределения, а затем подобрать квазислучайное число с аналогичными свойствами. Числовые характеристики случайной величины можно оценить с помощью приведенных ранее программ, но наглядное представление о законе распределения получают с помощью различных графических моделей. Распространенная форма такого графического представления, называемая гистограммой, основана на разбиении выбранного отрезка $[x_{min}, x_{max}]$ изменения случайной величины x_i на m равных частей, называемых классовыми интервалами, и определении числа попаданий ψ_j (частот попаданий) случайной величины с объемом выборки n в каждый из таких интервалов. Гисто-

грамму строят из m прямоугольников, основания которых пропорциональны ширине классовых интервалов, а высоты — частотам попаданий.

Вычисление частот попадания ψ_j удобно программировать на входном языке ЯМК34 с использованием оператора косвенной засылки в память для распределения случайных чисел из заданного интервала по классовым интервалам. Для удобства пользования программой целесообразно автоматизировать очистку регистров памяти, предназначенных для хранения частот попадания, перед выполнением программы, а также определение классовых интервалов по заданному их числу m и граничным значениям $x_{\min}$ и $x_{\max}$ исходного интервала квазислучайных чисел. В этом случае для хранения частот попадания остается 11 свободных регистров памяти и, следовательно, можно строить гистограммы с числом классовых интервалов $m \leqslant 11$.

Программа 106/34. Вычисление частот попадания ψ_i в $m \leqslant 11$ классовых интервалов отрезка $[x_{\min}, x_{\max}]$

```
1   +   П0  Сх  КП0   —   ИП0  x=0 03   ИПД
П0  →   →   ↑   С/П   ИПС —    XY  ÷    ИП0
×   1   +   ПД  КИПД  1   +    КПД БП   11
```

Инструкция: $x_{\min} = PC$, $x_{\max} = PY = PZ$, $m = РД$ (для этого выполнить $x_{\max} = PX$ ↑ ↑ $m = PX = РД$) В/О С/П $PX = 0$, $x_1 = PX$ С/П $x_2 = PX$ С/П ... $x_n = PX$ С/П $P0 = \psi_1$, $P1 = \psi_2$ $P3 = \psi_3$, ...

Перед каждым пуском этой программы очередное значение случайного или квазислучайного числа приходится вводить в регистр X, что приводит к значительным затратам времени. Поэтому в тех случаях, когда последовательность квазислучайных чисел формируется программно, целесообразно совместить программные реализации формирования квазислучайных чисел и вычисления частот их попаданий в классовые интервалы. Такую программу удобно дополнить счетчиком для автоматического вычисления частот попадания ψ_j случайного числа с заданным объемом n выборки в m классовых интервалов (удобно принимать $m = 10$). Эту программу можно использовать и для проверки качества различных алгоритмов формирования квазислучайных чисел из интервала (0,1).

Программа 107/34. Проверка программ формирования квазислучайных чисел по частотам попадания ψ_j в 10 классовых интервалов

```
1  0  ×  1  +  ПВ  КИПВ  1  +
КПВ L0 00 С/П
```

Инструкция: заменить многоточие операторами вычисления последовательности квазислучайных чисел (с использованием регистров C и $Д$ для хранения текущих данных), $0 = P1 = P2 = \ldots = PA$, $x_0 = РД$ (если регистр $Д$ выделен для временного хранения x_i), $n = P0$ В/О С/П $P1 = \psi_1$, $P2 = \psi_2$, ..., $PA = \psi_{10}$.

В качестве примера исследуем распределение в интервале (0,1) квазилинейных чисел, формируемых по рекуррентной формуле $x_i =$

$= Д\,(10^{x_i - 1})$. Заменив многоточие в базовой программе 107/34 операторами вычисления x_i, составим следующую рабочую программу:

ИПД	10^x	↑	ПД	КИПД	→	ИПД	—	ПД	1
0	×	1	+	ПВ	КИПВ	1	+	КПВ	L0
00	С/П	1	0	П0	БП	00			

Заключительный фрагмент этой программы после оператора С/П обеспечивает автоматическое вычисление частот попаданий очередных 10 элементов формируемой последовательности при повторных пусках программы нажатием только клавиши С/П. Для другого числа элементов следует ввести это число в регистр *0* и нажать клавиши В/О и С/П.

Записывая для контроля изменения частот попаданий результаты вычислений, при $x_0 = 0{,}5$ для $n = 100$ получим частоты $\psi_j = 9, 7, 15, 10, 9, 13, 14, 13, 6, 4$, для $n = 200$ получим $\psi_j = 18, 23, 25, 19, 18, 21, 23, 25, 15, 13$, для $n = 300$ получим $\psi_j = 33, 34, 37, 30, 28, 31, 28, 39, 20, 20$, для $n = 400$ получим $\psi_j = 50, 40, 52, 39, 36, 37, 50, 30, 27$ (время обработки одного отсчета около 11 с).

По частотам попадания случайной величины в различные классовые интервалы судят о параметрах распределения этой величины. Для этого используют различные критерии, но большинство из них отличается лишь способом решения соответствующей системы уравнений.

После определения параметров распределения случайной величины полученные оценки (особенно в тех случаях, когда нет уверенности в правильном определении закона распределения) чаще всего проверяют по критерию согласия χ^2 с помощью функции (называемой статистикой)

$$y = \Big(\sum_{j=1}^{m} (\psi_j - np_j)^2\Big)/np_j = \sum_{j=1}^{m} (\psi_j^2/np_j) - n,$$

где m — число классовых интервалов, на который разбит исследуемый интервал значений, принимаемых случайной величиной; ψ_j—частота попадания в j-й классовый интервал; n — объем выборки; p_j — вероятность попадания случайной величины в j-й классовый интервал, соответствующая проверяемому закону распределения $p\,(x)$.

При бесконечном объеме выборки распределение статистики y стремится к распределению χ^2 с $r = m - s - 1$ степенями свободы, где s — число параметров распределения $p\,(x)$. Подсчитав статистику y по числу степеней свободы r и выбранному уровню значимости α, определяющему вероятность отклонения правильной гипотезы о законе распределения (ошибка первого рода по критерию Неймана—Пирсона), находят квантиль $\chi^2_{r,\alpha}$ из уравнения

$$p(\chi^2 > \chi^2_{r,\alpha}) \equiv \int_{\chi^2_{r,\alpha}}^{\infty} p(\chi^2)\,d\chi^2 = 1 - F(\chi^2_{r,\alpha}).$$

Если оказывается, что $y > \chi^2_{r,\alpha}$, то χ^2 — критерий с уровнем значимости α отрицает правильность определения параметров закона

распределения. В противном случае оценка закона распределения верна и отсутствует противоречие между теоретически определенным законом и экспериментальными данными.

Вычисление статистики y можно автоматизировать, включив в программу определение значений p_i для рассматриваемого закона распределения. Если принять, что ширина всех классовых интервалов настолько мала, что применение к каждому из них формул трапеций или прямоугольников для интегрирования соответствует допустимой методической погрешности, то сравнительно просто автоматизировать процедуру проверки по критерию χ^2 большинства встречающихся на практике распределений случайных величин. Однако методическая погрешность значительно уменьшается и при использовании следующей программы, где при интегрировании каждый классовый интервал разбивается на четыре части и методическая погрешность уменьшается в 32 раза по сравнению со случаем, когда классовые интервалы приняты в качестве элементарных отрезков интегрирования. Возможно еще большее уменьшение интервалов интегрирования, но в этом случае значительно возрастает время выполнения программы и влияние операционных погрешностей становится соизмеримым с влиянием методической погрешности.

Программа 108/34. Проверка согласия по критерию χ^2

П5	XY	П2	Cx	П6	ИП2	ИПО	—	4	÷
П3	ИПО	П	57	ПП	53	4	ПП	50	2
ПП	50	4	ПП	50	ИП4	+	ИП3	×	3
÷	ИП1	×	↑	ИП5	—	x^2	XY	÷	ИП6
+	П6	ИП2	ПО	С/П	П5	XY	П2	БП	05
×	ИП4	+	П4	ИПО	ИП3	+	ПО	...	В/О

Инструкция: заменить многоточие операторами вычисления функции $p(x)$ для проверяемого закона распределения при $x = PO$ (для записи коэффициентов функции можно использовать регистры $7, \ldots, Д$), $x_{н1} = PO$, $n = P1$, $x_{н2} = PY$, $\psi_1 = PX$ В/О С/П $x_{н3} = PY$, $\psi_2 = PX$ С/П $x_{н4} = PY$, $\psi_3 = PX$ С/П ... $x_{max} = PY$, $\psi_m = PX$ С/П $PX = y$ ($x_{нi}$ — начало i-го классового интервала).

В качестве примера рассмотрим вычисление статистики y при проверке гипотезы о нормальном распределении случайной величины с параметрами $m_1 = 3{,}6$ и $\sigma^2 = 1{,}2$, выборка которой с $n = 100$ сгруппирована по шести классовым интервалам.

При исходных данных $\sigma^2 = P7$, $m_1 = P8$ вычисление функции $p(x) = (e^{-(x-m_1)^2/2})\sqrt{2\pi}\sigma$ для нормального распределения обеспечивается фрагментом ИПО ИП8 — x^2 /—/ 2 ÷ e^x 2 π × ИП7 √ × ÷, который записывается в программу 108/34 вместо многоточия.

Пусть классовыми интервалами являются [—2, 2), [2, 3) [3; 3, 5), [3,5; 4), [4, 5), [5, 9], причем в каждый из них соответственно попадает 3, 9, 30, 45, 9 и 4 значений случайной величины. Тогда после занесения исходных данных в регистры 7 и 8 следует выполнить $x_{н1} = -2 = PO$, $100 = P1$, $x_{н2} = P2$, $\psi_1 = 3 = PX$ В/О С/П

$x_{н3} = 3 = PY$, $\psi_2 = 9 = PX$ С/П $x_{н4} = 3{,}5 = PY$, $\psi_3 = 30$ С/П $x_{н5} = 4 = PY$, $\psi_4 = 45 = PX$ С/П $x_{н6} = 5 = PY$, $\psi_5 = 9$ С/П $x_{max} = 9 = PY$, $\psi_6 = 4$ С/П $PX = y = 91{,}28259$.

По вычисленному значению статистики y для $r = 6 - 2 - 1 = 3$ и выбранного уровня значимости, например, $\alpha = 0{,}05$ по таблице квантилей распределения χ^2 [6] находим $\chi^2_{3;0,05} = 7{,}815$. Так как вычисленное значение статистики значительно превосходит эту величину, то принятая гипотеза несостоятельна с вероятностью ошибки 0,05, хотя по частотам попадания α предположение о нормальном законе распределения с выбранными параметрами представлялось обоснованным.

Для решения этой же задачи с помощью микрокалькулятора с входным языком ЯМК21 предназначена следующая программа, но по ней статистика вычисляется со значительно большей погрешностью, так как в качестве интервала интегрирования принят классовый интервал.

Программа 109/21. Проверка согласия по критерию χ^2

```
P3  ↑  F2  +  2  ÷  ПП FCx F1 4  ×  P7
F2  ПП FCx F3 ПП FCx F2  ↑  F3 P2 —  ↑
F7  ×  ↑  F8  ×  ↑  F6  ×  P7 С/П ↑  F7
XY  —  x²  XY ÷  ↑  →  +  ←  С/П ... F7
+  P7  В/О
```

Инструкция: заменить многоточие операторами вычисления функции $p(x)$ при $x = PX$ и использовании для коэффициентов регистров *4*, *5* и *C1*, ..., *C5* кольцевого стека памяти, занести в регистр *6* нормирующий множитель функции, разделенный на —6 (если этот множитель равен единице, то занести —1/6), $n = P8$, $0 = C6$, $x_{min} = P2$, $x_{н2} = PX$ В/О С/П $\psi_1 = PX$ С/П $x_{н2} = P2$, $0 = C6$, $x_{н3} = PX$ В/О С/П $\psi_2 = PX$ С/П ... $x_{н\,m} = P2$, $0 = C6$, $x_{max} = PX$ В/О С/П $\psi_m = PX$ С/П $PX = y$.

В качестве простейшего примера использования программируемого микрокалькулятора для статистических испытаний оценим изменения вещественного параметра передачи канала с обратной связью $a = \mu / (1 + \beta\mu)$ при случайных изменениях μ и β.

Пусть номинальные значения $\mu = 100$, $\beta = 0{,}1$, случайные изменения β достаточно малы, а μ изменяется случайным образом в пределах 10 % с распределением по равномерному закону. Так как случайные изменения a детерминированы относительно случайных изменений μ в интервале [90, 110], то согласно принятой модели a изменения этого параметра будут находиться в интервале [9; 9,1666666]. Приведя для удобства оценки изменения μ и a в одинаковые интервалы [0, 1], можно вычислить значения случайных величин $\Delta a'_i$ по заданным значениям $\Delta\mu'_i$ с помощью программы

```
↑ 2 0 × 9 0 + ↑ ИПС ×
1 + ÷ 9 — 6 0 × С/П БП
00
```

с инструкцией; $\beta = PC$, $\Delta\mu' = PX$ В/О С/П $PX = \Delta a'$. Для значений $\Delta\mu'$, соответствующих границам 10 классовых интервалов, на которые можно разбить интервал [0, 1], по этой программе соответственно получим $\Delta a' = 0$; 0,1176; 0,2308; 0,3396; 0,4444; 0,5455; 0,6429; 0,7368; 0,8276; 0,9153; 0,99999996. Эти ре-

зультаты свидетельствуют о смещении границ классовых интервалов и соответствующем изменении распределения случайной величины a по отношению к μ.

Оценку изменения закона распределения теоретически можно найти методом статистических испытаний, дополнив составленную программу фрагментом генерирования случайной величины x_i в интервале [0,1] и ее преобразования в интервал изменения a, а также фрагментом для определения частот попадания ψ_j в классовые интервалы и организации цикла для выборки объемом n. В этом случае получим, например, программу

ИПД	π	+	e^x	ПВ	КИПВ	→	ИПВ	— ПД
2	0	×	9	0	+		↑ ИПС	× 1
+	÷	9	—	6	0		× 1	+ ПВ
КИПВ	1	+	КПВ	L0	00			

с инструкцией; произвольное число $x_0 = РД$, $\beta = РС$, $n = РО$ В/О С/П $Р1 = \psi_1$, $Р2 = \psi_2$, ..., $Р9 = \psi_9$, $РА = \psi_{10}$ (время обработки и генерирования одного квазислучайного числа около 11 с).

При $x_0 = 0$, $\beta = 0{,}1$; $n = 100$ (время счета около 20 мин) по этой программе получим частоты попаданий в 10 классовых интервалов $\psi_j =$ 10, 12, 9, 7, 5, 14, 10, 13, 8, 12. Этих данных явно недостаточно для оценки изменения закона распределения и объем выборки должен быть увеличен по крайней мере в 10 раз. Еще большие затраты времени необходимы для анализа одновременного изменения по случайным законам μ и β, так как в этом случае необходимо обеспечить генерирование с заданным законом квазислучайных чисел μ_i и β_i, а детерминированная оценка изменения границ классовых интервалов существенно усложняется.

Этот пример свидетельствует как об алгоритмической простоте метода статистических испытаний, так и о больших затратах времени при его реализации с помощью микрокалькуляторов. Поэтому при необходимости достаточно точных оценок результатов статистических испытаний приходится обращаться к ЭВМ со значительно большим быстродействием.

Глава 8

ОПТИМИЗАЦИЯ РЕШЕНИЙ ИНЖЕНЕРНЫХ ЗАДАЧ

1. ПОСТАНОВКА ЗАДАЧИ ОПТИМИЗАЦИИ

Задачи инженерного проектирования, как и другие задачи синтеза, допускают множество решений, удовлетворяющих требованиям технического задания к рабочим характеристикам проектируемого объекта. Выбор оптимального варианта результата проектирования (проекта) соответствует удовлетворению условий качества, предусматриваемых или подразумеваемых в техническом задании на проектирование. Требования оптимальности (например, требование выполнения проекта в заданный срок) предъявляются и к самому процессу проектирования.

Теоретические модели процесса и результата проектирования, как и любые другие системы, характеризуются структурой и параметрами, для оптимизации которых используют разные методы. Задача

оптимизации структуры особенно сложна в тех случаях, когда отсутствуют критерии качества очередных шагов изменения структуры и их оптимальность удается оценить лишь после определения всех или, по крайней мере, нескольких путей решения задачи.

Наиболее общий метод поиска оптимальной структуры решения задачи заключается в *полном переборе* всех допустимых путей решения и выборе из них наиболее полно соответствующих заданным критериям оптимальности. Среди различных методов перебора основными являются слепой перебор и перебор в глубину.

При *слепом* переборе для каждого очередного состояния решения задачи определяют все состояния, к которым можно перейти, выполнив простейший шаг. Перебор продолжают до определения всех возможных путей перехода из исходного состояния в конечное, после чего выбирают оптимальный путь решения задачи.

При переборе *в глубину* для каждого очередного состояния определяют лишь одно следующее состояние, повторяя такие шаги до достижения конечного состояния, соответствующего искомому результату решения, или тупикового состояния, из которого нельзя перейти к следующему состоянию. В последнем случае возвращаются к предыдущему состоянию и находят для него следующее состояние. Перебор в глубину обеспечивает относительно быстрый поиск пути решения и эффективен в тех случаях, когда его оптимальность можно оценить независимо от других путей решения. Если же требуется найти все пути решения, то перебор в глубину не имеет преимуществ по сравнению со слепым перебором.

В общем случае метод полного перебора весьма трудоемок и практически применим лишь при небольшом числе возможных шагов и вариантов решения задачи. Поэтому для поиска решения сложных задач обычно используют эвристические методы аналогий, индукции и подзадач. Метод *аналогий*, широко используемый в инженерной практике, заключается в поиске для решаемой задачи другой задачи с аналогичной математической моделью исходных условий и известным алгоритмом решения. Этот метод лежит в основе математического моделирования, позволяющего свести решение конкретных прикладных задач к решению типовых математических задач.

При поиске решения часто используют метод *индукции*, заключающийся в обобщении результатов, полученных для частных случаев решения. К этому методу можно отнести и использование различного рода допущений, упрощающих математическую модель условий задачи и ее решение.

В тех случаях, когда результат решения задачи известен и требуется найти оптимальный путь его достижения, широко используют метод *подзадач* (метод «решения с конца»). Этот метод заключается в поиске состояний, из которых можно оптимально перейти в конечное и предыдущие состояния (поиск решения подзадач), что позволяет избежать тупиковых состояний и найти кратчайший путь решения задачи.

Для иллюстрации методики поиска оптимальной структуры решения рассмотрим следующую инженерную задачу: найти способ перемещения за минимальное число операций (например, перемещений подъемного крана) горки

контейнеров различного размера с площадки A на площадку C при использовании дополнительной площадки B, причем большие контейнеры нельзя ставить на меньшие ни в одном из состояний решения задачи. Следовательно, требование оптимальности решения заключается в минимальном числе шагов, соответствующем минимальной стоимости выполняемой работы при ограничении на способ укладки контейнеров.

По условиям задачи контейнеры одинакового размера допустимо ставить в любом порядке и, следовательно, их количество не влияет на структуру решения задачи, увеличивая лишь число одинаковых шагов. Поэтому будем учитывать лишь контейнеры различного размера, обозначая их (начиная с меньших) последовательностью символов A, B и C площадок, на которых в каждом состоянии решения задачи находится контейнер. В этом случае исходное состояние (все контейнеры на площадке A) опишется последовательностью AAA,... а конечное — последовательностью CCC... .

В простейшем случае контейнеров двух размеров (рис. 40, *а*) методом полного перебора несложно составить граф состояний (рис. 40,*б*), вершины которых соответствуют допустимым условиями задачи состояниям, а ветви — шагам

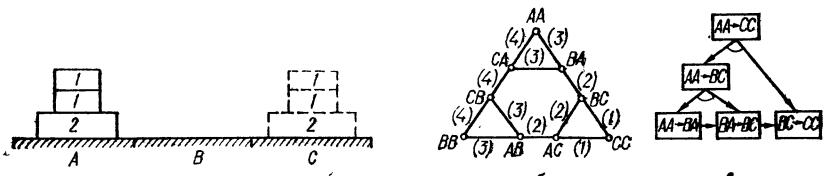


Рис. 40

(переходам между состояниями) решения задачи. После составления такого графа легко определить оптимальный (кратчайший) путь решения задачи, содержащий вершины AA, BA, BC, CC, с минимальным числом шагов $n = 3$.

Для упрощения поиска кратчайшего пути решения по сложному графу состояний с большим числом вершин целесообразно оценить (взвесить) все ветви, входящие в конечную вершину, числом 1 оставшихся до решения шагов, ветви, входящие в предыдущие вершины, числом 2, взвесив подобным образом все ветви. Тогда для выбора оптимального шага достаточно выбрать шаг с минимальным весом (для примера веса ветвей указаны в скобках на рис. 40,*б*).

Решение рассматриваемой задачи можно упростить, используя метод подзадач. Исследовав конечное состояние, показанное на рис. 40,*а* штриховыми линиями, несложно установить, что в предыдущем состоянии меньшие контейнеры должны находиться на площадке B, а больший — на площадке C (состояние BC). В это состояние можно перейти из состояния BA, которое достигается одним шагом из исходного состояния AA. Поиск оптимального решения в этом случае можно отобразить графом подзадач (рис. 40,*в*), из которого следует, что исходная задача перехода $AA \to CC$ вначале разбивается на подзадачи переходов $AA \to BC$ и $BC \to CC$, после чего первая разбивается на элементарные (одношаговые) задачи $AA \to BA$ и $BA \to BC$.

Повторив аналогичный поиск для задачи с числом $m = 3$ контейнеров различных размеров, можно установить оптимальный путь решения задачи ($AAA \to CBA \to CBA \to BBA \to BBC \to ABC \to ACC \to CCC$) за минимальное число $n = 7$ шагов. Сравнивая числа m различных контейнеров и n оптимальных шагов, можно найти связь между ними в виде формулы $n = 2^m - 1$. Проверив это соотношение для $m = 4$, допустимо обобщить эту формулу на произвольное число m контейнеров различного размера. Отыскав оптимальные пути решения задачи при различных m, можно составить таблицу оптимальных перемещений контейнеров, удобную для практических целей.

При большом числе m удобнее использовать не перечень оптимальных шагов, а алгоритм, обеспечивающий определение оптимального очередного шага. Подобный алгоритм известен для классической задачи о ханойской башне или пирамидке [5], аналогичной рассматриваемой задаче. Применительно к послед-

ней этот алгоритм заключается в поочередном перемещении каждого нечетного контейнера (при их порядковой нумерации в исходном состоянии порядковыми номерами сверху вниз) на другую площадку в последовательности $CABC...$ Если при этом сопоставить номера контейнеров различного размера, начиная с меньших, с разрядами (начиная с младших) двоичного представления номера N шага, то первая единица справа в этом двоичном представлении будет указывать на номер перемещаемого контейнера. Двоичные представления номеров N шагов можно найти с помощью микрокалькулятора, например, по программе 139/34 в приложении, пригодной для оптимального решения задачи с $m \leqslant 8$ контейнерами различного размера (или m групп контейнеров одинакового размера в каждой группе). Для случая $m > 8$ можно модифицировать такую программу, разбивая результат промежуточных вычислений на два восьмиразрядных блока, заносимых в различные регистры памяти, что обеспечит преобразование в двоичный код десятичных представлений чисел $N \leqslant 2^{16} - 1 = 64355$. Однако практически уже при $m = 8$ минимальное число $n = 255$ перемещений достаточно велико и целесообразно изменить условия задачи. Так, при удвоении числа площадок для размещения контейнеров оптимальное число перемещений сокращается до $2(2^4 - 1) = 30$ вместо 255.

Таким образом, возможны, по крайней мере, два способа описания алгоритма оптимального решения задачи — перечнем (списком) оптимальных шагов и программой, генерирующей код очередного оптимального шага. При использовании ЭВМ предпочтительнее второй способ, связанный с меньшими требованиями к ресурсу памяти, особенно в тех случаях, когда выбор оптимального шага зависит от изменения условий оптимальности в процессе решения задачи. Этот способ часто реализуется методами теории игр, предусматривающими выбор оптимальных путей (стратегий) решения задач с конфликтными ситуациями. Противником в таких играх могут быть и стихийные силы природы, к которым можно отнести и непредвиденные заранее трудности, возникающие при решении задачи.

Строго говоря, к конфликтным ситуациям относятся все этапы поиска оптимальных путей решения задачи, связанные с проверкой выполнения определенных условий. Игровая задача, например, возникает при разработке защиты проектируемого изделия от последствий возможных ошибочных действий его пользователя или воздействия стихийных сил природы, причем эти действия могут быть как детерминированными, так и случайными.

В качестве примера рассмотрим разработку аппаратуры, которая должна автоматически приводиться в определенное состояние в зависимости от вероятности дождя на следующий день. Если предсказание погоды определяется метеорологическими средствами, то такая аппаратура должна содержать устройство, вычисляющее случайные числа 1 (будет дождь) или 0 (не будет дождя) с достаточно высокой вероятностью. Такое устройство должно реализовать предсказание событий 1 или 0 с заданной вероятностью по программам, подобным, например, программе 103/34, причем вероятность дождя должна определяться по известным метеорологическим данным.

Если вероятность дождя определить затруднительно, целесообразно учесть связь между периодами дождей и деятельностью циклонов. Вследствие этой связи вероятность изменения погоды на следующий день в среднем меньше вероятности ее сохранения и, как показывает практика, при предсказании на завтра сегодняшней погоды число верных прогнозов в среднем превышает 50 %. Более точно можно предсказать дождь, вычисляя корреляционную функцию последовательности дождливых и ясных дней в виде последовательности квазислучайных символов 1 и 0 с помощью следующей программы, определяющей коэффи-

циент корреляции по трем предыдущим элементам последовательности и генерирующей соответственно символы 1 или 0.

Программа 110/34. Вычисление предсказания x_i' символов 1 или 0 последовательности x_i по корреляционной функции

sin	$x \geqslant 0$	04	Cx	↑	2	×	1	—	ПО
ИП4	×	ПС	ИП3	П4	ИПО	×	ПВ	ИП2	П3
ИП0	П2	×	ИП5	+	П5	ИП2	×	ПА	ИПВ
ИП6	+	П6	ИП3	×	ПВ	ИПС	ИП7	+	П7
ИП4	×	ПС	ИПВ	+	ПВ	ИПА	+	$x \geqslant 0$	53
1	БП	54	Cx	С/П	БП	04			

Инструкция: очистить регистры 2...; 7, произвольное число $a = PX$ В/О С/П $PX = x_1'$, $x_1 = PX$ С/П $PX = x_2'$, $x_2 = PX$ С/П $PX = x_3'$, $x_3 = PX$ С/П... (переключатель Р—Г установить в положение Р).

При $a = 1983$ для последовательности $x_i = 000111000111000111000111000111$ получим предсказуемую последовательность $x_i' = 100001100011100111000111000111$, в которой верны 24 из 30 предсказаний.

Подобные программы применимы и в тех случаях, когда в процессе оптимизации структуры решения задачи или оптимизации параметров приходится выбирать одно из двух возможных решений в зависимости от предыдущего изменения условий принятия решений. Принадлежность подобных задач к игровым становится очевидной при использовании микрокалькулятора с такими программами в игре «чет — нечет», когда один из участников загадывает, а другой отгадывает одно из двух возможных задуманных состояний («чет» или «нечет» или 1 и 0). Программа 103/34 при $p(1) = 0{,}5$ обеспечивает тем больший успех микрокалькулятора в качестве загадывающего, а программа 110/34 — отгадывающего участника этой игры, чем дольше она длится. Кстати, причина такого успеха заключается не в «мыслительных» способностях программируемого микрокалькулятора, а в его способности генерировать по соответствующей программе случайные события, тогда как для человека принятие решения всегда коррелировано.

К играм с природой можно отнести и экспериментальный поиск оптимальных инженерных решений на макете проектируемого объекта или с помощью его цифровой модели, реализованной на ЭВМ.

В качестве простейшего примера рассмотрим задачу моделирования движения корабля с массой m в зависимости от изменения по величине и направлению силы тяги F движителя. Согласно закону Ньютона с учетом сопротивления воды, пропорционального квадрату скорости корабля, выберем исходным дифференциальное уравнение

$$m\bar{v}' = \bar{F} - k\bar{v}^2,$$

где $\bar{v}$ — вектор скорости движения корабля и k — коэффициент пропорциональности, зависящий от конструкции корабля.

Переходя к декартовой системе координат x, y и обозначая символом φ угол между положительным направлением оси x и направлением силы тяги, представим это уравнение двумя скалярными уравнениями

$$mv_x' = F\cos\varphi - kv_x|v_x|,\ mv_y' = F\sin\varphi - kv_y|v_y|.$$

Заменяя составляющие скорости производными координат по времени, получим дифференциальные уравнения второго порядка

$$mx'' = F\cos\varphi - kx'|x'|,\ my'' = F\sin\varphi - ky'|y'|.$$

Согласно рассмотренной в гл. 7 методике заменим производные разностными отношениями и получим аппроксимирующие разностные уравнения

$$m(x_{t+1} - x_t - \Delta x_t) = (\Delta t)^2 F_t \cos\varphi_t - k\Delta x_t |\Delta x_t|;$$
$$m(y_{t+1} - y_t - \Delta y_t) = (\Delta t)^2 F_t \sin\varphi_t - k\Delta y_t |\Delta y_t|$$

и расчетные формулы

$$x_{t+1} = x_t + \Delta x_t + P_t \cos\varphi_t - B\Delta x_t |\Delta x_t|;$$
$$y_{t+1} = y_t + \Delta y_t + P_t \sin\varphi_t - B\Delta y_t |\Delta y_t|,$$

где $\Delta x_t = x_t - x_{t-1}$, $\Delta y_t = y_t - y_{t-1}$, нормированная сила тяги $P_t = (\Delta t)^2 F_t/m$, $B = k/m$.

Выбрав размерности расстояний и времени и задаваясь массой m, максимальной силой тяги F_{max} и максимальной скоростью корабля, определим $B = F_{max}/v_{max}^2$.

Составленные расчетные формулы реализуем на входном языке ЯМК34 следующей цифровой моделью, включающей счетчик («часы») числа выполнений программы.

Программа 111/34. Цифровая модель движения корабля

КИП4	ИП9	ИП8	sin	×	ИП3	↑	x^2	$\sqrt{}$	×
ИПВ	×	—	ИП3	+	П3	ИП2	+	П2	ИП9
ИП8	cos	×	ИП6	↑	x^2	$\sqrt{}$	×	ИПВ	×
—	ИП6	+	П6	ИП5	+	П5	С/П	БП	00

Инструкция: установить переключатель Р—Г в положение Г, $0 = P3 = P4 = P6$, $x_0 = P5$, $y_0 = P2$, $B = PB$, $\varphi_t = P8$, $P_t = P9$ (значения φ_t и P_t достаточно вводить только при их изменениях) (В/О) С/П $PX = P5 = x_t$, $PY = P2 = y_t$, $P6 = \Delta x_t$, $P3 = \Delta y_t$, $P4 = t$.

Для использования этой цифровой модели следует начертить карту акватории с координатной сеткой, выбрать размерность Δt (для кораблей с большой инерционностью целесообразно выбирать $\Delta t = 1$ мин, для катеров и небольших кораблей более точные результаты получаются при $\Delta t = 1$ с), задаться характеристиками корабля m, v_{max}, F_{max}, $(\Delta P_t/\Delta t)_{доп}$, $(\Delta\varphi_t/\Delta t)_{доп}$ и вычислить B. Выбрав начальные координаты корабля x_0, y_0 и задаваясь нормированной тягой и курсовым углом, можно исследовать поведение корабля в различных навигационных условиях и выбрать его оптимальные характеристики. Следовательно, подобные цифровые модели могут быть использованы как для оптимизации характеристик корабля в процессе математического эксперимента, так и для тренировок судоводителей в процессе навигационных игр.

Связь методики оптимизации с теорией игр и теорией операций обусловлена тем, что обычно оптимальные решения являются компромиссом между противоречивыми требованиями оптимальности (например, требованиями максимальной надежности и минимальной стоимости проектируемого объекта), отображающими конфликтные ситуации инженерного проектирования. Однако теория игр и операций в основном дает лишь общие рекомендации и не может охватить бесконечного многообразия конфликтных ситуаций.

Оптимизация структуры решения задачи или структуры проектируемого объекта существенно упрощается при наличии количественных критериев выбора решения на каждом шаге изменения структуры в процессе ее оптимизации. В этом случае, как и при оптимизации параметров любой системы, задача сводится к поиску оптимальных численных значений переменных.

Решение задач, в которых число неизвестных равно числу связывающих эти переменные уравнений, заключается в решении этой системы уравнений относительно неизвестных переменных. Если число n

переменных больше числа r уравнений, то значения r переменных могут быть найдены решением системы уравнений, а остальные $m = n - r$ переменных приходится подбирать так, чтобы их значения наилучшим образом (оптимально) удовлетворяли всем условиям задачи, включая критерии качества ее решения.

Методы целенаправленного поиска оптимальных значений переменных относят к разделу математики, называемому математическим программированием и не имеющему непосредственного отношения к программированию ЭВМ. Эти методы основаны на составлении вещественной функции оптимизируемых переменных, называемой целевой и принимающей экстремальное значение при оптимальных значениях переменных, что позволяет свести задачу оптимизации переменных к поиску экстремума этой функции.

Основную задачу математического программирования (задачу численной оптимизации) обычно формулируют следующим образом: найти экстремальное (минимальное или максимальное) значение целевой функции

$$\Phi = \Phi(x_1, x_2, \ldots, x_n), \tag{8.1}$$

соответствующее оптимальным значениям $x_1^*, x_2^*, \ldots, x_n^*$ оптимизируемых переменных при ограничениях в виде равенств

$$H_l(x_1, x_2, \ldots, x_n) = 0, \quad l = 1, 2, \ldots, r$$

и ограничениях в виде неравенств

$$G_j(x_1, x_2, \ldots, x_n) \geqslant 0, \quad j = 1, 2, \ldots, k.$$

При большом числе оптимизируемых параметров задача численной оптимизации может оказаться непосильной даже для ЭВМ высокой производительности, но встречающиеся в инженерной практике задачи с небольшим числом оптимизируемых переменных могут быть успешно решены с помощью непрограммируемых и особенно программируемых микрокалькуляторов.

2. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Математическое программирование называют *линейным*, если целевая функция и левые части всех ограничений линейны относительно оптимизируемых переменных. В этом случае задача численной оптимизации сводится к поиску экстремального значения целевой функции

$$\Phi = d_1x_1 + d_2x_2 + \ldots + d_nx_n + d_0$$

при ограничениях-равенствах

$$b_{l1}x_1 + b_{l2}x_2 + \ldots + b_{ln}x_n + b_l = 0, \quad l = 1, 2, \ldots, r$$

и ограничениях-неравенствах

$$b_{j1}x_1 + b_{j2}x + \ldots + b_{jn}x_n + b_j \geqslant 0, \quad j = 1, 2, \ldots, k.$$

При решении этой задачи число переменных на первом этапе уменьшают, исключив согласно ограничениям-равенствам r называемых *свободными* переменных

$$x_l = e_{l1}x_1 + e_{l2}x_2 + \ldots + e_{lm}x_m + e_{lm}x_m + e_l; \quad l = 1, 2, \ldots, r \tag{8.2}$$

и сводят основную задачу линейного программирования к поиску минимума* целевой функции

$$\Phi = c_1x_1 + c_2x_2 + \ldots + c_mx_m + c_0$$

при ограничениях-неравенствах

$$a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jm}x_m + a_j \geqslant 0, \quad j = 1, 2, \ldots, k$$

с вычислением на втором этапе r свободных переменных по формуле (8.2).

В пространстве m оптимизируемых переменных каждой точке с проекциями x_1, x_2, ..., x_n на координатные оси соответствует определенное значение целевой функции. Параллельные плоскости m-го порядка (прямые линии при $m = 2$ и гиперплоскости при $m > 3$), на которых целевая функция постоянна, называют плоскостями *равного уровня*.

Каждому ограничению-неравенству соответствует граничная плоскость, разделяющая пространство оптимизируемых переменных на две части, в одной из которых и на граничной плоскости удовлетворяется это неравенство. Область пространства (многогранник), ограниченная такими плоскостями, называют областью *допустимых* решений, а оптимальному решению соответствует вершина такой области или ее грань, параллельная плоскости равных уровней целевой функции. Задача линейного программирования и заключается в отыскании такого оптимального решения, соответствующего оптимальным значениям x_1^*, x_2^*, ..., x_3^* переменных. Если в пространстве переменных нет области, где удовлетворяются все ограничения-неравенства, то они несовместны и задача оптимизации неразрешима.

Для примера рассмотрим следующую типичную задачу линейного программирования. Цех ежедневно должен изготовить 100 изделий трёх типов, в том числе не менее 20 изделий каждого типа. На изготовление изделий каждого типа затрачивается соответственно 4; 3,4 и 2 кг металла при его запасе 340 кг и 4,75; 11 и 2 кг пластмассы при запасе 700 кг. Требуется составить оптимальный план выпуска x_1, x_2 и x_3 изделий соответствующего типа, обеспечивающий максимальную стоимость продукции при стоимости изделия каждого типа соответственно 4, 3 и 2 руб.

Решение рассматриваемой задачи заключается в поиске максимального значения целевой функции (стоимости продукции)

$$\Phi = 4x_1 + 3x_2 + 2x_3$$

при ограничениях-неравенствах $x_1 \geqslant 20$; $x_2 \geqslant 20$; $x_3 \geqslant 20$; $4x_1 + 3{,}4x_2 + 2x_3 \leqslant 340$; $4{,}75x_1 + 11x_2 + 2x_3 \leqslant 700$ и ограничении-равенстве $x_1 + x_2 + x_3 = 100$.

Подставив в исходную целевую функцию и ограничения-неравенства свободную переменную $x_3 = 100 - x_1 - x_2$, сведем задачу к максимизации функции двух переменных

$$\Phi = 2x_1 + x_2 + 200$$

при ограничениях-неравенствах $x_1 \geqslant 20$; $x_2 \geqslant 20$; $0{,}375x_1 + x_2 \leqslant 60$; $x_1 + 0{,}7x_2 \leqslant 70$; $x_1 + x_2 \leqslant 80$.

Условия этой задачи, как и других задач линейного программирования при $m = 2$, удобно отображать графически линиями на плоскости переменных x_1 и x_2 (рис. 41). Решая совместно по два равенства, соответствующие пересекающимся граничным линиям, несложно найти координаты вершин области до-

* Эта формулировка применима и для поиска минимума линейной функции z, соответствующего максимуму функции $\Phi = -z$.

пустимых решений $(x_{1j}, x_{2j}) = (20; 20), (20; 52), (32; 48), (46, 66666; 33,33333), (56; 20)$ и значения целевой функции в этих вершинах $\Phi_j = 260; 292; 310; 326,6666; 332$.

Следовательно, оптимальное решение при максимуме целевой функции $\max\Phi = 332$ соответствует значениям $x_1^* = 56; x_2^* = 20; x_3^* = 100 - x_1^* - x_2^* = 24$, причем (как следует из подстановки переменных в исходные неравенства) полностью расходуется металл, но остается 166 кг пластмассы.

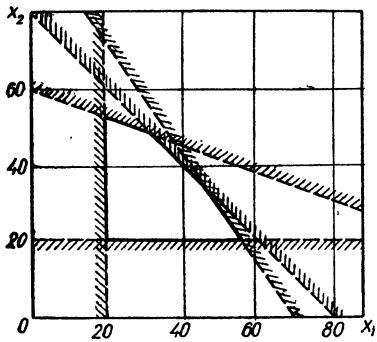


Рис. 41

В случае нецелочисленных значений координат оптимальной вершины (при целочисленных значениях оптимальных переменных по условиям задачи) следует выбрать в области допустимых значений ближайшую точку с целочисленными значениями координат и принять ее в качестве оптимума.

Графический метод при $m > 3$ практически неприменим, а при отсутствии алгоритма, упорядочивающего перебор вершин, вычисление координат пересечения граничных плоскостей громоздко. Поэтому при оптимизации большого числа переменных прибегают к регулярным методам линейного программирования, среди которых наиболее универсален *симплекс-метод*.

При использовании симплекс-метода после исключения свободных членов целевую функцию и ограничения-неравенства (исключая неравенства $x \geqslant 0$ или $x \leqslant 0$, определяющие допустимые знаки переменных) записывают в упорядоченной форме равенств

$$\Phi = -c_1(-x_1) - c_2(-x_2) - \ldots - c_m(-x_m) + c_0;$$
$$y_j = a_{j1}(-x_1) + a_{j2}(-x_2) + \ldots + a_{jm}(-x_m) + a_j,\ j = 1, 2, \ldots, k,$$

где y_j — неотрицательные переменные, приводящие неравенства к равенствам. Для удобства расчета эти равенства записывают в матрицу, называемую исходной симплекс-таблицей:

	$-x_1$	$-x_2$	...	$-x_m$	1
y_1	a_{11}	a_{12}	...	a_{1m}	a_1
y_2	a_{12}	a_{22}	...	a_{2m}	a_2
...	...	...	...	...	...
y_k	a_{k1}	a_{k2}	...	a_{km}	a_k
Φ	$-c_1$	$-c_2$	...	$-c_m$	c_0

Оптимизацию целевой функции симплекс-методом разбивают на поиск опорного решения, соответствующего одной из вершин области допустимых решений, и поиск оптимального решения, соответствующего вершине с максимальным значением целевой функции.

Опорному решению соответствует симплекс-таблица с неотрицательными значениями всех свободных членов $a_j = a_{j,m+1}$. Если в исходной таблице это условие не соблюдается, то опорное решение находят согласно следующему алгоритму:

1. В строке с наибольшим по модулю отрицательным свободным членом a_h выбирают отрицательный элемент a_{hs} и s-й столбец называют разрешающим. Если в s-й строке все коэффициенты положительны, то задача оптимизации неразрешима в связи с противоречивостью исходных условий.

2. Для положительных элементов разрешающего столбца (кроме элемента Ф-строки) вычисляют неотрицательные частные a_j/a_{js}, а элемент a_{rs} и r-ю строку, соответствующие наименьшему значению a_r/a_{rs}, называют *разрешающими*.

3. Выполняют преобразование симплекс-таблицы, называемое шагом модифицированного жорданова исключения, вычисляя по значениям элементов таблицы и записывая в новую таблицу значения элемента $a'_{rs} = 1/a_{rs}$, остальных элементов r-й строки $a'_{ri} = a_{ri}/a_{rs}$, остальных элементов s-го столбца $a'_{js} = -a_{js}/a_{rs}$ и остальных элементов $a'_{ji} = a_{ji} - a_{js}a_{ri}/a_{rs}$, после чего меняют местами символы переменных (без учета отрицательного знака) в s-м столбце и r-й строке.

4. Повторяют предыдущие пункты до получения опорного решения

Оптимальное решение находят из опорного согласно следующему алгоритму:

1. Выбирают разрешающим s-й столбец с максимальным по модулю отрицательным элементом Ф-строки. Разрешающие строку и элемент выбирают по наименьшему неотрицательному значению a_r/a_{rs}, как и при поиске опорного решения.

2. Выполняют шаг модифицированного жорданова исключения. Преобразование таблицы целесообразно начинать с Ф-строки и столбца свободных членов — если все они неотрицательны, то оптимальное решение найдено и нет необходимости в преобразовании остальных элементов таблицы.

3. Если Ф-строка содержит отрицательные элементы, то предыдущие шаги повторяют до отыскания оптимального решения.

Симплекс-таблицу можно упростить, если в процессе вычислений в строке с нулевым свободным членом все элементы неотрицательны — в этом случае вычеркивают столбцы таблицы, содержащие положительные коэффициенты в такой строке, и саму строку.

В симплекс-таблице, соответствующей оптимальному решению, максимальное значение целевой функции записано в правой нижней клетке, оптимальные значения переменных x_i^* и y_j^*, символы которых записаны в крайнем левом столбце, равны свободным членам в той же строке, а оптимальные значения переменных, символы которых записаны в верхней строке, равны нулю. Ненулевые

пустимых решений $(x_{1j}, x_{2j}) = (20; 20), (20; 52), (32; 48), (46, 66666; 33,33333), (56; 20)$ и значения целевой функции в этих вершинах $\Phi_j = 260; 292; 310; 326,6666; 332$.

Следовательно, оптимальное решение при максимуме целевой функции $\max\Phi = 332$ соответствует значениям $x_1^* = 56; x_2^* = 20; x_3^* = 100 - x_1^* - x_2^* = 24$, причем (как следует из подстановки переменных в исходные неравенства) полностью расходуется металл, но остается 166 кг пластмассы.

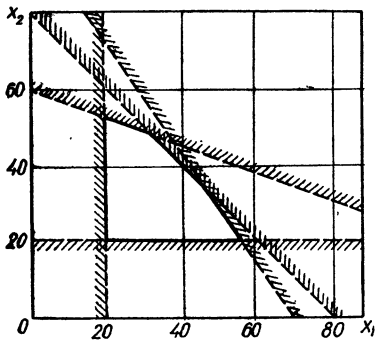


Рис. 41

В случае нецелочисленных значений координат оптимальной вершины (при целочисленных значениях оптимальных переменных по условиям задачи) следует выбрать в области допустимых значений ближайшую точку с целочисленными значениями координат и принять ее в качестве оптимума.

Графический метод при $m > 3$ практически неприменим, а при отсутствии алгоритма, упорядочивающего перебор вершин, вычисление координат пересечения граничных плоскостей громоздко. Поэтому при оптимизации большого числа переменных прибегают к регулярным методам линейного программирования, среди которых наиболее универсален *симплекс-метод*.

При использовании симплекс-метода после исключения свободных членов целевую функцию и ограничения-неравенства (исключая неравенства $x \geqslant 0$ или $x \leqslant 0$, определяющие допустимые знаки переменных) записывают в упорядоченной форме равенств

$$\Phi = -c_1(-x_1) - c_2(-x_2) - \ldots - c_m(-x_m) + c_0;$$
$$y_j = a_{j1}(-x_1) + a_{j2}(-x_2) + \ldots + a_{jm}(-x_m) + a_j,\ j = 1, 2, \ldots, k,$$

где y_j — неотрицательные переменные, приводящие неравенства к равенствам. Для удобства расчета эти равенства записывают в матрицу, называемую исходной симплекс-таблицей:

	$-x_1$	$-x_2$	...	$-x_m$	1
y_1	a_{11}	a_{12}	...	a_{1m}	a_1
y_2	a_{12}	a_{22}	...	a_{2m}	a_2
...	...	...	...	...	...
y_k	a_{k1}	a_{k2}	...	a_{km}	a_k
Φ	$-c_1$	$-c_2$	...	$-c_m$	c_0

Оптимизацию целевой функции симплекс-методом разбивают на поиск опорного решения, соответствующего одной из вершин области допустимых решений, и поиск оптимального решения, соответствующего вершине с максимальным значением целевой функции.

Опорному решению соответствует симплекс-таблица с неотрицательными значениями всех свободных членов $a_j = a_{j,m+1}$. Если в исходной таблице это условие не соблюдается, то опорное решение находят согласно следующему алгоритму:

1. В строке с наибольшим по модулю отрицательным свободным членом a_h выбирают отрицательный элемент a_{hs} и s-й столбец называют разрешающим. Если в s-й строке все коэффициенты положительны, то задача оптимизации неразрешима в связи с противоречивостью исходных условий.

2. Для положительных элементов разрешающего столбца (кроме элемента Ф-строки) вычисляют неотрицательные частные a_j/a_{js}, а элемент a_{rs} и r-ю строку, соответствующие наименьшему значению a_r/a_{rs}, называют *разрешающими*.

3. Выполняют преобразование симплекс-таблицы, называемое шагом модифицированного жорданова исключения, вычисляя по значениям элементов таблицы и записывая в новую таблицу значения элемента $a'_{rs} = 1/a_{rs}$, остальных элементов r-й строки $a'_{ri} = a_{ri}/a_{rs}$, остальных элементов s-го столбца $a'_{js} = -a_{js}/a_{rs}$ и остальных элементов $a'_{ji} = a_{ji} - a_{js}a_{ri}/a_{rs}$, после чего меняют местами символы переменных (без учета отрицательного знака) в s-м столбце и r-й строке.

4. Повторяют предыдущие пункты до получения опорного решения

Оптимальное решение находят из опорного согласно следующему алгоритму:

1. Выбирают разрешающим s-й столбец с максимальным по модулю отрицательным элементом Ф-строки. Разрешающие строку и элемент выбирают по наименьшему неотрицательному значению a_r/a_{rs}, как и при поиске опорного решения.

2. Выполняют шаг модифицированного жорданова исключения. Преобразование таблицы целесообразно начинать с Ф-строки и столбца свободных членов — если все они неотрицательны, то оптимальное решение найдено и нет необходимости в преобразовании остальных элементов таблицы.

3. Если Ф-строка содержит отрицательные элементы, то предыдущие шаги повторяют до отыскания оптимального решения.

Симплекс-таблицу можно упростить, если в процессе вычислений в строке с нулевым свободным членом все элементы неотрицательны — в этом случае вычеркивают столбцы таблицы, содержащие положительные коэффициенты в такой строке, и саму строку.

В симплекс-таблице, соответствующей оптимальному решению, максимальное значение целевой функции записано в правой нижней клетке, оптимальные значения переменных x_i^* и y_j^*, символы которых записаны в крайнем левом столбце, равны свободным членам в той же строке, а оптимальные значения переменных, символы которых записаны в верхней строке, равны нулю. Ненулевые

значения переменных y_j^* определяют величины свободных членов неравенств, преобразующих их в равенства при оптимальных значениях переменных x_i^*. Таким образом, использование симплекс-метода в основном сводится к несложным вычислениям, выполнимым на микрокалькуляторе любого типа.

Для примера решим симплекс-методом ранее решенную задачу, составив по ее условиям после исключения свободной переменной x_3 (рис. 41) начальную симплекс-таблицу (табл. 17). Выбрав по свободному члену $a_{i1} = a_{13} = -20$ разрешающий элемент $a_{51} = 1$ и выполнив шаг модифицированного жорданова исключения, получим новую таблицу (табл. 17), в которой выберем разрешающим элемент $a_{32} = 0{,}3$. Выполнив преобразование, получим симплекс-таблицу, соответствующую опорному решению в вершине области допустимых решений с координатами $x_1 = 46{,}66666$ и $x_2 = 33{,}33333$. Выбрав разрешающим элемент $a_{22} = 3{,}3333333$ и выполнив преобразование, получим оптимальное решение (табл. 17) с $\max\Phi = 332$, $x_1^* = 56$, $x_2^* = 20$, $x_3^* = 100 - x_1^* - x_2^* = 24$, причем оптимальная вершина области допустимых решений находится на пересечении граничных линий второго и третьего неравенств, так как $y_3^* = y_5^* = 0$.

17. Решение задачи об оптимальном плане

0		$-x_1$	$-x_2$	1	2		$-y_5$	$-y_3$	1
	y_1	−1	0	−20		y_1	3,333333	−2,333333	26,66666
	y_2	0	−1	−20		y_2	−3,333333	3,333333	13,33333
	y_3	1	1	80		x_2	−3,333333	3,333333	33,33333
	y_4	0,375	1	60		y_4	2,083333	−2,458333	9,166666
	y_5	1	0,7	70		x_1	3,333333	−2,333333	46,66666
	Φ	−2	−1	200		Φ	3,333333	−1,333333	326,6666
1		$-y_5$	$-x_2$	1	3		$-y_5$	$-y_2$	1
	y_1	1	0,7	50		y_1			36
	y_2	0	−1	−20		y_3			4
	y_3	−1	0,3	10		x_2			20
	y_4	−0,375	0,7375	33,75		y_4			19
	x_1	1	0,7	70		x_1			56
	Φ	2	0,4	340		Φ	2	0,4	332

Вычисления при использовании симплекс-метода можно ускорить, автоматизировав их с помощью программируемого микрокалькулятора. Выполнение шага модифицированного жорданова исключения на входном языке ЯМК34 реализуется для симплекс-таблиц с числом столбцов, ограниченным (после одного цикла выполнения программы) лишь числом регистров памяти.

Программа 112/34. Преобразование симплекс-таблиц с числом столбцов $m < 14$

```
ИПД ÷ КПС ИПС 1 + ПС XY С/П ИПД
КИПС × —  БП  03
```

Инструкция: вычислить и занести в новую симплекс-таблицу: 1) разрешающий элемент, выполнив $a_{rs}^{(p-1)} = РД$, $1 = РХ$ В/О С/П $РХ = \alpha_{rs}^{(p)}$; 2) остальные элементы разрешающего столбца, выполнив $-a_{is}^{(p-1)} = РХ$ В/О С/П $РХ = a_{is}^{(p)}$ для всех i (кроме $i = r$); 3) остальные элементы разрешающей строки, выполнив $(0 =$ РС$)$, $a_{rj}^{(p-1)} = РХ$ В/О С/П $РХ = a_{rj}^{(p)}$ для всех j (кроме $j = s$); 4) остальные элементы неразрешающих строк, выполнив для элементов каждой i-й строки (кроме $i = r$) $(0 =$ РС, $a_{is}^{(p-1)} = РД)$ $a_{ij}^{(p-1)} = РХ$ С/П $РХ = a_{ij}^{(p)}$ для всех j (кроме $j = s$) (значение j для контроля сохраняется в регистре Y).

На входном языке ЯМК21 в связи с отсутствием операторов косвенной адресации и меньшим объемом памяти выполнение модифицированного жорданова исключения удается реализовать несколько менее успешно.

Программа 113/21. Преобразование $m + 1 \leqslant 11$ столбцов симплекс-таблицы

↑ F8 /—/ P7 С/П ПП ВП С/П ПП ВП С/П ПП
ВП С/П ПП ВП С/П ПП ВП С/П ПП ВП С/П ПП
ВП С/П P6 F3 ПП 7 С/П P6 F4 ПП 7 С/П
P6 F5 ПП 7 С/П P6 F2 ← P2 ↑ F7 ×
↑ F6 + В/О

Инструкция: 1) вычислить и занести в новую симплекс-таблицу разрешающий элемент, выполнив $a_{rs}^{(p-1)} = РХ$ $1/x$ P8 $РХ = a_{rs}^{(p)}$; 2) ввести в память элементы разрешающей строки (пропустив $j = s$) $a_{r1} = C1$, $a_{r2} = C2$, ..., $a_{r6} = C6$, $a_{r7} = P2$, $a_{r8} = P3$, $a_{r9} = P4$, $a_{r10} = P5$; 3) вычислить и занести в новую таблицу элементы неразрешающих строк $(i \neq r)$, выполнив для каждой строки $a_{is}^{(p-1)} = РХ$ В/О С/П $РХ = a_{is}^{(p)}$ и для всех элементов строки (кроме s-го $a_{ij}^{(p-1)} = РХ$ С/П $РХ = a_{ij}^{(p)}$; если $m < 8$, то после вычисления элементов строки повернуть $8 - m$ раз кольцевой стек против часовой стрелки; 4) вычислить и занести в новую таблицу элементы разрешающей строки, вызывая из памяти $a_{rj}^{(p-1)}$ и умножая эти числа на содержимое регистра *8*.

Приведенные программы пригодны и для преобразования таблиц с произвольным числом столбцов. Для этого после преобразования элементов предельного числа столбцов вводят в память следующую допустимую по количеству последовательность элементов разрешающей строки и вычисляют элементы соответствующей части таблицы.

В качестве примера решим следующую типовую задачу линейного программирования. На организацию автобазы с трехсменной работой выделен фонд заработной платы для 145 водителей и 300 тыс. руб. для приобретения не более 30 автомобилей трех типов стоимостью 5, 10 и 12 тыс. руб. каждый, обслуживаемых соответственно одним, двумя и тремя водителями в смену при производительности 2,1; 3,6 и 3,8 т · км. Требуется определить число автомобилей каждого типа, обеспечивающих максимальную производительность при указанных ограничениях.

18. Решение симплекс-методом задачи о максимальной производительности автобазы

0		$-x_{11}$	$-x_{12}$	$-x_{13}$	$-x_{21}$	$-x_{22}$	$-x_{23}$	$-x_{31}$	$-x_{32}$	$-x_{33}$	1
	y_1 y_2 y_3	1 5 1	2 5 1	3 5 1	2 10 1	4 10 1	6 10 1	2 12 1	4 12 1	6 12 1	145 300 30
	Φ	—2,1	—4,2	—6,3	—3,6	—7,2	—10,8	—3,8	—7,6	—11,4	0
1		$-x_{11}$	$-x_{12}$	$-x_{13}$	$-x_{21}$	$-x_{22}$	$-x_{23}$	$-x_{31}$	$-x_{32}$	$-y_1$	1
	x_{33} y_2 y_3	0,16666666 3 0,83333333	0,33333333 1 0,66666666	0,5 —1 0,5	0,33333333 6 0,66666666	0,66666666 2 0,33333333	1 —2 0	0 33333333 8 0,66666666	0,66666666 4 0,33333333	0,16666666 —2 —0,16666666	24,166666 10 5,8333333
	Φ	—0,2	—0,4	—0,6	0,2	0,4	0,6	0	0	1,9	275,5
2		$-x_{11}$	$-x_{12}$	$-y_3$	$-x_{21}$	$-x_{22}$	$-x_{23}$	$-x_{31}$	$-x_{32}$	$-y_1$	1
	x_{33} y_2 x_{13}			—1 2 2							18,333333 21,666666 11,666666
	Φ	0,8	0,4	1,2	1	0,8	0,6	0,8	0,4	1,7	282,5

Обозначив символами x_{ij} число автомобилей i-го типа, предназначенных для работы j смен в день, составим целевую функцию

$$\Phi = 2{,}1x_{11} + 4{,}2x_{12} + 6{,}3x_{13} + 3{,}6x_{21} + 7{,}2x_{22} + 10{,}8x_{23} + 3{,}8x_{31} + 7{,}6x_{32} + 11{,}4x_{33}$$

при ограничениях

$$x_{11} + 2x_{12} + 3x_{13} + 2x_{21} + 4x_{22} + 6x_{23} + 2x_{31} + 4x_{32} + 6x_{33} \leqslant 145;$$
$$5(x_{11} + x_{12} + x_{13}) + 10(x_{21} + x_{22} + x_{23}) + 12(x_{31} + x_{32} + x_{33}) \leqslant 300;$$
$$x_{11} + x_{12} + x_{13} + x_{21} + x_{22} + x_{23} + x_{31} + x_{32} + x_{33} \leqslant 30.$$

Составим по этим данным исходную симплекс-таблицу (табл. 18) и убедившись, что она соответствует опорному решению, выберем разрешающим элемент $a_{19} = 6$ и, выполнив шаг модифицированного жорданова исключения при помощи программируемого микрокалькулятора, получим новую симплекс-таблицу. Выбрав в ней разрешающим элемент $a_{33} = 0{,}5$, получим оптимальное решение (табл. 18), соответствующее полному использованию фонда заработной платы ($y_1^* = 0$) и допустимого количества автомобилей ($y_3^* = 0$), но неприемлемое в связи с дробностью оптимального числа автомобилей различных типов. Приняв ближайшие целые значения $x_{13}^* = 12$ и $x_{33}^* = 18$, лежащие в области допустимых решений, получим значение целевой функции $\Phi = 280{,}8$ т · км при 144 водителях и остатке 24 тыс. руб. от покупки автомобилей.

Несмотря на универсальность симплекс-метода, при решении многих задач линейного программирования целесообразно использовать алгоритмы, не связанные с громоздким преобразованием таблиц. Примером может служить задача о назначениях, заключающаяся в таком распределении m видов работ между n исполнителями, при котором достигается максимальный показатель оптимальности. Решение этой задачи заключается в максимизации целевой функции

$$\Phi = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}x_{ij}$$

при ограничениях

$$x_{i1} + x_{i2} + \ldots + x_{im} = 1, \quad i = 1, 2, \ldots, n;$$
$$x_{1j} + x_{2j} + \ldots + x_{nj} = 1, \quad j = 1, 2, \ldots, m,$$

где (при $m \geqslant n$) c_{ij} — показатель качества (оптимальности) выполнения j-го вида работ i-м исполнителем, а $x_{ij} = 1$, если i-й исполнитель назначен на j-й вид работ и $x_{ij} = 0$ в противном случае.

Решение этой задачи, сводящееся к перебору суммарных показателей качества допустимых назначений при использовании симплекс-метода, достаточно громоздко даже с помощью микрокалькуляторов [11]. Однако, определяя допустимые сочетания алгоритмами метода обобщенных чисел [10], можно, например, составить следующую программу автоматического решения задачи о назначениях при небольших m.

Программа 114/34. Решение задачи о назначениях при $m \leqslant 3$

ИП7	ИП5	ИП3	ПП	32	ИП7	ИП2	ИП6	ПП	32
ИП8	ИП4	ИП3	ПП	32	ИП8	ИП1	ИП6	ПП	32
ИП9	ИП4	ИП2	ПП	32	ИП9	ИП5	ИП1	ПП	32
ИПД	С/П	+	+	ПА	ИПС	1	+	ПС	ИПА
ИП0	—	$x \geqslant 0$	48	ИПА	П0	ИПС	ПД	В/О	

Инструкция: (c_{11}= *P7*, c_{12}= *P8*, c_{13}= *P9*, c_{21}= *P4*, c_{22}= *P5*, c_{23}= = *P6*, c_{31}= *P1*, c_{32}= *P2*, c_{33}= *P3*), 0 = *PO* = *PC* В/О С/П *PX* = = *PД* = k, *PA* = max Φ, где k — номер оптимального сочетания назначений в последовательности (x_{11}, x_{22}, x_{33}), (x_{11}, x_{23}, x_{32}), (x_{12}, x_{21}, x_{33}), (x_{12}, x_{23}, x_{31}), (x_{13}, x_{21}, x_{32},) (x_{13}, x_{22}, x_{31}).

При $m = 2$ вместо элементов c_{1j} следует занести единицы. При $3 < m \leqslant 8$ для перебора допустимых сочетаний назначений можно воспользоваться программой 53/34, определив после этого суммарный показатель качества для каждого допустимого сочетания и выбрав сочетание с максимальным показателем качества.

3. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Математическое программирование называют *нелинейным*, если целевая функция (8.1) или левые части хотя бы одного ограничения нелинейны или полилинейны (содержат произведения аргументов) относительно оптимизируемых переменных. В качестве целевых при нелинейном программировании обычно выбирают неотрицательные вещественные функции, минимум которых соответствует максимуму обратной функции, что позволяет свести задачи поиска экстремума к задачам минимизации целевой функции.

Методы нелинейного программирования используют, в частности, для решения систем уравнений, выбирая в качестве целевой функции сумму квадратов или модулей невязок уравнений. Эти же методы, как показано далее, применимы и для поиска корней уравнений высокой степени.

Задача минимизации (называемая при отсутствии ограничений *безусловной* оптимизацией) существенно усложняется при сложной форме поверхностей равного уровня целевой функции и ограничений. Если эту задачу не удается решить аналитически (например, по корням производных целевой функции), то прибегают к численным методам нелинейного программирования.

В пространстве n оптимизируемых переменных любой точке X соответствует значение целевой функции $\Phi(X)$ и направленный в эту точку из начала координат вектор $X = [x_1, x_2, \ldots, x_n]^t$, образованный проекциями точки на координатные оси. Две точки пространства X_k и X_{k+1} связаны векторным соотношением $X_{k+1} = = X_k + A_k = X_k + a_k\xi_k$, где направление вектора $A_k = X_{k+1} - X_k$ длиной $a_k = |A_k|$ определяется единичным вектором $\xi_k = A_k/|A_k|$. Приращения длины вектора связаны с приращениями переменных линейными выражениями $\Delta x_{ik} = \lambda_{ik}\Delta a_k = \Delta a_k \cos\varphi_{ik}$, где φ_{ik} —угол между положительным направлением i-й координатной оси и направлением вектора ξ_k.

Большинство численных методов нелинейного программирования [5, 14, 18] сводится к выбору на каждой k-й итерации вектора ξ_k направления и *одномерного поиска* из точки X_k минимума целевой функции в этом направлении. Последовательность таких итераций приводит в конечном итоге к локальному минимуму целевой функции, а различия между методами минимизации в основном связаны с мето-

дикой выбора направления и одномерного поиска на каждой итерации.

Существенное значение имеет выбор начальной точки X_0, определяющий время и, при многоэкстремальности целевой функции, результат оптимизации. Так как не существует общего подхода к определению точки X_0, то ее выбирают случайно или аналитически исследуют функцию в области допустимых решений, грубо определяя область нахождения минимума или, по крайней мере, монотонного изменения функции и выбирая начальную точку на границе этой области.

Одномерный поиск в общем случае состоит из этапа определения начального интервала, в котором находится оптимальное значение a_k^*, соответствующее минимуму целевой функции в выбранном направлении, и этапа сокращения ширины этого интервала до значения,

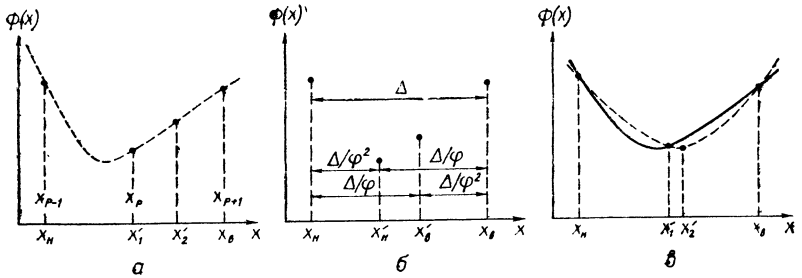


Рис. 42

определяемого требуемой или достижимой точностью оптимального решения X^*. В простейшем случае поиска оптимума вдоль координатной оси задача численной оптимизации сводится к минимизации функции одной переменной $\Phi(x)$.

При минимизации такой функции после выбора начальной точки x_0 вычисляют значения $\Phi(x_j)$ для ряда значений аргумента x_j с постоянным или возрастающим шагом. В последнем случае очередной шаг $\Delta x_j = x_j - x_{j-1} = \varphi \Delta x_{j-1}$ часто определяют умножением предыдущего шага на число $\varphi = 1 + 1/\varphi \approx 1{,}618034$, называемое *золотым сечением*. Поиск прекращают на $(p+1)$-м шаге при возрастании функции в соответствии с неравенствами

$$\ldots > \Phi(x_{p-1}) > \Phi(x_p) < \Phi(x_{p+1}) \qquad (8.3)$$

и в качестве интервала нахождения оптимума выбирают $[x_{p-1}, x_{p+1}]$, так как при выборе меньшего интервала $[x_p, x_{p+1}]$ возможна ошибка (рис. 42, *а*).

Методы сокращения интервала оптимума в основном относятся к одной из двух групп методов — последовательного деления и последовательного поиска. В методах *последовательного деления* на каждой итерации в сокращаемом интервале с нижней $x_н$ и верхней $x_в$ границами и внутренней точкой x'_1 выбирают вторую точку x'_2,

причем внутренние точки в зависимости от выполнения неравенства $x_2' > x_1'$ обозначают $x_н'$ и $x_в'$. Если выполняется условие

$$\Phi(x_н') \geqslant \Phi(x_в'), \tag{8.4}$$

то оптимум находится в сокращенном интервале $[x_н, x_в']$, а при невыполнении этого условия — в интервале $[x_н', x_в]$. Подобные итерации продолжают до получения решения с требуемой или достижимой точностью.

В простейшем методе *половинного* деления на каждой итерации в сокращаемом интервале с внутренней точкой x_1' (рис. 42, *а*) проверяют условие $\Phi(x_н) \geqslant \Phi(x_в)$, при выполнении которого выбирают $x_2' = x_в' = x_1' + (x_в - x_1')/2$ и $x_1' = x_н'$, а при невыполнении вычисляют $x_2' = x_н' = x_н + (x_1' - x_н)/2$ и принимают $x_1' = x_в'$. После этого сокращают интервал по условию (8.4).

В процессе численной оптимизации с помощью ЭВМ часто используют метод *золотого сечения*, при котором на каждой итерации в сокращаемом интервале (рис. 42, *б*) определяют точки $x_н' = x_н + (x_в - x_н)/\varphi^2 = x_в - (x_в - x_н)/\varphi$ и $x_в' = x_н + (x_в - x_н)/\varphi = x_в - (x_в - x_н)/\varphi^2$, где $\varphi = 1{,}618034$. После сокращения интервала по условию (8.4) сохранившаяся внутренняя точка соответствует одной из требуемых и по приведенным формулам дополнительно необходимо определить лишь одну точку.

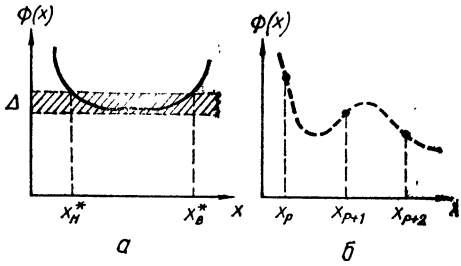
Вблизи минимума аналитические функции, как следует из разложения в ряд Тейлора, близки к квадратичным. Поэтому при небольших интервалах нахождения оптимума эффективен метод *квадратичной аппроксимации* на каждой итерации минимизируемой функции по трем точкам $x_н$, x_1', $x_в$ с вычислением второй внутренней точки сокращаемого по условию (8.4) интервала согласно формуле

$$x_2' = -\frac{(x_1'^2 - x_в^2)\Phi(x_н) + (x_в^2 - x_н^2)\Phi(x_1') + (x_н^2 - x_1'^2)\Phi(x_в)}{(x_1' - x_в)\Phi(x_н) + (x_в - x_н)\Phi(x_1') + (x_н - x_1')\Phi(x_в)},$$

определяющей минимум аппроксимирующей квадратичной функции (рис. 42, *в*).

В методах *последовательного поиска* после выбора начального значения x_0 отыскивают с постоянным или изменяющимся шагом по условию (8.3) границу интервала x_1, после чего продолжают поиск в противоположном направлении с меньшим шагом другой границы интервала. Продолжая подобные итерации сокращения интервала, заканчивают вычисления, когда значение шага Δx_k становится меньше наперед заданного малого числа ε. Это объясняется тем, что условие прекращения поиска оптимума $|x_{вk} - x_{нk}| \leqslant \varepsilon$ невыполнимо, когда изменения функции вблизи минимума меньше погрешности ее вычисления (рис. 43, *а*). В этом же случае прекращение вычислений по условию $|\Delta x_k| \leqslant \varepsilon$ обеспечивает вычисление границ $x_н^*$ и $x_в^*$ интервала оптимума с погрешностью Δx_k, а положение минимума — с погрешностью $(x_в^* - x_н^*)/2$.

В простейшем методе последовательного равномерного поиска на каждой итерации после определения с постоянным шагом Δx_k границы x_k интервала принимают шаг $\Delta x_{k+1} = -\Delta x_k/d$ (где делитель шага $d > 1$) и продолжают вычисления до выполнения условий их прекращения. При выборе больших значений d уменьшается шаг поиска и для поиска границы интервала требуется большее число шагов, но ширина интервала на k-й итерации оказывается меньшей. Кроме того, при выборе меньших значений d (и больших шагов) возможен пропуск острых минимумов целевой функции (рис. 43,*б*). Поэтому выбор начального шага и делителя d зависит от характера изменения функции (который заранее не всегда известен) — если функция изменяется медленно, то можно выбирать большие значения Δx_0 и меньшие значения d, но при быстрых изменениях функции Δx_0 следует выбирать меньше, а значение d большим.



а *б*

Рис. 43

Для минимизации функции одной переменной с помощью непрограммируемого микрокалькулятора целесообразно использовать методы последовательного деления, требующие меньшего числа вычислений функции. Если затраты времени на вычисление функции небольшие, то целесообразно начинать поиск методом половинного деления, выбирая по мере приближения к минимуму точки деления в соответствии с накапливаемой информацией о поведении функции. Если вычисления функции связаны со значительными затратами времени, то целесообразно начинать поиск методом золотого сечения, переходя по мере приближения к минимуму к методу квадратичной аппроксимации. При расчетах на программируемом микрокалькуляторе можно автоматизировать минимизацию функции одной переменной любым из рассмотренных методов деления или последовательного поиска. Если известен начальный интервал аргумента функции одной переменной, содержащий ее минимум, то уточнение этого интервала можно выполнить методом деления по золотому сечению, часто используемому при программировании универсальных ЭВМ.

Программа 115/34. Уточнение положения минимума функции $\Phi(x)$ одной переменной методом деления по золотому сечению

ИП7 П8 ИП9 ИП8 — П7 ИП4 — $x < 0$ 12
ИП6 С/П ИП7 ИП5 ÷ ИП8 + П3 ИП7 ИП5
x^2 ÷ ИП8 + П7 ПП 39 П6 ИП3 ПП
39 ИП6 — $x \geqslant 0$ 00 ИП3 П9 БП 02 ...
В/О

Инструкция: заменить в программе многоточие операторами вычисления при $x = P7$ функции $\Phi(x)$; $2 \cdot 10^{-7} < \varepsilon = P4$, $(\sqrt{5}+1)/2 =$

$= P5$, $x_н^{(0)} = P7$, $x_в^{(0)} = P9$ В/О С/П $PX = P4 = \Phi(x_н^*)$, $P8 = x_н^*$, $P9 = x_в^*$.

Для функции $\Phi(x) = e^x - 3x$ при $\varepsilon = 1 \cdot 10^{-4}$, $x_н^{(0)} = 0$, $x_в^{(0)} = 2$ по этой программе получим $\Phi(x_н^*) = -0{,}295837$; $x_н^* = 1{,}0983511$; $x_в^* = 1{,}0984329$ (время счета около 6 мин), при максимальной точности ($\varepsilon = 2 \cdot 10^{-7}$) получим $\Phi(x_н^*) = -0{,}2958371$; $x_н^* = 1{,}0983786$; $x_в^* = 1{,}0983787$ (время счета около 14 мин).

Если обе границы начального интервала нахождения оптимума не известны, то целесообразно использовать метод последовательного равномерного поиска, совмещающий поиск начального интервала и его уточнение.

Программа 116/34. Минимизация функции $\Phi(x)$ одной переменной методом последовательного равномерного поиска

ИП7 ИП8 + П7 . . .ИП4 ХУ П4 — $x < 0$
00 ИП8 /—/ d ÷ П8 x^2 ИП9 — $x < 0$
00 ИП4 С/П БП 00

Инструкция: заменить в программе символ d оператором набора делителя шага (обычно $d = 2, 3, \ldots, 9$), многоточие — операторами вычисления (при $x = P7$) минимизируемой функции, $\Phi(x_0) = P4$ (вместо этого можно принимать $1 \cdot 10^{99} = P4$), $1 \cdot 10^{-14} \leqslant (\varepsilon/d)^2 = P9$, $x_0 = P7$, $\Delta x_0 = P8$ В/О С/П $PX = P4 = \Phi(x^*)$, $P7 = x^*$, $P8 = \Delta x^*/d$.

После выполнения программы целесообразно пустить ее еще раз нажатием клавиши С/П, что обеспечит вычисление второго граничного значения интервала оптимума. При медленных изменениях функции в окрестностях оптимума может оказаться вычисленным не граничное значение x^*, а некоторое значение аргумента внутри сокращенного интервала, в котором мантисса функции вычисляется с погрешностью порядка единицы последнего разряда. Так, для функции $\Phi(x) = e^x - 3x$ при $d = 5$, $(\varepsilon/5)^2 = 1 \cdot 10^{-14}$, $x_0 = 0$, $\Delta x_0 = 0{,}5$ по программе 116/34 получим $\Phi(x^*) = -0{,}2958369$; $x^* = 1{,}0984458$ и $\Delta x^*/5 = 0{,}512 \cdot 10^{-8}$ (время счета около 4 мин). Вычисленное значение x^* отличается от полученных по предыдущей программе, но также верное, так как в интервале аргумента [1,0984; 1,099] вычисляемые микрокалькулятором значения функции флюктуируют в пределах двух единиц последнего разряда мантиссы (см. рис. 43, *а*). Эта программа, обеспечивающая максимальное число свободных ячеек программной памяти для записи операторов вычисления функции, применима и для поиска максимума — для этого достаточно заменить оператор $x < 0$ по адресу 09 оператором $x \geqslant 0$ и принять $\Phi(x_0) = 0$.

В качестве примера рассмотрим задачу о строительстве дороги минимальной стоимости между пунктами A и B (рис. 44) при условии, что $a = 100$ км, $b = 200$ км, в зонах 1 и 2 стоимость одного километра дороги с учетом эксплуатационных затрат составляет соответственно $c_1 = 10$ и $c_2 = 40$ тыс. руб., а граница между зонами описывается функцией $y = a^2/(x + a)$.

Так как при заданных условиях стоимость строительства дороги

$$C = c_1 \sqrt{x^2 + y^2} + c_2 \sqrt{(b - x)^2 + (a - y)^2},$$

то в качестве целевой выберем функцию

$$\Phi = \sqrt{x^2 + y^2} + 4\sqrt{(b-x)^2 + (a-y)^2},$$

приняв допустимую погрешность вычисления оптимального значения x^*, соответствующего наименьшей стоимости строительства, не более 10 м.

При $d = 5$, $(\varepsilon / d) = 4 \cdot 10^{-6}$, $x_0 = 100$ км, $\Delta x_0 = 20$ км, $\Phi(x_0) = 560$ с помощью программы 116/34 получим (время счета около 5 мин) $\min\Phi = 452{,}21884$; $x^* = 175{,}3472$; $y^* = 36{,}387783$ и $\Delta x^* = 1{,}28 \cdot 10^{-3}$, что соответствует длине трассы $l^* = 247{,}35628$ км при стоимости $C_{\min} = 4522188{,}4$ руб. По сравнению с прямолинейной ($l = 223{,}60679$ км; $C = 55901695$ руб.) оптимальная трасса обеспечивает выигрыш в 1067,9811 тыс. руб.

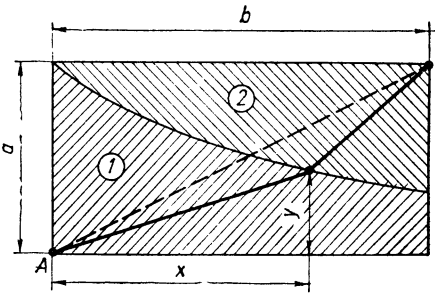


Рис. 44

В общем случае минимизации функций нескольких переменных затраты времени зависят от удачного выбора начальной точки X_0 и стратегии поиска, так как не существует универсальных алгоритмов, в равной мере пригодных для минимизации любых функций. При расчетах на микрокалькуляторах среди разнообразных методов нелинейного программирования [5, 13, 18] приходится выбирать простейшие из них, отличающиеся несложной для реализации стратегией поиска и минимальным объемом вычислений на каждой итерации. К таким методам минимизации функций нескольких переменных относятся скорейший и координатный спуски.

При *скорейшем* спуске на каждой k-й итерации для точки X_k вычисляют производные $\Phi_i' = \partial\Phi/\partial x_i$ по направлениям координатных осей и выполняют одномерный поиск оптимума в направлении антиградиента $-\nabla = -[\Phi_1', \Phi_2', \ldots, \Phi_n']$, нормального поверхностям равного уровня целевой функции и направленного в сторону ее уменьшения. Длина шага Δa в этом направлении соответствует приращениям оптимизируемых переменных $\Delta x_i = \lambda_i \Delta a =$

$$= \Delta a\,|\,\Phi_i'\,|/|\,\nabla\,| = \Delta a\,|\,\Phi_i'\,|\Big/\sqrt{\sum_{i=1}^{n}\Phi_i'}.$$

В методе *координатного* спуска одномерный поиск локального оптимума выполняют на каждой итерации по одной из координатных осей, выбором которых отличаются различные модификации этого метода. Координатный спуск может быть выполнен без вычисления производных и, следовательно, применим для минимизации недифференцируемых функций.

Однако простейшие методы скорейшего и координатного спуска в их традиционной форме непригодны для минимизации функций с крутыми оврагами [5] — областями пространства переменных с вытянутыми линиями равного уровня. Поэтому для составления универсальной базовой программы с максимальными возможностями целесообразно модифицировать метод координатного спуска, автоматизи-

руя, например, на каждой итерации лишь поиск границы интервала оптимума одной переменной по условию (8.3) и переход к следующей переменной. При таком неполном координатном спуске пользователь микрокалькулятора выбирает наилучшую стратегию (изменяя при необходимости значения шага поиска и переменных перед каждой итерацией) в соответствии с накапливаемой в процессе вычислений информацией. В частности, возрастание функции после выполнения только одного шага согласно условию (8.3) свидетельствует о целесообразности повторения поиска минимума той же переменной при изменении знака шага.

Программа 117/34. Минимизация функции $\Phi(X) = \Phi(x_1, \ldots, x_n)$ с $n \leqslant 10$ переменными методом неполного координатного спуска

```
КИПС ИПД  +   КПС ...  ИП0 XY П0  —  x<0
00   ИП0 С/П  ИПВ ИПС 1    +   ПС  —  x<0
00   1   ПС   БП  00
```

Инструкция: заменить в программе многоточие операторами вычисления (при $x_i = Pi$) минимизируемой функции, $1 \cdot 10^{99} = P0$, $n = PB$, $1 = PC$, $\Delta x = PД$, $x_i = Pi$; для оптимизации i-й переменной ($PC = i$), изменив при необходимости Δx или x_i нажимать клавиши В/О и С/П, для перехода к оптимизации следующей переменной (или x_1 после x_n) нажать клавишу С/П; результаты $PX = \Phi(X^{(k)})$, $Pi = x^{(k)}$.

При небольшом числе оптимизируемых переменных для ускорения вычислений целесообразно составлять программу без операторов косвенной адресации, подобную следующей.

Программа 118/34. Минимизация функции $\Phi(X) = \Phi(x_1, x_2)$ методом неполного координатного спуска

```
ИП7 ИП9 +  П7 ПП 20   x<0 00 ИП4 С/П
ИП8 ИП9 +  П8 ПП 20   x<0 10 БП  08
...  ИП4 XY П4 —  В/О
```

Инструкция: заменить в программе многоточие операторами вычисления (при $x_1 = P7$, $x_2 = P8$) минимизируемой функции; $1 \cdot 10^{99} = P4$, $x_1 = P7$, $x_2 = P8$, $\Delta x = P9$; для оптимизации x_1 нажимать клавиши В/О и С/П; для оптимизации x_2 после x_1 нажимать только клавишу С/П; результаты: $PX = \Phi(X^{(k)})$, $P7 = x_1$, $P8 = x_2$.

Для проверки различных алгоритмов численной оптимизации при программировании универсальных ЭВМ в качестве тестовой используют функцию Розенброка

$$\Phi(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

при начальных значениях $X_0 = [x_1 \; x_2] = [-1{,}2 \; 1]$ [18].

Заменив в программе 118/34 многоточие операторами вычисления этой функции, выполним поиск ее минимума в соответствии с шагами, указанными в табл. 19. Как следует из данных, приведенных в этой таблице, удалось достаточно быстро спуститься вдоль крутого оврага функции Розенброка и выйти в область ее минимума, изменяя

19. Минимизация функции Розенброка последовательным поиском границ интервалов оптимума переменных

k	$\Delta x_k = P9$	Пуск	$PX = \Phi(X_k)$		$P1 = x_{1k}$	$P2 = x_{2k}$
0	—	—	1000	—	—1,2	1
1	0,2	В/О С/П	16,2	—	—0,8	1
2	—	С/П	—	34,6	—	1,2
3	—0,2	С/П	—	9	—	0,4
4	0,2	В/О С/П	7,72	—	—0,4	—
5	—0,2	С/П	—	4,52	—	0
6	0,2	В/О С/П	2,92	—	0,4	—
7	—	С/П	—	6,12	—	0,4
8	—	В/О С/П	5,8	—	0,8	—
9	—	С/П	—	2,6	—	0,8
10	—	В/О С/П	4	—	1	—
11	—	С/П	—	4	—	1,2
12	—0,02	С/П	—	0,04	—	0,98
13	0,001	С/П	—	0,0001	—	1,001
14	—0,0001	С/П	—	$1 \cdot 10^{-6}$	—	0,9999
15	0,00001	С/П	—	$1 \cdot 10^{-8}$	—	1,00001

только знак шага при слежении оврага и уменьшая модуль шага при приближении к минимуму.

Уровень автоматизации поиска повышается при использовании различных модификаций координатного спуска в соответствии со свойствами минимизируемых функций и надлежащем выборе условий перехода к оптимизации следующей переменной и прекращения вычислений. Полную автоматизацию достаточно гладких (в частности, выпуклых* [18]) функций обеспечивает, например, следующая программа.

Программа 119/34. Минимизация гладких функций $\Phi(X)$ с числом $n \leqslant 9$ переменных методом координатного спуска

КИПВ	ИПД	+	КПВ	...	ИП0	XY	П0	XY	—
$x \geqslant 0$	00	ИПД	/—/	3	÷	ПД	x^2	ИПС	8
÷	x^2	—	$x < 0$	00	ИПС	ПД	п	ИПВ	1
+	ПВ	—	$x < 0$	00	1	ПВ	ИПА	ИПС	9
÷	ПС	ПД	x^2	—	$x \geqslant 0$	00	ИП0	С/П	

Инструкция: заменить в программе символ n и многоточие соответственно операторами набора числа n переменных и вычисления (при $x_i — Pi$) минимизируемой функции; $1 \cdot 10^{99} = P0$,$(\varepsilon/9)^2 = PA$, $1 = PB$, $\Delta x = PC = PД$, $x_i^{(0)} = Pi$ В/О С/П $PX = P0 = \Phi(X^*)$, $Pi = x_i^*$.

---

* *Выпуклыми* называют функции, поверхности равных уровней которых пересекаются любой прямой не более чем в двух точках.

В качестве тестовой для проверки программы выберем функцию [18]

$$\Phi(X) = \Phi(x_1, \ldots, x_n) = \exp\left(\sum_{i=1}^{n} x_i\right) / \prod_{i=1}^{n} x_i^i \qquad (8.5)$$

с ограничениями $x_i > 0$ и оптимумом $x_i = i$. При $n = 3$, $\varepsilon = 1 \cdot 10^{-4}$, $x_i^{(0)} = \Delta x = 0{,}5$ получим $\Phi(X^*) = 3{,}7354514$; $x_1^* = 1{,}000254$; $x_2^* = 2{,}0002288$; $x_3^* = 3{,}0000256$ (время счета около 21 мин).

Время минимизации гладких функций и длину базовой части программы можно уменьшить при вычислениях на каждой итерации с постоянным шагом одной границы оптимума каждой переменной и выборе $\Delta x = -\Delta x/d$ для следующей итерации. В пространстве оптимизируемых переменных траектория поиска оптимума в этом случае напоминает сходящуюся спираль.

Программа 120/34. Минимизация выпуклых функций $\Phi(X)$ с числом $n \leqslant 10$ переменных методом спирального координатного спуска

КИПС	ИПД	+	КПС	...	ИП0	XY	П0	XY	—
$x \geqslant 0$	00	n	ИПС	1	+	ПС	—	$x < 0$	00
1	ПС	ИПВ	ИПД	/—/	5	÷	ПД	x^2	—
$x \geqslant 0$	00	ИП0	С/П						

Инструкция: заменить в программе многоточие и символ соответственно операторами вычисления (при $x_i = Pi$) минимизируемой функции и набора числа n переменных; $1 \cdot 10^{99} = P0$, $(\varepsilon/5)^2 = PB$. $1 = PC$, $\Delta x = PД$, $x_i^{(0)} = Pi$ В/О С/П $PX = P0 = \Phi(X^*)$, $Pi = x_i^*$.

После ввода базовой части этой программы сохраняется 65 (вместо 50 для предыдущей программы) ячеек программной памяти, что повышает допустимую сложность минимизируемой функции. В частности, по этой программе возможна минимизация функции (8.5) с $n \leqslant 9$ вместо $n \leqslant 7$ по предыдущей программе. При $n = 3$, $\varepsilon = 1 \cdot 10^{-4}$, $\Delta x = x_i^{(0)} = 0{,}5$ по программе 120/34 для функции (8.5) получим $\Phi(X^*) = 3{,}7354523$; $x_1^* = 1{,}000672$; $x_2 = 2{,}000672$; $x_3 = 3{,}000352$ (время счета около 16 мин).

В тех случаях, когда процесс поиска границ интервала оптимума не удается автоматизировать в связи с ограниченной емкостью запоминающих устройств микрокалькулятора, следует ограничиться автоматизацией вычисления только минимизируемой функции. При использовании программируемых микрокалькуляторов с бóльшими быстродействием и емкостью запоминающих устройств, чем у микрокалькуляторов с входными языками ЯМК21 и ЯМК34, возможности автоматизации нелинейного программирования существенно расширяются даже при использовании только рассмотренных методов.

4. МИНИМИЗАЦИЯ МНОГОЭКСТРЕМАЛЬНЫХ ФУНКЦИЙ

Задача численной оптимизации усложняется при нескольких минимумах целевой функции. Если требуется найти оптимальные значения переменных, соответствующие любому из локальных миниму-

мов, то применим любой из рассмотренных методов нелинейного программирования. Однако обычно задача минимизации многоэкстремальных функций заключается в отыскании всех ее минимумов, из которых часто требуется выбрать глобальный (наименьший из всех) минимум или минимум, удовлетворяющий дополнительному условию оптимальности решения задачи.

Рассмотренные методы обеспечивают поиск локального минимума, и для отыскания следующего минимума нужно выбрать новую начальную точку. Если она выбрана случайным образом, то процесс минимизации может привести к ранее найденному минимуму и, следовательно, бесполезным затратам времени. Кроме того, в этом случае общее число минимумов и критерий прекращения минимизации оказываются неопределенными.

Для последовательного поиска всех минимумов многоэкстремальной функции необходимо предварительно оценить их количество и области расположения. Эту задачу можно решить, построив в пространстве переменных поверхности равного уровня целевой функции, но такой подход связан со значительными затратами времени и практически возможен лишь при $n \leqslant 3$. Часто полезные сведения о свойствах функции удается получить методами математического анализа, особенно при решении методами нелинейного программирования уравнений или систем уравнений.

В качестве примера рассмотрим несложную систему уравнений

$$x_1^2 + x_1^3 = 1,\ x_1^3 - x_2 = 0,$$

решение которой обычными методами, рассмотренными в гл. 5, затруднено в связи с необходимостью выбора начального приближения вблизи корней, расположение которых неизвестно.

Первое уравнение удовлетворяется при любых знаках переменных, тогда как второе справедливо лишь при их одинаковых знаках, и, следовательно решения уравнения будут симметричны относительно координатных осей. Исключив x_2 из первого уравнения, составим целевую функцию по его невязке

$$\Phi = (x_1^2 + x_2^2 - 1)^2.$$

Заменив в программе 116/34 многоточие операторами x^2 ИП7 ↑ x^2 × ↑ × 1 — x^2 и приняв $x_t^{(0)} = 0$, $\Delta x = 0{,}2$; $\Phi(x_t^{(0)}) = 1 \cdot 10^{99}$, $(\varepsilon/2)^2 = 2{,}5 \cdot 10^{-11}$ (что соответствует пяти верным цифрам значения корня), получим $x_1^* = 0{,}82602542$; $\Phi(x_1^*) = 5{,}5225 \cdot 10^{-10}$; $\Delta x^*/2 = 3{,}05 \cdot 10^{-6}$ (время счета около 3 мин). Вычислив в обычном режиме $x_2^* = (x_1^*)^3$ и округлив результаты до верных цифр, с учетом симметрии находим оба решения: $X^* = [0{,}82603\ \ 0{,}56361]$ и $X^* = [-0{,}82603\ \ -0{,}50361]$.

Однако часто методами математического анализа удается лишь грубо оценить границы монотонного изменения функции в пространстве переменных и экстремальные области приходится находить и разделять численными методами. Такой прямой поиск при использовании микрокалькуляторов целесообразно реализовать наиболее экономными алгоритмами и программами с минимальным числом базовых операторов для организации поиска. Этому требованию отвечает равномерный последовательный поиск с постоянным шагом Δx_1 минимумов и мак-

симумов исследуемой функции в интервалах $[x_{1н}, x_{1в}]$ одной переменной x_1 при фиксированных значениях остальных переменных. Многократный повторный поиск при изменении этих переменных на величины Δx_i и, при необходимости, изменении шага поиска и границ интервала x_1 обеспечивает приближенное определение границ экстремальных областей.

При вычислениях на программируемых микрокалькуляторах подобный поиск целесообразно организовать с помощью цикла с поочередным определением минимумов по условию (8.3) и максимумов по обратному условию с остановкой выполнения программы для регистрации полученных данных и при достижении границы заданного интервала $x_в$. Так как при равномерном поиске с шагом Δx_1 значение экстремума находится в интервале шириной $2\Delta x_1$ (см. рис. 42, *а*), то такой поиск обеспечивает определение экстремумов с погрешностями Δx_i по каждой переменной.

Программа 121/34. Одномерный поиск экстремальных областей функции $\Phi(x)$ нескольких переменных ($n \leqslant 10$)

```
ПП 12  x⩾0 00  С/П ПП  12  x<0 05  С/П
БП 00  ИПД ИПВ —  x⩾0 09  ИПВ ИПС +
ПВ ... ИП0 XY  П0  —  В/О
```

Инструкция: заменить в программе многоточие операторами вычисления (при $x_1 = P1$ и использовании регистров 2, ..., 9, *A*) исследуемой функции; перед каждым пуском программы ввести $x_{1н} = PB$, $\Delta x_1 = PC$, $x_{1в} = PД$, при необходимости изменить значения переменных, заносимых в выбранные регистры перед первым пуском программы, $1 \cdot 10^{99} = P0$ (при возрастании функции для $x_1 < x_{1н}$) (В/О) С/П $PX = \Delta\Phi < 0$, $P0 = \Phi_{min}$, $PB = x_{1(min)}$ С/П $PX = \Delta\Phi > 0$, $P0 = \Phi_{max}$, $PB = x_{1(max)}$ С/П $PX = \Delta\Phi < 0$, $P0 = \Phi_{min}$, $PB = x_{1(min)}$ ... С/П $PX = x_{1в} - x_1 \leqslant 0$.

Существенное значение при вычислениях по этой программе имеет выбор шага Δx_1 и приращений Δx_i. При малых значениях этих приращений возрастает время поиска, но при больших значениях возможен пропуск экстремальной области (см. рис. 43, *б*). Уменьшая шаг Δx_1, можно оценить границы экстремума с более высокой точностью, но для точного определения оптимальных значений переменных для уменьшения затрат времени целесообразно воспользоваться одной из ранее рассмотренных программ, выполняя поиск вблизи найденного экстремума функции.

Затраты времени при исследовании функции по рассмотренному алгоритму уменьшаются при предварительном определении методами математического анализа возможно более точных границ области поиска.

В качестве примера исследуем функцию двух переменных [18]

$$\Phi(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2,$$

монотонно возрастающую, как легко установить, при $|x_1| \geqslant 5$ и $|x_2| \geqslant 5$. Записав в программу 121/34 вместо многоточия операторы вычисления этой функции (при $x_2 = F2$), при $x_{1н} = -5$, $x_{1в} = 5$,

$x_2 = -5$, $\Delta x_1 = 0{,}5$ получим по этой программе $x_{1(\min)} = -3{,}5$ (при $\Phi_{\min} = 224$, 3125), $x_{1(\max)} = 1$ ($\Phi_{\max} = 586$), $x_{1(\min)} = 4$ ($\Phi_{\min} = 484$). Повторяя поиск при изменении x_2 на $\Delta x_2 = 0{,}5$, получаем в выбранной области значения оптимумов $X_1 = [-4 \quad -3{,}5]^{\mathrm{T}}$, $X_2 = [4 \quad -2]^{\mathrm{T}}$, $X_3 = [3 \quad 1{,}5]^{\mathrm{T}}$, $X_4 = [-3 \quad 3]^{\mathrm{T}}$, соответствующих (с погрешностью 0,5) минимумам функции (рис. 45, *а*). Более точное значение оптимумов (рис. 45, *б*) целесообразно находить с помощью ранее рассмотренных программ минимизации.

Процесс численной оптимизации упрощается и затраты времени сокращаются при рациональном использовании имеющейся информации о свойствах минимизируемой многоэкстремальной функции. Рассмотренные в гл. 5 методы применимы для поиска с помощью мик-

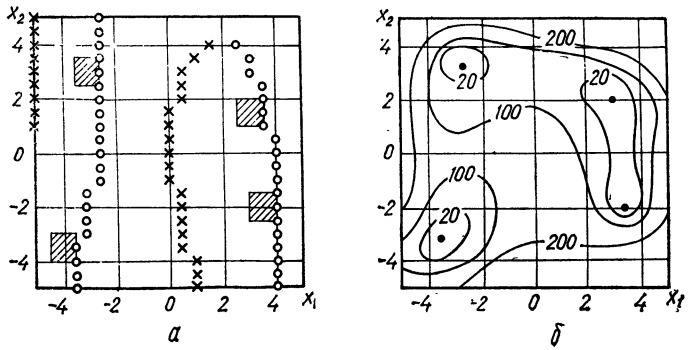


Рис. 45

рокалькулятора корней уравнений (5.5) степени $n \leqslant 7$, причем точность вычисленных корней оказывается низкой в связи с накоплением операционных погрешностей. Между тем невязка уравнений в корнях равна нулю и, следовательно, при использовании соответствующего степенного многочлена в качестве целевой функции корни алгебраических уравнений могут быть найдены с высокой точностью (определяемой лишь точностью вычисления многочлена) методами нелинейного программирования. Для этого нужно учесть основные свойства алгебраических многочленов, рассмотренные в гл. 5, из которых следует:

1. Для поиска всех корней уравнения $A(p) = 0$ достаточно ограничиться (см. рис. 37) областью плоскости $p = \sigma + j\omega$, ограниченной снизу вещественной осью (включая эту ось) и сверху полуокружностью радиуса $R = 1 + m_0/a_0$, где m_0— наибольший модуль коэффициентов многочлена, за исключением a_0, или любым контуром, охватывающим эту полуокружность.

2. Область поиска корней можно уточнить, дополнительно исследуя набег фазового угла при обходе по любому замкнутому контуру внутри области расположения корней (набег фазового угла $\varphi_\Sigma = 2\pi s$, где s — число корней внутри контура).

3. В качестве начальных точек поиска целесообразно выбрать точки пересечения линий равных фазовых углов $\varphi = 90° + k180°$ или $\varphi = 180° + k180°$ с контрольной линией, охватывающей сверху

область расположения корней. При обходе по контрольной линии в таких точках знак $\mathrm{Re}A\,(p)$ изменяется при сохранении знака $\mathrm{Im}\,A(p)$, что удобно использовать как для определения начальных точек поиска, так и для спуска к корням вдоль линий равных фаз $\varphi = 90° + k360°$ (k — целое число).

При учете этих замечаний для поиска корней алгебраического многочлена можно воспользоваться следующими программами.

Программа 122/34. Вычисление модуля, вещественной и мнимой частей алгебраического многочлена степени $n \leqslant 8$ с автоматическим изменением составляющих комплексного аргумента

ИПС	ИП9	+	ПС	ИП8	ИПД	×	ПВ	ИП8	ИПС
×	ИП7	ПП	47	ИП6	ПП	47	ИП5	ПП	47
ИП4	ПП	47	ИП3	ПП	47	ИП2	ПП	47	ИП1
ПП	47	ИП0	+	ПА	x^2	ИПВ	x^2	+	$\sqrt{}$
С/П	ИПД	ИП9	+	ПД	БП	04	+	ПА	ИПС
×	ИПД	ИПВ	×	—	ИПС	ИПВ	×	ИПД	ИПА
×	+	ПВ	XY	В/0					

Инструкция: ($a_0 = P0$, $a_1 = P1$, $a_2 = P2$, ..., $a_8 = P8$) $\Delta x = P9$, $\sigma_0 = PC$, $\omega_0 = PД$; для вычисления многочлена при хранящихся в памяти значениях $\sigma_i = \sigma_{i-1}$, $\omega_i = \omega_{i-1}$ нажать клавиши БП 0 4 С/П, для вычисления многочлена при значениях $\sigma_i = \sigma_{i-1} + \Delta x$, $\omega_i = \omega_{i-1}$ нажать клавиши В/О С/П; для вычисления многочлена при значениях $\sigma_i = \sigma_{i-1}$, $\omega_i = \omega_{i-1} + \Delta x$ при первом пуске программы нажать клавиши БП 4 1 С/П, при последующих пусках достаточно нажать клавишу С/П; после выполнения программы $PX = A(p)$, $PA = \mathrm{Re}\,A(p)$, $PB = \mathrm{Im}\,A(p)$, $PC = \sigma_i$, $PД = \omega_i$. Время счета около 60 с.

Эту программу несложно модифицировать для поиска корней нормированного ($a_9 = 1$) многочлена степени $n = 9$. Для этого достаточно начальный фрагмент из 11 операторов заменить более коротким фрагментом ИПД ПВ ИПС ПА ИП8 ПП 42 и изменить адреса остальных обращений к памяти на 42, а для вычисления многочлена при $\sigma_i = \sigma_{i-1}$ и $\omega_i = \omega_{i-1} + \Delta x$ при первом пуске программы нажимать клавиши БП 3 6 С/П.

Программу 122/34 можно использовать и при $n < 8$, но в этом случае часть времени выполнения программы будет бесполезно затрачиваться на операции над нулевыми коэффициентами при старших степенях аргумента. Поэтому при $n < 8$ целесообразно составлять программы для определенных значений n. При этом для уменьшения времени выполнения программы целесообразно изъять фрагмент вычисления модуля, так как о его величине можно судить по значениям $\mathrm{Re}A$ и $\mathrm{Im}A$. Кроме того, для уменьшения затрат времени целесообразно организовать хранение вычисленных значений $\mathrm{Re}A$ и $\mathrm{Im}A$ также в регистрах X и Y — в этом случае одно из этих значений высвечивается на индикаторе, а для вызова другого достаточно нажать клавишу XY. Примером такой программы может служить следующая.

Программа 123/34. Вычисление вещественной и мнимой частей алгебраического многочлена степени $n = 6$ с автоматическим изменением составляющих комплексного аргумента

ИПС	ИП9	+	ПС	ИП6	ИПД	×	ПВ	ИП6	ИПС
×	ИП5	ПП	35	ИП4	ПП	35	ИП3	ПП	35
ИП2	ПП	35	ИП1	ПП	35	ИП0	+	С/П	ИПД
ИП9	+	ПД	БП	04	+	ПА	ИПС	×	ИПД
ИПВ	×	—	ИПС	ИПВ	×	ИПД	ИПА	×	+
ПВ	XY	В/О							

Инструкция: ($a_0 = P0$, $a_1 = P1$, $a_2 = P2$, ..., $a_6 = P6$) $\Delta x = P9$, $\sigma_0 = PC$, $\omega_0 = PД$; для вычисления многочлена при $\sigma_i = \sigma_{i-1}$, $\omega_i = \omega_{i-1}$ нажать клавиши БП 0 4 С/П; для вычисления многочлена при $\sigma_i = \sigma_{i-1} + \Delta x$, $\omega_i = \omega_{i-1}$ нажать клавиши В/О С/П; для вычисления многочлена при $\sigma_i = \sigma_{i-1}$, $\omega_i = \omega_{i-1} + \Delta x$ при первом пуске программы нажать клавиши БП 2 9 С/П, при последующих пусках нажимать только клавишу С/П; после выполнения программы $PX = PA = \mathrm{Re}\, A$, $PY = PB = \mathrm{Im}\, A$, $PC = \sigma_i$, $PД = \omega_i$. Время счета около 40 с.

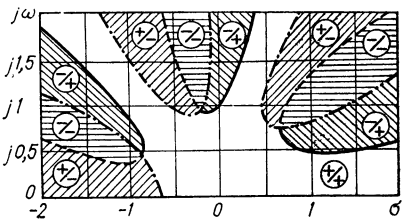


Рис. 46

При поиске корней с помощью подобных программ целесообразно в качестве контрольной линии для определения начальных точек выбрать верхнюю часть прямоугольника, описывающего полуокружность радиуса $R = 1 + m_0/a_0$ (рис. 46), и принять значение Δx таким, чтобы на контрольной линии укладывалось не менее $2n$ таких значений, но не слишком малым, чтобы не увеличивалось время поиска.

Приняв $\sigma_0 = R$, $\omega_0 = 0$, следует выполнять программу при $\omega_i = \omega_{i-1} + \Delta x$ до точки $\sigma_i = R$, $\omega_i = R$, после чего продолжить выполнение программы при $\sigma_i = \sigma_{i-1} + \Delta x$ (изменив знак Δx на отрицательный) до точки $\sigma_i = -R$, $\omega_i = R$ и, наконец, при $\omega_i = \omega_{i-1} + \Delta x$ — до точки $\sigma_i = -R$, $\omega_i = 0$. Значения $\mathrm{Re}\, A$ и $\mathrm{Im}\, A$ после каждого выполнения программы следует записывать в вычислительный бланк, что позволит определить интервалы $[\sigma_i, \sigma_{i-1}]$ или $[\omega_i, \omega_{i-1}]$, в которых изменяется знак $\mathrm{Re}\, A$ или $\mathrm{Im}\, A$ и, следовательно, пересекается линия равной фазы $\varphi = 90° + k\,90°$.

Выбрав интервалы, в которых пересекаются линии равных фаз $\varphi = 90° + k360°$ с контрольной линией (где знак $\mathrm{Re}A$ изменяется на отрицательный при сохранении положительного знака $\mathrm{Im}A$), следует провести поиск вблизи такого интервала параллельно ему внутри области нахождения корней с меньшим шагом для уточнения положения линии равной фазы и далее спускаться вдоль нее к корню методом координатного спуска, уменьшая шаг по мере уменьшения абсолютных значений $\mathrm{Re}A$ и $\mathrm{Im}A$.

Часто удается ускорить определение корней, исследовав поведение многочлена на дополнительных контрольных линиях внутри области нахождения корней. В частности, если при обходе по контрольной линии, включающей первую четверть окружности радиуса R (или описывающей ее линии) и отрезок положительной части мнимой оси, набег фазы $\varphi_\Sigma = 0$, то в этой области корни отсутствуют и, следовательно, исследуемый многочлен устойчив (все его корни находятся в левой полуплоскости).

В качестве примера на рис. 46 показаны контрольная прямоугольная линия и линии равных фаз $\varphi = 90° + k90°$ многочлена $p^6 + p^5 + p^4 + p^3 + p^2 + p + 1$, а вычислительный бланк для поиска корней многочлена при помощи программы 123/34 приведен в табл. 20. Как следует из данных этой таблицы, на первых 20 итерациях (выполнениях программы) с шагом $\Delta x = 0{,}5$ пройдена контрольная линия для определения интервалов, в которых изменяются знаки $\mathrm{Re}A$ или $\mathrm{Im}A$. На следующих пяти итерациях пройден отрезок положительной полуоси мнимой части аргумента, причем оказалось, что в точке $\sigma = 0$, $\omega = 1$ эта контрольная линия пересекается линией равной фазы $\varphi = 90° + 360°$. На остальных итерациях из этой точки методом координат-

20. Поиск корня многочлена $A(p) = p^6 + p^5 + p^4 + p^3 + p^2 + p + 1$ с помощью программы 123/34

Исходные данные: $\sigma_0 = 2 = PC$, $\omega_0 = 0 = PД$, $1 = PO = P1 = \ldots = P6$

i	Пуск	Re A	Im A	σi	ωi
1	0,5 = Р9 БП 2 9 С/П	43,67	134,7	2	0,5
2	С/П	—154	125	2	1
3	С/П	—314,7	—125,1	2	1,5
4	С/П	—205	—614	2	2
5	—0,5 = Р9 В/О С/П	129,4	—265,7	1,5	2
6	В/О С/П	139	—14	1	2
7	В/О С/П	29,23	7,15	0,5	2
8	В/О С/П	—51	26	0	2
9	С/П	—7,578	5,718	0	1,5
10	С/П	0	1	0	1
11	—0,1 = Р9 В/О С/П	—0,192	0,546	—0,1	1
12	В/О С/П	—0,15	0,054	—0,2	1
13	С/П	—0,059	0,085	—0,2	0,98
14	—0,02 = Р9 В/О С/П	—0,025	0,0005	—0,22	0,98
15	—0,002 = Р9 С/П	—0,017	0,0043	—0,22	0,978
16	В/О С/П	—0,013	0,0038	—0,222	0,978
17	С/П	—0,0053	0,00002	—0,222	0,976
18	—0,0005 = Р9 В/О С/П	—0,0043	0,02002	—0,2225	0,976
19	—0,001 = Р9 С/П	—0,0049	0,00179	—0,2225	0,975
20	—0,0001 = Р9 С/П	0,00007	0,00013	—0,2225	0,9749
21	0,00005 = Р9 С/П	—0,00012	0,00004	—0,2225	0,97495
22	—0,00001 = Р9 В/О С/П	—0,00011	0,000001	—0,22251	0,97495
23	—0,00002 = Р9 С/П	—0,000029	0,00004	—0,22251	0,97493
24	—0,00001 = Р9 В/О С/П	—0,000010	—0,000000	—0,22252	0,97493
25	—0,000001 = Р9 В/О С/П	—0,000019	—0,000032	—0,222521	0,97493
26	—0,000002 = Р9 С/П	—0,000000	—0,000000	—0,222521	0,974928

ного спуска реализован поиск корня с невязкой модуля, меньшей $1 \cdot 10^{-6}$, что достаточно в большинстве случаев.

Рассмотренные программы можно использовать как для поиска корней алгебраических корней многочленов, так и для построения линий равных фаз и модулей (в частности, при построении силовых и эквипотенциальных линий электрических полей или линий тепловых потоков и изотерм при исследовании тепловых полей). При этом корни определяются независимо и точность их поиска ограничена только операционной погрешностью вычисления многочлена. Программу 122/34 можно использовать и для поиска корней алгебраических уравнений с многочленами степени $n = 9$, предварительно нормировав многочлен ($a_n = 1$) и соответственно изменив начало программы 122/3 и адреса переходов. Дополнительно повысить степень n многочлена можно, отказавшись от автоматизации изменения составляющих аргумента и вычисляя значения многочлена по программам, приведенным в настоящей книге, рассмотренным способом. Поиск всех корней может быть существенно ускорен при использовании программируемых микрокалькуляторов с большей емкостью памяти и большим быстродействием.

5. МИНИМИЗАЦИЯ ФУНКЦИЙ ПРИ ОГРАНИЧЕНИЯХ

Задача нелинейного программирования часто существенно усложняется при линейных и особенно нелинейных ограничениях. Обычная процедура минимизации целевых функций с r ограничениями-равенствами предусматривает выделение r свободных переменных (с подстановкой их значений, выраженных через остальные переменные, в целевую функцию), минимизацию функции $m = n - r$ переменных с последующим вычислением оптимальных значений свободных переменных по ограничениям-равенствам. Ограничения-неравенства обычно учитывают определением области допустимых решений, ограниченной поверхностями, на которых эти неравенства становятся равенствами, и поиском минимумов в этой ограниченной области.

В некоторых случаях для решения системы нелинейных уравнений, определяющих свободные переменные, как и для определения границ допустимой области решений, также приходится прибегать к громоздким методам нелинейного программирования. Иногда задачу минимизации функций с ограничениями удается свести к задаче безусловной минимизации, но выбор целевой функции при этом зависит от конкретных условий задачи [18] и приходится оценивать возможные затраты времени на выбор варианта решения задачи оптимизации.

На практике при решении задачи минимизации функций с ограничениями встречаются с многокритериальными условиями решения, когда для выбора оптимального решения, удовлетворяющего всем критериям качества, приходится находить компромисс между удовлетворением частных критериев оптимальности.

В качестве примера рассмотрим задачу изготовления с минимальными отходами железного листа заготовок для сварки прямоугольного контейнера без крышки со сторонами a, b и h объемом $v = abh = 1$ м^3. При отсутствии ограни-

чений на размеры листа эта задача сводится к безусловной минимизации площади $S = ab + 2h\,(a + b)$ при ограничении-равенстве $abh = 1$ на объем изготовленного контейнера. При подстановке свободной переменной $h = 1/ab$ задача сводится к безусловной минимизации функции двух переменных

$$S = ab + 2\,(a + b)\,/\,ab. \qquad (8.6)$$

В связи с симметрией этой функции относительно оптимизируемых переменных ее можно заменить функцией $S = x^2 + 4/x$ одной переменной $x = a = b$ при ограничении-равенстве $x^2h = 1$ для определения свободной переменной h. Введение в целевую функцию такого вида квадрата невязки ограничения-равенства приводит к задаче безусловной оптимизации размеров $x = a = b$ и h по целевой функции

$$S = x^2 + 4/x + (x^2h - 1)^2,$$

но поиск минимума в этом случае усложняется.

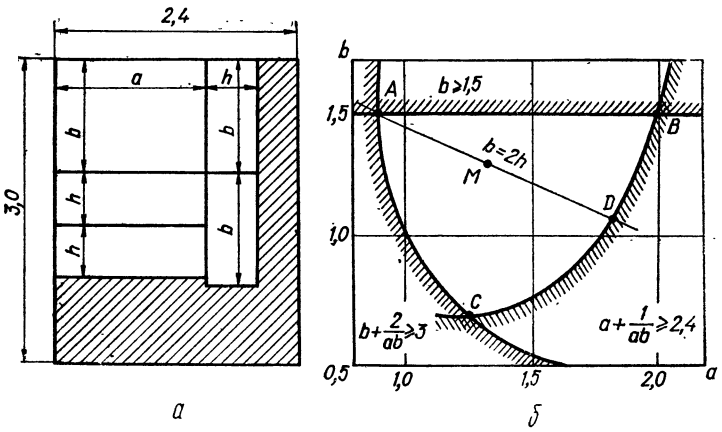


Рис. 47

Рассматриваемая задача решается более просто, если учесть, что среди параллелограммов одинакового объема наименьшую площадь граней имеет куб. Рассматривая контейнер без крышки объемом 1 м^3 как половину куба объемом 2 м^3, несложно вычислить оптимальные размеры заготовок по соотношениям $a^* = b^* = 2h^* = \sqrt[3]{2} = 1{,}26$ м^2 и прямоугольного листа (без учета потерь на разрезы) со сторонами $a + h = 1{,}89$ м и $b + 2h = 2{,}52$ м, из которого можно вырезать заготовки с минимальными отходами на разрезы.

Однако обычно размеры листов отличаются от оптимальных и возникает задача такого раскроя листов, при котором форма и размеры остатка от заготовок наиболее полно удовлетворяют другие нужды производства. Предположим, например, что заготовки для контейнера объемом 1 м^3 вырезают из листа размером 2,4 × 3 м. В этом случае целесообразно, прежде всего, оптимизировать размеры заготовок для контейнера при выборе структуры раскроя листа (рис. 47, *а*), близкой к оптимальной по критерию минимальной площади заготовок, соответствующему целевой функции при свободной переменной $h = 1/ab$ и ограничениях-неравенствах $a + 1/ab \leqslant 2{,}4$; $2b \leqslant 3$; $b + 2/ab \leqslant 3$, определяемых размерами листа.

Для выбора размеров заготовок, удовлетворяющих требованиям минимальных отходов при различной форме остатка, из которого могут быть изготовлены другие изделия, определим с помощью микрокалькулятора ограниченную неравенствами область допустимых решений на плоскости переменных a и b (рис. 47, *б*). Точка M в такой области соответствует оптимальным размерам контейнера $a^* = b^* = 1{,}26$ м, $h^* = 0{,}63$ м при площади остатка $\bar{S} = 2{,}4278$ м^2.

Если форма остатка (рис. 48, *а*) не удовлетворяет другим нуждам производства, то оптимальным может оказаться выбор другой точки в допустимой области, включая ее границу, при которой площадь заготовок увеличится по сравнению с минимальной, но будут более полно удовлетворены другие нужды производства и, следовательно, критерий максимального использования площади листа.

Например, при выборе решения в точке A ($a = 0{,}889$ м, $b = 1{,}5$ м, $h = 0{,}75$ м, $S = 4{,}917$ м^2) площадь остатка $\overline{S} = 2{,}283$ м^3 меньше, чем в предыду-

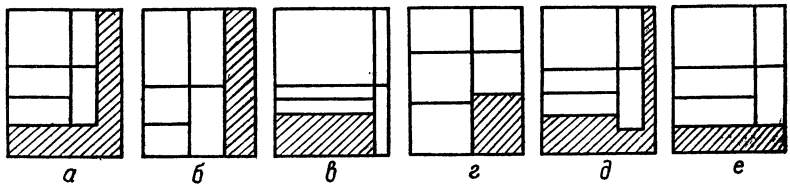


Рис. 48

щем случае, но его форма (рис. 48, *б*) может обеспечить минимум отходов при изготовлении других изделий. При выборе точки B на границе допустимой области ($a = 1{,}89$ м, $b = 1{,}5$ м, $h = 0{,}32$ м) площадь остатка $\overline{S} = 1{,}784$ м^2 еще меньше, но его форма (рис. 48,*в*) может оказаться наиболее приемлемой. Это может относиться и к точке C, соответствующей площади остатка $\overline{S} = 1{,}873$ м^2 (рис. 48, *г*). Оптимальное по критерию наилучшего использования листа решение может оказаться и внутри области допустимых решений, например, в точке, соответствующей остатку сложной конфигурации (рис. 48, *д*), приемлемой в определенных случаях. Точки, лежащие в допустимой области на прямой $b = 2h$

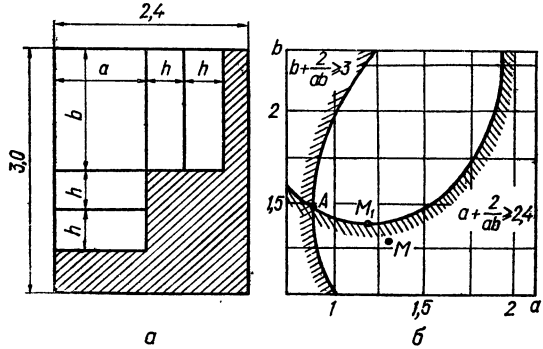


Рис. 49

(см. рис. 47, *б*), соответствуют изменению формы остатка от прямоугольной в точке A до прямоугольной в точке D (рис. 48, *е*).

Возможен выбор оптимального решения и при другой структуре раскроя. Так, при способе раскроя, показанном на рис. 49, *а*, допустимая область решений ограничена неравенствами $a + 2/ab \leqslant 2{,}4$; $b + 2/ab \leqslant 3$, причем оптимальная для заготовок контейнера точка M лежит за пределами области допустимых решений (рис. 49, *б*). В подобных случаях ближайшее к минимуму значение целевой функции на границе допустимой области называют *условным* минимумом, для поиска которого минимизируют целевую функцию на участке границы, ближайшем к минимуму за пределами допустимой области. В рассматриваемой задаче условный минимум несложно найти, подставив в целевую функцию (8.6) значение $b = 2/(a\,(2{,}4 - a))$ из равенства $a + 2/ab = 24$, соответствующее ближайшему к минимуму отрезку границы допустимой области. Минимизировав полученную таким образом целевую функцию $S_1 = 4{,}8 / (a\,(2{,}4 - a)) + a\,(2{,}4 - a)$ одной переменной, получим точку условного минимума M_1,

в которой $a^*= 1{,}2$ м; $b^*= 1{,}389$ м; $h^*= 0{,}6$ м и $S = 4{,}7733$ м². В этом случае площадь остатка $S = 2{,}427$ м² максимальна для выбранной структуры раскроя, но форма может оказаться неприемлемой. Поэтому следует исследовать изменения формы и площади остатка и в других точках области допустимых решений, выбрав наилучшее решение. Например, в точке A (рис. 49, *б*) при $a = 0{,}9$ м; $b = 1{,}48$ м; $h = 0{,}75$ м площадь остатка меньше ($S = 2{,}2944$ м²), но его форма прямоугольна, что может оказаться более приемлемым для удовлетворения других нужд производства при минимальных отходах листа.

Подобным образом следует рассмотреть и другие способы раскроя листа, выбрав среди всех такой, при котором отходы производства минимальны. При строгом решении подобных задач для каждой структуры следует составить целевую функцию, определяющую площадь остатка листа в зависимости от размеров заготовок всех изделий, которые могут быть изготовлены из листа, и минимизировать ее. Такая функция оказывается, как правило, многоэкстремальной и обычно задача заключается в поиске глобального минимума, хотя иногда выбирают минимум, соответствующий изготовлению наиболее нужных изделий и не являющийся глобальным.

В общем случае минимизация нелинейных функций с ограничениями-неравенствами, особенно нелинейными, является наиболее сложной задачей нелинейного программирования, связанной с громоздкими вычислениями, не всегда приводящими к искомому результату. Возможность использования даже программируемых микрокалькуляторов для решения таких задач существенно ограничена их относительно малыми быстродействием и емкостью запоминающих устройств. Однако во многих случаях при небольшом числе переменных и ограничений для решения этих задач могут быть успешно использованы и программируемые микрокалькуляторы.

Особенно удобно решение задач с ограничениями при минимизации целевых функций одной или двух переменных, когда можно использовать графическое представление ограничений на прямой или плоскости. Для минимизации многоэкстремальных целевых функций с ограничениями в этом случае можно использовать программы, подобные программе 121/34, для грубого определения экстремальных областей в области допустимых решений с последующим использованием, при необходимости, программ, реализующих модифицированные алгоритмы координатного спуска к минимумам овражных функций. При этом следует учитывать, что во многих случаях объем вычислений и затраты времени существенно сокращаются при предварительном исследовании свойств функции и ограничений методами анализа.

В качестве примера рассмотрим минимизацию тестовой [18] функции $\Phi= (x_1- 2)^2+ (x_2- 1)^2$ при ограничениях $-x_1^2+ x_2 \geqslant 0$ и $-x_1- x_2+ 2 \geqslant 0$. Область допустимых решений для таких условий задачи ограничена линиями, на которых справедливы равенства $x_1^2= x_2$ и $x_1= 2 - x_2$. Из совместного решения этих равенств несложно найти точки $X = [-2\ 4]$ и $X = [1\ 1]$, в которых пересекаются граничные линии, а простое исследование равенств показывает, что значения переменных в допустимой области ограничены интервалами $-2 \leqslant x_1 \leqslant 1$ и $1 \leqslant x_2 \leqslant 4$.

Заданная целевая функция имеет две независимые составляющие, каждая из которых равна нулю соответственно при $x_1= 2$ и $x_2= 1$ и возрастает при других значениях этих переменных. Следовательно, минимальное значение целевой функции в области допустимых решений $\min \Phi = 1$ соответствует точке оптимума $X^*= [1\ 1]$, лежащей на пересечении граничных линий.

ПРИЛОЖЕНИЕ

В дополнение к основному тексту книги приведем библиотеку программ в компактной записи на входных языках ЯМК34 и ЯМК21. В программах на входном языке ЯМК34, часть операторов которого обозначена различными символами на клавиатуре микрокалькуляторов разных типов, использованы символы команд, указываемые на клавиатуре микрокалькулятора типа «Электроника Б3-34». Это, в частности, относится к символам команд П, ИП, XY и ↑, более удобных при составлении программ, чем соответствующие обозначения этих команд $x \to$ П, П $\to x$, $\leftrightarrow$ и В↑, указываемые, например, на клавиатуре микрокалькуляторов типов «Электроника МК-54» и «Электроника МК-56». Обозначения величин в инструкциях к выполнению программ и контрольных примерах объяснены в заголовках программ и основном тексте книги.

ПРОГРАММЫ НА ВХОДНОМ ЯЗЫКЕ ЯМК34

Программа 124/34. Вычисление многочлена $A(x)$ вещественного аргумента x произвольной степени

П9	КИП0	ИП0	С/П	ИП9	ИП7	×	+	П9	ИП0
$x=0$	01			ИП9	С/П				

Инструкция: $x = P7$, $n = P0$, $a_n = PX$ В/О С/П $PX = n-1$, $a_{n-1} = PX$ С/П $PX = n-2, \ldots,$ $a_1 = PX$ С/П $PX = 0$, $a_0 = PX$ С/П $PX = A(x)$.

Контрольный пример: при $x = 2$ получим $A(x) = 3x^3 + 2x^2 + x + 0{,}5 = 34{,}5$.

Программа 125/34. Вычисление многочлена $A(p)$ комплексного аргумента $p = x + fy$ произвольной степени

П9	Сх	П6	КИП0	ИП0	С/П	ИП9	ИП7	×	ИП6
ИП8	×	—	+	ИП9	ИП8	×	ИП6	ИП7	×
+	П6	XY	П9	ИП0	$x=0$	03	—	С/П	

Инструкция: $n = P0$, $x = P7$, $y = P8$, $a_n = PX$ В/О С/П $PX = n-1$, $a_{n-1} = PX$ С/П $PX = n-2, \ldots,$ $a_1 = PX$ С/П $PX = 0$, $a_0 = PX$ С/П $PX = P9 = \mathrm{Re}\, A(p)$, $PY = P6 = \mathrm{Im}\, A(p)$.

Контрольный пример: при $p = 2 + j2$ получим $A(p) = 3p^3 + 2p^2 + p + 0{,}5 = -45{,}5 + j66$.

Программа 126/34. Вычисление многочлена A (jy) мнимого аргумента jy произвольной степени n

П4	Сх	П5	КИП0	ИП0	С/П	ИП5	/—/	ИП8	×
+	ИП4	ИП8	×	П5	ХУ	П4	ИП0	$x=0$	03
—	С/П								

Инструкция: $n = P0$, $y = P8$, $a_n = PX$ В/О С/П $PX = n-1$, $a_{n-1} = PX$ С/П $PX = n-2, \ldots$, $a_{n-2} = PX$ С/П $PX = 0$, $a_0 = PX$ С/П $PX = P4 = \mathrm{Re}\,A(\mathrm{j}y)$, $PY = P5 = \mathrm{Im}\,A(\mathrm{j}y)$.

Контрольный пример: при $y = 2$ получим $A(\mathrm{j}y) = 3(\mathrm{j}y)^3 + 2(\mathrm{j}y)^2 + \mathrm{j}y + 0{,}5 = -7{,}5 - \mathrm{j}22$.

Программа 127/34. Вычисление многочлена степени $n \leqslant 12$ мнимого аргумента $\mathrm{j}y = p$

ПД	ИПС	×	ИПВ	ХУ	ИПА	ПП	41	ИП9	ПП
41	ИП8	ПП	41	ИП7	ПП	41	ИП6	ПП	41
ИП5	ПП	41	ИП4	ПП	41	ИП3	ПП	41	ИП2
ПП	41	ИП1	ПП	41	ИП0	ПП	41	С/П	БП
00	ХУ	/—/	ИПД	×	+	ХУ	ИПД	×	В/О

Инструкция: ($a_0 = P0$, $a_1 = P1$, ..., $a_9 = P9$, $a_{10} = PA$, $a_{11} = PB$, $a_{12} = PC$) $y = PX$ (В/О) С/П $PX = \mathrm{Im}\,A(p)$, $PY = \mathrm{Re}A(p)$.

Контрольный пример: при $p = \mathrm{j}2$ получим $A(p) = 12p^{12} + 11p^{11} + 10p^{10} + 9p^9 + 8p^8 + 7p^7 + 6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 1 = 40633 - \mathrm{j}18678$ (время счета около 45 с).

Программа 128/34. Вычисление произведения $C(p)$ степени $n+2$ многочлена $A(p)$ произвольной степени n на трехчлен $b_1p^2 + b_2p + b_0$

П9	ХУ	3	+	П3	Сх	П7	П8	ИП9	БП
12	С/П	П9	ИП2	×	ИП7	+	П6	ИП9	ИП1
×	ИП8	+	П7	ИП9	ИП0	×	П8	КИП3	ИП6
ИП3	БП	11							

Инструкция: ($b_2 = P2$, $b_1 = P1$, $b_0 = P0$) $n = PY$, $a_n = PX$ В/О С/П $PX = n+2$, $PY = c_{n+2}$, $a_{n-1} = PX$ С/П $PX = n+1$, $PY = c_{n+1}$, $a_{n-2} = PX$ С/П $PX = n$, $PY = c_n$, $a_{n-3} = PX$ С/П $PX = n-1$, $PY = c_{n-1} \ldots a_0 = PX$ С/П $PX = 2$, $PY = c_2$, $P7 = c_1$, $P8 = c_0$.

Контрольный пример: $(3p^2 + 2p + 1)(4p^2 + 3p + 2) = 12p^4 + 17p^3 + 16p^2 + 7p + 2$.

Программа 129/34. Вычисление частного С (p) и остатка r от деления многочлена $A(p)$ произвольной степени n на двучлен $b_1p + b_0$

П9	ХУ	П2	ИП0	ИП1	÷	П4	КИП2	ИП9	ИП1
÷	ИП2	С/П	ИП9	ИП4	×	—	П9	БП	07

Инструкция: ($b_1 = P1$, $b_0 = P0$) $n = PY$, $a_n = PX$ В/О С/П $PX = n-1$, $PY = c_{n-1}$, $a_{n-1} = PX$ С/П $PX = n-2$, $PY = c_{n-2}$,

$a_{n-2} = PX$ С/П ... $PX = 1$, $PY = c_1$, $a_1 = PX$ С/П $PX = 0$, $PY = c_0$, $a_0 = PX$ С/П $PX = -99999999$, $PY = r$.

Контрольный пример: $(3p^3 + 2p^2 + p + 0{,}5)/(2p + 1) = 1{,}5p^2 + 0{,}25p + 0{,}375$ с остатком $r = 0{,}0625$.

Программа 130/34. Решение системы из двух линейных уравнений по формулам Крамера

ИП7	ИП5	×	ИП8	ИП4	×	—	ПД	ИП5	ИП9
×	ИП8	ИП6	×	—	ИПД	÷	П1	ИП7	ИП6
×	ИП4	ИП9	×	—	ИПД	÷	П2	ИП1	С/П
БП	00								

Инструкция: ($a_{11} = P7$, $a_{12} = P8$, $q_1 = P9$, $a_{21} = P4$, $a_{22} = P5$, $q_2 = P6$) (В/О) С/П $PX = P1 = x_1$, $PY = P2 = x_2$, $РД = \Delta$.

Контрольный пример: для системы уравнений $x_1 + x_2 = 2$, $2x_1 + 3x_2 = 1$ получим $x_1 = 5$, $x_2 = -3$, $\Delta = 1$.

Программа 131/34. Решение системы из трех линейных уравнений методом оптимального исключения

ИПА	ИП7	÷	ПА	ИП9	ИП7	÷	П9	ИП8	ИП7
÷	П8	ИП5	ИП4	ИП8	×	—	ПД	ИП6	ИП4
ИП9	×	—	ИПД	÷	П6	ИПВ	ИП4	ИПА	×
—	ИПД	÷	ПВ	ИП9	ИП8	ИП6	×	—	П9
ИПА	ИП8	ИПВ	×	—	ПА	ИПС	ИП1	ИПА	×
—	ИП2	ИПВ	×	—	ИП3	ИП9	—	ИП2	ИП6
×	—	÷	ПС	ИПВ	ИПС	ИП6	×	—	ПВ
ИПА	ИПС	ИП9	×	—	ПА	С/П	БП	00	

Инструкция: $a_{11} = P7$, $a_{12} = P8$, $a_{13} = P9$, $a_{21} = P4$, $a_{22} = P5$, $a_{23} = P6$, $a_{31} = P1$, $a_{32} = P2$, $a_{33} = P3$, $q_1 = PA$, $q_2 = PB$, $q_3 = PC$ (В/О) С/П $PX = PA = x_1$, $PB = x_2$, $PC = x_3$.

Контрольный пример: для системы уравнений

$$\begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 50 \\ 32 \\ 17 \end{bmatrix}$$

получим $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ (время счета около 30 с).

Программа 132/34. Вычисление определенных интегралов $I(x)$ по составной формуле Симпсона

П0	ПП	40	ПС	ИПВ	ИПА	ПВ	—	ИП0	÷
ПА	ПП	40	1	ПП	28	4	ПП	28	2
БП	14	ИПС	3	÷	ИПА	×	С/П	×	ИПС
+	ПС	КИП0	ИП0	$x \neq 0$	22	ИПВ	ИПА	+	ПВ
...	В/О								

Инструкция: заменить многоточие операторами вычисления подынтегральной функции (при $x = PB$ и свободных регистрах *1*, ..., *9*, *Д*), $a = PA$, $b = PB$, четное число $n = PX$ В/О С/П $PX = I(x)$.

Контрольный пример: для вычисления интеграла

$$I(x) = \int_1^5 \frac{x^3}{x^4 + 16}\, dx,$$

заменив многоточие операторами ИПВ ↑ x^2 × ИПВ x^2 x^2 1 6 + ÷ при $1 = PA$, $5 = PB$, $4 = PX$ получим $I = 0{,}91723623$ (время счета около 40 с).

Программа 133/34. Вычисление интегрального синуса Si (x)

П2	9	—	$x<0$	35	ИП2	x^2	П3	ИП2	1
П4	→	КИП4	→	ИП4	÷	КИП4	→	ИП4	÷
ИП3	/—/	×	↑	ИП4	÷	ИП2	+	П2	Вх
—	$x=0$	11	ИП2	С/П	ИП2	x^2	$1/x$	↑	↑
↑	6	0	×	1	—	×	3	0	×
1	+	×	2	4	×	2	—	×	1
+	ИП2	cos	×	ИП2	÷	П3	→	3	0
×	1	—	ИП2	sin	×	×	ИП3	—	π
2	÷	+	С/П						

Инструкция: $x = PX$ В/О С/П $PX = \mathrm{Si}\,(x)$.

Контрольный пример: время вычисления Si (0,1) = 0,099944467 около 25 с, время вычисления Si (20) = 1,5483802 — около 22 с.

Программа 134/34. Вычисление коэффициентов ортогональных многочленов Лежандра $P_n(x)$

П2	1	П6	П8	+	П3	2	П7	×	3
—	П5	П4	ИП8	×	ИП6	÷	П8	КИП6	ИП4
2	—	$x<0$	12	ИП8	С/П	КИП3	ИП3	КИП3	→
ИП3	×	ИП8	×	ИП7	÷	ИП5	/—/	÷	П8
ИП7	2	+	П7	ИП5	2	—	П5	БП	24

Инструкция: $n = PX$ В/0 С/П $PX = a_n$ С/П $PX = a_{n-2}$ С/П $PX = a_{n-4}$ . . . СП $PX = 0$.

Контрольный пример: $P_7(x) = 26{,}8125x^7 - 43{,}3125x^5 + 19{,}6875x^3 - 2{,}1875x$.

Программа 135/34. Вычисление коэффициентов ортогональных многочленов Эрмита $H_n(x)$

П7	1	П8	+	П0	2	П9	ИП8	С/П	КИП0
ИП0	КИП0	→	ИП0	×	ИП8	×	ИП9	/—/	÷
П8	ИП9	2	+	П9	БП	07			

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-2}$ С/П $PX = a_{n-4}$... С/П $PX = 0$.

Контрольный пример: $H_{10}(x) = x^{10} - 45x^8 + 630x^6 - 3150x^4 + 4725x^2 - 945$.

Программа 136/34. Вычисление коэффициентов ортогональных многочленов Лагерра $L_n(x)$

П7	Cx	П4	ИП7	π	×	cos	П8	С/П	ИП4
ИП7	—	x^2	ИП8	×	КИП4	→	ИП4	÷	/—/
БП	07								

установить переключатель Р—Г в положение Р.

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-1}$ С/П $PX = a_{n-2}$ . . . С/П $PX = 0$.

Контрольный пример: $L_7(x) = -x^7 + 49x^6 - 882x^5 + 7350x^4 - 29400x^3 + 52920x^2 - 35280x + 5040$.

Программа 137/34. Вычисление коэффициентов ортогональных многочленов Чебышева $T_n(x)$ при $n>2$

ПД	2	П7	П5	—	П1	ИП7	2	×	П7
1	П8	КИП1	ИП1	$x=0$	06	ИП7	С/П	/—/	ИПД
×	4	÷	П6	ИП8	×	С/П	ИП6	ИП5	÷
П7	1	П8	КИП5	ИП5	П4	2	—	П0	ИП4
ИПД	—	ИП8	×	П8	КИП4	КИП0	ИП0	$x=0$	39
ИП7	БП	21							

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-2}$ С/П $PX = a_{n-4}$ . . . С/П $PX = 0$.

Контрольный пример: $T_{10}(x) = 512x^{10} - 1280x^8 + 1120x^6 - 400x^4 + 50x^2 - 1$ (дробные результаты округлить до целых чисел).

Программа 138/34. Вычисление гиперболических и обратных гиперболических функций вещественного аргумента

10^x	0	,	8	+	2	×	8	+	ПД
КБПД	ИП9	x^2	1	БП	34	ИП9	e^x	ИП9	БП
42	БП	50	ИП9	e^x	ИП9	/—/	e^x	—	2
÷	С/П	БП	00	—	$\sqrt{\ }$	ИП9	+	ln	С/П
БП	00	/—/	e^x	+	2	÷	С/П	БП	00
ИП9	1	+	ИП9	1	—	÷	ln	2	÷
С/П	БП	00	ИП9	e^x	ИП9	/—/	e^x	—	ИП9
e^x	ИП9	/—/	e^x	+	÷	С/П	БП	00	00
00	ИП9	БП	63	ИП9	x^2	1	+	$\sqrt{\ }$	ИП9
+	ln	С/П	БП	00					

Инструкция: $x = P9$, вычислить соответствующую тригонометрическую или обратную тригонометрическую функцию аргумента 1, выполнить (В/О) С/П, зарегистрировать высвечиваемое значение гиперболической или обратной гиперболической функции.

Контрольный пример: для вычисления sh0,5 = 0,521095 ввести 0,5 = *P9*, вычислить sin 1 = 0,84147103, нажать клавиши В/О и С/П; для вычисления arch5 = 2,2924316 ввести 5 = *P9*, вычислить arccos 1 = 0, нажать клавишу С/П.

Программа 139/34. Преобразование десятичных представлений целых чисел $N_{10} < 256$ в двоичные представления N_2

ПД	П9	Cx	П7	1	2	8	ПС	ИПД	ИПС
—	$x \geqslant 0$	18	ПД	1	П8	БП	20	Cx	П8
ИП7	1	0	×	ИП8	+	П7	ИПС	2	÷
ПС	1	—	$x < 0$	08	ИП7	С/П	БП	00	

Инструкция: N_{10}= PX (В/О) С/П $PX = N_2$, $P9 = N_{10}$ (время счета около 60 с).

Контрольный пример: для N_{10}= 170 получим N_2= 10101010.

ПРОГРАММЫ НА ВХОДНОМ ЯЗЫКЕ ЯМК21

Программа 140/21. Вычисление функции arctg x с абсолютной погрешностью менее 0,05″

P8	x^2	P2	1	+	↑	F6	×	$\sqrt{}$	↑	F2	+
↑	F5	+	↑	F4	×	$\sqrt{}$	↑	F2	+	↑	F7
+	$\sqrt{}$	P3	F8	9	0	×	↑	F3	÷	С/П	БП
P0											

Инструкция: 0,8114228 = $P7$; 1,6211707 = $P4$; 1,056546 = $P5$; $P6$ = 0,403225 $x = PX$ (В/О) С/П PX = arctg x в градусах.

Программа 141/21. Преобразование комплексного числа A = = Re A + jIm A в тригонометрическую форму $A = |A| e^{j\varphi_A}$ с вычислением φ_A° в интервале [0,360°] при предельной погрешности 0,03°

P8	x^2	XY	P7	x^2	+	$\sqrt{}$	P6	F8	↑	F7	÷
P2	x^2	1	+	1	,	6	2	×	$\sqrt{}$	+	5
$1/x$	+	$\sqrt{}$	9	0	XY	÷	↑	F2	×	P3	F7
$x < 0$	FCx	F3	1	8	0	+	P3	БП	P9	F3	$x < 0$
P9	3	6	0	+	P3	F3	↑	F6	С/П	БП	P0

Инструкция: Re $A = PY$, Im A= PX (В/О) С/П $PX = |A|$, PY= = φ_A°.

Контрольный пример: $1 + j1 = 1{,}414213e^{j45°}$, $-2 + j2$ = = $2{,}828427e^{j135°}$, $-3 - j3 = 4{,}242640e^{j225°}$, $4 - j4 = 5{,}656854e^{j315°}$ (время счета около 8 с).

Программа 142/21. Преобразование комплексного числа A = = Re A + jIm A в тригонометрическую форму $A = |A| e^{j\varphi_A}$ с вычислением φ_A° в интервале [—180, 180°] с максимальной точностью

P8	x^2	XY	P7	x^2	+	P6	÷	↑	F5	×	$\sqrt{}$
+	XY	→	F4	+	↑	F3	×	↑	←	XY	×
$\sqrt{}$	→	F2	×	/—/	1	+	↑	←	+	↑	F6
×	$\sqrt{}$	↑	F7	—	XY	÷	9	0	×	↑	F8
$x < 0$	—	F1	/—/	↑	F6	$\sqrt{}$	С/П	БП	P0		

Инструкция: (0,1885772 = *P2*; 9,1670712 · 10^{-2} = *P3*; 17,684716 = = *P4*; 126,10829 = *P5*) Re *A* = *PY*, Im *A* = *PX* (В/О) С/П *PX* = = |*A*|, *PY* = φ_A°. Для проверки можно воспользоваться данными контрольного примера к предыдущей программе.

Программа 143/21. Вычисление результатов *C* = *A* *O* *B* арифметических операций над комплексными числами *A* и *B*

↑	F4	+	P7	F8	↑	F5	+	P8	С/П	F8	↑
F5	×	←	БП	F3	БП	FXY	F4	×	БП	FВП	F4
x^2	↑	F5	x^2	+	$1/x$	↑	F4	×	P4	F5	/—/
×	P5	БП	FXY	←	F7	↑	F5	×	↑	→	+
P8	F7	↑	F4	×	↑	→	—	P7	С/П	↑	F7

Инструкция: Re*A* = *P7*, Im*A* = *P8*, Re*B* = *P4*, Im*B* = *P5* БП О С/П (для сложения БП + С/П, для вычитания F4 /—/ P4 F5 /—/ P5 БП + С/П, для умножения БП × С/П, для деления БП ÷С/П) *P7* = Re*C*, *P8* = Im*C*.

Контрольный пример: для вычисления (—2 + j2)(2 — j3) : (3 + + j4) + (6 + j6) = 7,84 + j6,88 выполнить: —2 = *P7*, 2 = *P8*, 2 = = *P4*, —3 = *P5* БП × С/П (*P7* = 2, *P8* = 10) 3 = *P4*, 4 = *P5* БП ÷ С/П (*P7* = 1,84; *P8* = 0,88) 6 = *P4*, 6 = *P5* БП + С/П *P7* = 7,84; *P8* = 6,88.

Программа 144/21. Вычисление многочлена *A* (*x*) вещественного аргумента степени *n* ⩽ 11

P3	↑	F6	×	↑	F5	ПП	6	F4	ПП	6	F8
ПП	6	F7	ПП	6	ПП	P6	ПП	P6	ПП	P6	ПП
P6	ПП	P6	ПП	P6	F2	→	P2	+	С/П	БП	P0
F2	→	P2	+	↑	F3	×	↑	В/О			

Инструкция: (a_0 = *P2*, a_1 = *C1*, a_2 = *C2*, ..., a_6 = *C6*, a_7 = *P7*, a_8 = *P8*, a_9 = *P4*, a_{10} = *P5*, a_{11} = *P6*) *x* = *PX* (В/О) С/П *PX* = *A* (*x*).

Контрольный пример: время вычисления многочлена $A(x) = 11x^{11} + 10x^{10} + 9x^9 + 8x^8 + 7x^7 + 6x^6 + 5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 0{,}5 = 40962{,}5$ при $x = 2$ составляет около 18 с.

Программа 145/21. Вычисление многочлена *A* (*x*) вещественного аргумента произвольной степени

P2	F7	↑	F8	×	↑	F2	+	P7	F4	1	—
P4	С/П	БП	P0								

Инструкция: *n* = *P4*, a_n = *P7*, *x* = *P8*, a_{n-1} = *PX* В/0 С/П *PX* = *n* — 1, a_{n-2} = *PX* С/П *PX* = *n* — 2, ..., a_0 = *PX* С/П *PX* = 0, *P7* = *A* (*x*).

Для проверки можно воспользоваться данными контрольного примера к предыдущей программе.

Программа 146/21. Вычисление многочлена *A* (*p*) мнимого аргумента *p* = j*y* степени *n* ⩽ 10

P6	x^2	↑	F5	×	/—/	↑	F8	+	↑	F6	×
P3	F6	x^2	↑	F4	×	/—/	↑	F7	+	ПП	P7
ПП	P7	ПП	P7	ПП	P7	ПП	P7	ПП	P7	ПП	P7
↑	F3	*XY*	С/П	БП	P0	↑	F6	×	↑	F3	XY
P3	F6	×	/—/	↑	F2	→	P2	+	В/О		

Инструкция: (a_0= *P2*, a_1= *C1*, a_2= *C2*, ..., a_6= *C6*, a_7= *P7*, a_8= *P8*, a_9= *P4*, a_{10}= *P5*) y = *PX* (В/О) С/П *PX* = *P3* = Re*A* (jy), *PY* = Im*A* (jy).

Контрольный пример: время вычисления многочлена $A(p) = 10p^{10} + 9p^9 + 8p^8 + 7p^7 + 6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 0{,}5 = -8519{,}5 + j3850$ при $p = j2$ составляет около 25 с.

Программа 147/21. Вычисление многочлена $A(p)$ мнимого аргумента $p = jy$ произвольной степени

P2	F7	↑	F3	×	↑	F8	XY	P8	F3	×	/—/
↑	F2	+	P7	F4	1	—	P4	С/П	БП	P0	

Инструкция: (y = *P3*) n = *P4*, a_n= *P7*, 0 = *P8*, a_{n-1}= *PX* В/О С/П *PX* = $n - 1$, a_{n-2}= *PX* С/П *PX* = $n - 2$, ..., a_0= *PX* С/П *PX* = 0, *P7* = Re*A* (jy), *P8* = Im*A* (jy).

Для проверки можно воспользоваться данными контрольного примера к предыдущей программе.

Программа 148/21. Вычисление многочлена $A(p)$ комплексного аргумента $p = x + jy$ ($x \neq 0$, $y \neq 0$) степени $n < 8$

P4	÷	P5	F8	×	P6	↑	F5	÷	↑	F7	ПП
P/—/	ПП	,	ПП	,	ПП	,	ПП	,	ПП	,	ПП
,	F2	→	+	С/П	F2	→	P2	+	P3	↑	F5
×	↑	F6	XY	+	P6	F5	×	↑	F3	XY	—
↑	F4	×	P3	F6	↑	F4	×	P6	F3	↑	В/О

Инструкция: (a_1= *C1*, a_2= *C2*, ..., a_6= *C6*, a_7= *P7*, a_8= *P8*) a_0= *P2*, y = *PY*, x = *PX* В/О С/П *PX* = Re*A* (*p*), *P6*= Im*A* (*p*).

Контрольный пример: для многочлена $A(p) = 8p^8 + 7p^7 + 6p^6 + 5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 0{,}5$ при $p = 2 + j2$ получим Re*A* (*p*) = 38994,5; Im*A* (*p*) = —10814 (время счета около 40 с).

Программа 149/21. Вычисление многочлена $A(p)$ комплексного аргумента $p = x + jy$ произвольной степени

P4	Cx	P2	P3	F4	С/П	→	F3	↑	F8	×	→
F7	×	→	F2	↑	F8	×	↑	←	+	P3	F2
↑	F7	×	↑	←	—	↑	←	+	P2	F4	1
—	P4	БП	↑								

Инструкция: x = *P7*, y = *P8*, n = *PX* В/О С/П *PX* = n, a_n = *PX* С/П *PX* = $n - 1$, a_{n-1}= *PX* С/П *PX* = $n - 2$, ..., a_1= *PX* С/П *PX* = 0, a_0= *PX* С/П *PX* = —1, *P2*= Re*A* (*p*), *P3* = Im*A* (*p*).

Для проверки можно воспользоваться данными контрольного примера к предыдущей программе.

Программа 150/21. Вычисление коэффициентов многочлена $A(p)$ при смещении начала отсчета аргумента многочлена $B(p)$ на постоянную вещественную составляющую x_0 при степени $n \leqslant 8$

Р3	7		Р4	Р5		F8	↑	F3	×	↑	F7	+		Р7
Р6	F6		↑	F3		×	↑	F2	→	+	Р2	Р6		F4
1	—		Р4	$x=0$		F2	7	↑	F5	—	Р4	$x\neq 0$		РСх
F2	→		Р2	F4		1	—	БП	5	F5	1	—		Р5
Р4	$x=0$		F ↑	F8		↑	F3	×	↑	F7	+	Р7		С/П

Инструкция: $b_8 = P8$, $b_7 = P7$, $b_6 = C6$, $b_5 = C5$, ..., $b_1 = C1$, $b_0 = P2$, $x_0 = PX$ В/О С/П $PX = P7 = a_7$, $P8 = a_8$, $C6 = a_6$, $C5 = a_5$, ..., $C1 = a_1$, $P2 = a_0$.

Контрольный пример: для многочлена $B(p) = p^8 + p^7 + p^6 + p^5 + p^4 + p^3 + p^2 + p + 1$ при $x_0 = 1$ получим $A(p) = p^8 + 9p^7 + 36p^6 + 84p^5 + 126p^4 + 126p^3 + 84p^2 + 36p + 9$ (время счета около 130 с).

Программа 151/21. Вычисление коэффициентов произведения $C(p) = A(p)\,B(p)$ многочлена $B(p)$ произвольной степени m на многочлен $A(p)$ степени $n \leqslant 5$

6	Р8	Сх	→	F8		1	—	Р8		$x=0$		Р ↑	С/П	Р8
F7	ПП	Р6	F6	ПП		Р6	F5	ПП		Р6		F4	ПП	Р6
F3	ПП	Р6	F2	ПП		Р6	→	Р7		Сх		→	БП	FXY
↑	F8	×	↑	F7		→	+	Р7		В/О				

Инструкция: ($a_5 = P7$, $a_4 = P6$, $a_3 = P5$, $a_2 = P4$, $a_1 = P3$, $a_0 = P2$) В/О С/П $PX = 0$, $b_m = PX$ С/П $PX = c_{m+5}$, $b_{m-1} = PX$ С/П $PX = c_{m+4} \ldots b_0 = PX$ С/П $PX = c_5$, $0 = PX$ С/П $PX = c_4$, $0 = PX$ С/П $PX = c_3$, $0 = PX$ С/П $PX = c_2$, $0 = PX$ С/П $PX = c_1$, $0 = PX$ С/П $PX = c_0$.

Контрольный пример: $(5p^5 + 4p^4 + 3p^3 + 2p^2 + p + 0{,}5)/(p^2 + 2p + 3) = 5p^7 + 14p^6 + 26p^5 + 20p^4 + 14p^3 + 8{,}5p^2 + 4p + 1{,}5$ (время вычисления одного коэффициента около 15 с).

Программа 152/21. Вычисление коэффициентов частного $C(p) = A(p)/(b_1 p + b_0)$ и остатка r_0 для многочлена $A(p)$ произвольной степени

Р8	XY	Р7	Сх	Р4	С/П	↑		F4	—	↑	F7	÷		
Р3	↑	F8	×	Р4	F3	БП	↑							

Инструкция: $b_1 = PY$, $b_0 = PX$ В/О С/П $PX = 0$, $a_n = PX$ С/П $PX = c_{n-1}$, $a_{n-1} = PX$ С/П $PX = c_{n-2} \ldots a_1 = PX$ С/П $PX = c_0$, $a_0 = PX$ С/П $PX = r_0$, $P7 = b_1$, $P8 = b_0$.

Контрольный пример: $(3p^3 + 2p^2 + 1p + 0{,}5)/(2p + 1) = 1{,}5p^2 + 0{,}25p + 0{,}375$ с остатком $r_0 = 0{,}0625$.

Программа 153/21. Вычисление коэффициентов частного $C(p) = A(p)/(p - x_0)$ и остатка $r_0 = A(x_0)$ для многочлена $A(p)$ степени $n \leqslant 9$

Р3	7	Р6	F4	↑	F3	×		↑	F8	+	Р8	↑
F3	×	↑	F7	+	Р7	Р5		F5	↑	F2	→	+
Р2	Р5	F6	1	—	Р6	$x=0$	F3	С/П	БП	Р0		

Инструкция: $a_9 = P4$, $a_8 = P8$, $a_7 = P7$, $a_6 = C6$, $a_5 = C5$, ..., $a_1 = C1$, $a_0 = P2$, $x_0 = PX$ (В/О) С/П $PX = 0$, $P4 = c_8$, $P8 = c_7$, $P7 = c_6$, $C6 = c_5$, $C5 = c_4$, ..., $C1 = c_0$, $P2 = r_0 = A(x_0)$.

Контрольный пример: для многочлена $A(p) = p^9 + p^8 + p^7 + p^6 + p^5 + p^4 + p^3 + p^2 + p + 1$ при $x_0 = 1$ получим $C(p) = p^8 + 2p^7 + 3p^6 + 4p^5 + 5p^4 + 6p^3 + 7p^2 + 8p + 9$ с остатком $r_0 = A(x_0) = 10$ (время счета около 20 с).

Программа 154/21. Вычисление коэффициентов частного $C(p) = A(p)/(b_2p^2 + b_1p + b_0)$ и остатка $r_1p + r_0$ для многочлена $A(p)$ произвольной степени

Р7	Сх	Р2	Р3	→	С/П	↑	F2		—	↑	←	—
↑	F4	÷	Р8	↑	F3	→	F5		×	Р2	F6	×
Р3	F7	1	—	Р7	1	—	$x=0$	/—/		Сх	→	F8
БП	↑											

Инструкция: ($b_2 = P4$, $b_1 = P5$, $b_0 = P6$) $n = PX$ В/О С/П $a_n = PX$ С/П $PX = c_{n-2}$, $a_{n-1} = PX$ С/П $PX = c_{n-3}$ ... $a_2 = PX$ С/П $PX = c_0$, $a_1 = PX$ С/П $PX = r_1$, $a_0 = PX$ С/П $PX = r_0$.

Контрольный пример: $(p^4 + 2p^3 + 3p^2 + 4p + 5)/(p^2 + 2p + 0{,}5) = p^2 + 2{,}5$ с остатком $-p + 5{,}75$.

Программа 155/21. Вычисление коэффициентов интерполирующего многочлена $P(x)$ третьей степени с началом отсчета аргумента в середине табличного интервала по четырем отсчетам функции y_i с шагом h

F6	2	×	x^2	Р7	×	2	÷	Р8	F4	↑	F3
—	Р4	F3	+	Р3	F5	↑	F2	—	Р5	F2	+
↑	F3	—	8	÷	→	F7	÷	Р7	F3	↑	←
—	2	÷	Р2	F5	3	÷	↑	F4	—	8	÷
→	F8	÷	Р8	F4	↑	←	—	↑	F6	÷	С/П

Инструкция: $y_1 = P2$, $y_2 = P3$, $y_3 = P4$, $y_4 = P5$, $h = P6$ В/О С/П $PX = a_1$, $P2 = a_0$, $P7 = a_2$, $P8 = a_3$.

Контрольный пример: для отсчетов функции $y_i = 0{,}5;\ 2;\ 4{,}5;\ 5{,}25$ при $h = 2$ получим $P(x) = 3{,}296875 + 1{,}307291x - 0{,}046875x^2 - 0{,}05729166x^3$ (время счета около 40 с).

Программа 156/21. Вычисление коэффициентов интерполирующего многочлена $P(x)$ четвертой степени с началом отсчета аргумента в середине табличного интервала по пяти отсчетам y_i функции с шагом h

F6 x^2 1 2 × 2 × → F6 ÷ → F6
x^2 × 2 × → F6 ÷ → ПП /—/ P7 ПП
/—/ /—/ P8 ПП /—/ /—/ P6 ПП /—/ P5 С/П F4
2 × /—/ P4 ↑ F5 /—/ P5 — → F3 2
× P3 ↑ F2 — ↑ ← + ↑ ← ÷ В/О

Инструкция: $y_1 - y_3 = P2$, $y_2 - y_3 = P3$, $y_4 - y_3 = P4$, $y_3 - y_2 = P5$, $h = P6$ В/О С/П $PX = P5 = a_2$, $P8 = a_4$, $P7 = a_3$, $P6 = a_1$, $y_3 = a_0$.

Контрольный пример: для отсчетов $y_i = 10; 20; 35; 50; 55$ при $h = 2$ получим $P(x) = 35 + 8{,}125x + 0{,}052083333x^2 - 0{,}15625x^3 - 0{,}01302083x^4$ (время счета около 60 с).

Программа 157/21. Вычисление вещественного корня нелинейного уравнения $f(x) = 0$ методом половинного деления

F8 2 ÷ P8 sin $x = 0$ 1 F7 С/П F7 ↑ F8
+ P7 ··· ↑ F4 XY P4 × $x < 0$ P0 F8 /—/
P8 БП P0

Инструкция: заменить в программе многоточие операторами вычисления функции $f(x)$ при $x = P7$, свободных регистрах *2*, *3*, *5*, *6* и стека памяти, ввести $x_н = P7$, $\Delta = P8$, совпадающее по знаку число или $f(x_н) = P4$ В/О С/П $PX = P7 = x_1$, $P8 = \Delta x_1$, $P4 = f(x_1)$.

Контрольный пример: для уравнения $2^x - 5x - 3 = 0$ в интервале $-5 \leqslant x \leqslant 0$ заменяем многоточие операторами 5 × P2 F7 2 X^y 3 — ↑ F2 — , вводим $-5 = P7$, $5 = P8 = P4$ и получаем $x_1 = -0{,}4539964$; $\Delta x_1 = 0{,}745 \cdot 10^{-8}$; $f(x_1) = 3 \cdot 10^{-7}$ (время счета около 3,5 мин).

Программа 158/21. Вычисление вещественного корня нелинейного уравнения $f(x) = 0$ методом касательных Ньютона

P7 ··· P6 ··· ↑ F6 ÷ ↑ F7 XY — P7
— sin $x = 0$ F0 F7 С/П

Инструкция: заменить в программе многоточия соответственно операторами вычисления производной $f'(x)$ и функции $f(x)$ при $x = P7$ и свободных регистрах *2*, *3*, *4*, *5*, *8*, стека памяти, ввести начальное приближение $x_0 = PX$ В/О С/П $PX = x_1$.

Контрольный пример: для уравнения $2^x - 5x - 3 = 0$ в интервале $-5 \leqslant x \leqslant 0$ заменяем многоточия соответственно операторами 2 X^y 2 ln × 5 — для вычисления производной $f'(x) = 2^x \ln 2 - 5$ и операторами F7 5 × P2 F7 2 X^y 3 — ↑ F2 — для вычисления функции $f(x)$ и при $x_0 = -2$ получаем $x_1 = -0{,}4539964$ (время счета около 1,5 мин).

Программа 159/21. Вычисление вещественного корня нелинейного уравнения $x = \varphi(x)$ методом простых итераций

P7 F7 ... ↑ F7 XY P7 — sin $x = 0$ F0 F7 С/П

Инструкция: заменить многоточие операторами вычисления функции $\varphi(x)$ при $x = P7$, свободных регистрах *2*, *3*, *4*, *5*, *6*, *8* и стека памяти, начальное приближение $x_0 = PX$ В/О С/П $PX = x_1$.

Контрольный пример: для вычисления корня уравнения $2^x - 5x - 3 = 0$ в интервале $-5 \leqslant x \leqslant 0$ представляем уравнение в форме $x = (2^x - 3)/5$, заменяем многоточие в программе операторами 2 X^y 3 — 5 ÷ и при $x_0 = -5$ получаем $x_1 = -0{,}4539964$ (время счета около 50 с).

Программа 160/21. Решение уравнения $a_2p^2 + a_1p + a_0 = 0$ с повышенной точностью вычисления меньшего вещественного корня

F5	↑	F4	÷	2	÷	/—/	P2	F6	↑	F4	÷
↑	F2	x^2	—	$x \geqslant 0$	÷	$\sqrt{\ }$	P3	2	3	С/П	/—/
$\sqrt{\ }$	↑	F2	$x < 0$	F5	XY	/—/	+	P7	↑	F6	XY
÷	↑	F4	÷	P8	7	8	С/П				

Инструкция: ($a_2 = P4$, $a_1 = P5$, $a_0 = P6$) В/О С/П $PX = 78$ (корни вещественные), $P7 = x_1$, $P8 = x_2$ или $PX = 23$ (корни комплексно-сопряженные), $P2 = \mathrm{Re}p_{1,2}$ $P3 = \mathrm{Im}p_{1,2}$.

Контрольный пример: $p^2 + 1000p + 8 = 0$ получим $x_1 = -999{,}9919$; $x^2 = -0{,}008000064$, для $2p^2 - 4p + 10 = 0$ получим $p_{1,2} = 1 \pm j2$.

Программа 161/21. Решение нормированного уравнения $p^3 + a_2p^2 + a_1p + a_0 = 0$

Cx	↑	F4	+	P7	×	XY	P3	F5	+	P2	↑
F3	×	↑	F6	+	$x \geqslant 0$	P4	F3	↑	F8	—	P3
F8	2	÷	P8	↑	F3	+	XY	+	—	$x = 0$	P↑
F/	2	÷	/—/	P7	x^2	↑	F2	—	$x < 0$	8	/—/
$\sqrt{\ }$	0	С/П	$\sqrt{\ }$	↑	*F7*	XY	+	P2	—	+	С/П

Инструкция: ($a_2 = P4$, $a_1 = P5$, $a_0 = P6$) $1 + |a_{\max}| = P8$ В/О С/П $PX = 0$ (пара комплексно-сопряженных корней), $PY = \mathrm{Im}\,p_{1,2}$, $P7 = \mathrm{Re}p_{1,2}$, $P3 = x_3$ или $PX = x_1$ (все корни вещественные), $P2 = x_2$, $P3 = x_3$.

Контрольный пример: для уравнения $p^3 - 6p^2 + 11p - 6 = 0$ при $12 = P8$ получим $x_1 = 2$, $x_2 = 3$, $x_3 = 1$ (время счета около 3 мин), для уравнения $p^3 + p^2 + 4p - 170 = 0$ при $171 = P8$ получим $p_{1,2} = -3 \pm j5$, $x_3 = 5$ (время счета около 3 мин).

Программа 162/21. Разложение нормированного ($a_4 = 1$) многочлена $A(p)$ степени $n = 4$ на произведение многочленов $B(p)$ и $C(p)$ второй степени

P2	P3	←	↑	→	F2	—	P4	↑	F2	×	→
F8	↑	F3	—	↑	←	—	P5	F4	↑	F3	×
/—/	↑	F7	+	↑	F5	÷	P2	F6	↑	F5	÷
↑	F3	XY	P3	—	sin	$x = 0$	P↑	С/П			

Инструкция: $a_3 = C1$, $a_2 = P8$, $a_1 = P7$, $a_0 = P6$, начальное значение $C_0^{(0)} = PX$ В/О С/П $PX = 0$, $P2 = c_1$, $P3 = c_0$, $P4 = b_1$, $P5 =$

$= b_0$; если вычисления не сходятся при изменении $c_0^{(0)}$ (корни близки по модулю), то преобразовать многочлен $A(p)$, сместив начало отсчета вещественных составляющих корней по программе 150/21, и повторить вычисления.

Контрольный пример: для многочлена $A(p) = p^4 - 6p^3 + 18p^2 - 30p + 25$ при $1 = PX$ получим коэффициенты множителей $C(p) = p^2 - 4p + 5$ и $B(p) = p^2 - 2p + 5$ (время счета около 4 мин).

Программа 163/21. Вычисление вещественного корня x_1 алгебраического уравнения с нормированным ($a_5 = 1$) многочленом $A(p)$ степени $n = 5$, коэффициентов частного $B(p) = A(p)/(p - x_1)$ и остатка r_0

F7	2	÷	P7	F8	P6	5	P3	1	—	P3	F6
↑	F2	→	P2	+	↑	F8	×	P6	F3	$x = 0$	PXY
F2	→	→	F2	×	↑	F7	/—/	XY	$x \geqslant 0$	F6	F7
↑	F8	+	P8	XY	x^2	$\sqrt{}$	+	—	$x = 0$	P0	F8
С/П	1	↑	F8	×	↑	→	+	С/П	БП	P—	

Инструкция: $a_4 = C6$, $a_3 = C5$, $a_2 = C4$, $a_1 = C3$, $a_0 = C2$, значение наибольшего модуля коэффициентов со знаком, противоположным знаку свободного члена, $\pm|a_{max}| = P7 = P8$ В/О С/П $PX = x_1$ С/П $PX = b_3$ С/П $PX = b_2$ С/П $PX = b_1$ С/П $PX = b_0$ С/П $PX = r_0$.

Контрольный пример: для уравнения $p^5 + 4p^4 + 3p^3 + 2p^2 + p + 0{,}5 = 0$ при $-4 = P7 = P8$ получим $x_1 = -3{,}239666$; $B(p) = p^4 + 0{,}7603334p^3 + 0{,}5367733p^2 + 0{,}2610335p + 0{,}1543385$ с остатком $r_0 = -5{,}28 \cdot 10^{-6}$ (время счета около 7 мин).

Программа 164/21. Разложение нормированного ($a_6 = 1$) многочлена $A(p)$ степени $n = 6$ на произведение многочленов $B(p)$ второй и $C(p)$ четвертой степени

F7	ПП	F8	P2	F8	×	P4	ПП	Cx	F8	—	P3
↑	F8	×	P5	ПП	Cx	F4	—	P4	↑	ПП	Cx
F5	—	P5	↑	F6	→	P6	XY	÷	P8	↑	F4
ПП	P8	F5	÷	P7	F6	→	F7	С/П	БП	P0	F7
×	↑	F6	→	P6	XY	—	↑	В/О			

Инструкция: $a_5 = C6$, $a_4 = C5$, $a_3 = C4$, $a_2 = C3$, $a_0 = C2$, $a_1 = C1$, начальные приближения $b_1^{(0)} = P7$, $b_0^{(0)} = P8$ В/О С/П $PX = b_1^{(1)}$ С/П $PX = b_1^{(2)}$ ... С/П $PX = b_1^{(i)}$; при совпадении двух очередных значений b_1 принять $P8 = b_0$, $P2 = c_3$, $P3 = c_2$, $P4 = c_1$, $P5 = c_0$; если значения b_1 не сходятся, то использовать программу 165/21.

Контрольный пример: для многочлена $A(p) = p^6 - 7p^5 + 15p^4 - 45p^3 + 1004p^2 - 2768p + 1800$ при $b_1^{(0)} = b_0^{(0)} = 1$ после 10-й итерации получим $B(p) = p^2 - 3p + 2$ и $C(p) = p^4 - 4p^3 + p^2 - 34p + 900$ (время выполнения программы около 15 с).

Программа 165/21. Разложение нормированного ($a_6 = 1$) многочлена $A(p)$ степени $n = 6$ на произведение многочленов $B(p)$ второй и $C(p)$ четвертой степени

←	P6	↑	F8	÷	P5	ПП	Cx	F8	÷	P4	ПП
Cx	F5	—	↑	F8	—	P3	ПП	Cx	F4	—	↑
F8	÷	P2	↑	F6	←	P6	—	/—/	P7	×	↑
F3	+	ПП	P—	P8	F6	←	F7	С/П	БП	P0	↑
F7	×	↑	F6	←	P6	XY	—	↑	В/О		

Инструкция: $a_0 = C1$, $a_1 = C2$, $a_2 = C3$, $a_3 = C4$, $a_5 = C5$, $a_4 = C6$, начальные приближения $b_1^{(0)} = P7$, $b_0^{(0)} = P8$ В/О С/П $PX = b_1^{(1)}$ С/П $PX = b_1^{(2)}$ . . . С/П $PX = b_1^{(i)}$, $P8 = b_0$, $P2 = c_3$, $P3 = c_2$, $P4 = c_1$, $P5 = c_0$; если вычисления не сходятся при изменении начальных приближений (корни близки по модулю), то изменить с помощью программы 150/21 начало отсчета аргумента и повторить вычисления.

Контрольный пример: для многочлена $A(p) = p^6 + 2p^5 + 3p^4 + 4p^3 + 5p^2 + 6p + 7$ при $b_1^{(0)} = b_0^{(0)} = 2$ получим $b_1^{(1)} = 0{,}25$; $b_1^{(2)} = 1{,}14$; , . . ; $b_1^{(195)} = 0{,}8509894$; $b_1^{(196)} = 0{,}8509899$; $b_0 = 3{,}591390$ и коэффициенты многочлена $C(p) = p^4 + 1{,}149010p^3 - 1{,}569186p^2 + 1{,}208816p + 1{,}949105$.

Программа 166/21. Вычисление вещественного корня x_1 алгебраического уравнения с нормированным ($a_7 = 1$) многочленом $A(p)$ степени $n = 7$, коэффициентов частного $B(p) = A(p)/(p - x_1)$ и остатка r_0

P7	P8	F7	2	÷	P7	F8	P6	7	P3	1	—
P3	F6	↑	F2	→	P2	+	↑	F8	×	P6	F3
$x = 0$	FXY	F2	×	↑	F7	/—/	XY	$x \geqslant 0$	P6	F7	↑
F8	+	P8	XY	x^2	$\sqrt{}$	+	—	$x = 0$	P↑	F8	С/П
1	↑	F8	×	↑	F2	→	+	С/П	БП	F8	

Инструкция: $a_6 = C6$, $a_5 = C5$, ..., $a_1 = C1$, $a_0 = P2$, наибольший модуль коэффициентов со знаком, противоположным знаку свободного члена, $\pm |a_{max}| = PX$ В/О С/П $PX = x_1$ С/П $PX = b_5$ С/П $PX = b_4$ С/П $PX = b_3$ С/П $PX = b_2$ С/П $PX = b_1$ С/П $PX = b_0$ С/П $PX = r_0$.

Контрольный пример: для уравнения $p^7 + p^6 + p^5 + p^4 + p^3 + p^2 + p + 1 = 0$ при $-1 = PX$ получим $x_1 = -1$, $B(p) = p^6 + p^4 + p^2 + 1$ и $r_0 = 0$ (время счета около 10 мин).

Программа 167/21. Решение системы из двух линейных уравнений по формулам Крамера

F8	↑	F4	×	P6	F7	↑	F5	×	↑	F6	—
P6	F8	↑	F3	×	←	F5	↑	F2	×	↑	→
—	↑	F6	÷	→	F4	↑	F2	×	→	F7	↑
F3	×	↑	←	—	↑	F6	÷	P3	←	P2	F6
С/П	БП	P0									

Инструкция: (a_{11}= *P7*, a_{12}= *P8*, a_{21}= *P4*, a_{22}= *P5*) q_1= *P2*, q_2= *P3* (В/О) С/П $PX = P6 = \Delta$, $P2 = x_1$, $P3 = x_2$.

Контрольный пример: для системы уравнений $x_1 + 3x_2 = 1$; $7x_1 - 4x_2 = 2$ получим $\Delta = -25$; $x_1 = 0{,}2$; $x_2 = 0{,}4$.

Программа168/21. Решение системы из трех линейных уравнений по схеме единственного деления Гаусса

F4	ПП	P—	F5	ПП	P—	F6	ПП	P—	С/П	1/*x*	↑
←	×	P7	←	×	P2	←	P3	F7	ПП	P—	P7
F2	ПП	P—	↑	F7	÷	P3	←	←	←	ПП	P—
P2	↑	F4	×	→	F6	→	F5	ПП	P—	↑	←
—	С/П	↑	F3	×	/—/	↑	F8	←	+	P8	В/О

Инструкция: (a_{12}/a_{11}= *P4*, a_{13}/a_{11}= *P5*, q_1/a_{11}= *P6*) a_{22}= *C1*, a_{23}= = *C2*, q_2= *C3*, a_{32}= *C4*, a_{33}= *C5*, q_3= *C6*, a_{21}= *P3* В/О С/П a_{31}= = *P3* В/О С/П P /—/ P /—/ С/П $PX = x_1$, $P2 = x_2$, $P3 = x_3$.

Контрольный пример: для системы уравнений $2x_1 + 4x_2 + 6x_3 = 28$; $x_1 + 3x_2 + 2x_3 = 13$; $3x_1 + 4x_2 + 5x_3 = 26$ получим $x_1 = 1$, $x_2 = 2$, $x_3 = 3$.

Программа 169/21. Вычисления по формулам прямого хода схемы единственного деления Гаусса для заполнения p-й части бланка при решении системы из $n \leqslant 7$ линейных уравнений

→	F2	1	+	P2	P3	С/П	↑	F7	÷	P8	С/П
P4	F2	1	—	P2	F6	←	P6	↑	F8	×	/—/
↑	F4	+	P5	F2	$x \neq 0$	F/—/	F5	БП	XY	F3	P2
7	—	P3	F6	←	P6	F3	1	+	P3	$x = 0$	6
F2	P3	F5	БП	P1							

Инструкция: $a_{1p}^{(p-1)} = P7$, $0 = P2$, $a_{np}^{(p-1)} = PX$ В/О С/П $PX = 1$, $a_{n-1,p}^{(p-1)} = PX$ В/О С/П $PX = 2 \ldots a_{2p}^{(p-1)} = PX$ В/О С/П $PX = n - 1$, $a_{1p+1}^{(p-1)} = PX$ С/П $(PX = a_{1p+1}^{(p)})\, a_{2,p+1}^{(p-1)} = PX$ С/П $PX = a_{2,p+1}^{(p)}$, $a_{3,p+1}^{(p-1)} = PX$ С/П $PX = a_{3,p+1}^{(p)} \ldots a_{n,n+1}^{(p-1)} = PX$ С/П $PX = a_{n,n+1}^{(p)}$.

Для проверки можно воспользоваться данными табл. 13.

Программа 170/21. Вычисления коэффициентов p-й части бланка по методу Жордана—Гаусса с циклической перестановкой строк при решении системы из $n \leqslant 7$ линейных уравнений

↑	F7	÷	P3	Cx	P6	С/П	P4	F6	1	+	P6
F8	—	1	+	$x \neq 0$	P/—/	←	↑	→	XY	↑	←
F3	×	↑	F4	XY	—	БП	P1	→	↑	←	XY
↑	→	F6	1	—	P6	$x = 0$	P/—/	←	F3	P5	F4
↑	F7	÷	P3	F5	БП	P1					

Инструкция: $n = P8$, $a_{1p}^{(p-1)} = P7$, $a_{2p}^{(p-1)} = C1$, $a_{3p}^{(p-1)} = C2$, $a_{4p}^{(p-1)} = = C3$, $a_{5p}^{(p-1)} = C4$, $a_{6p}^{(p-1)} = C5$, $a_{7p}^{(p-1)} = C6$, $a_{1,p+1}^{(p-1)} = PX$ В/О С/П $PX = 0$, $a_{2,p+1}^{(p-1)} = PX$ С/П $PX = a_{1,p+1}^{(p)}$, $a_{3,p+1}^{(p-1)} = PX$ С/П $PX =$

$= a_{2,\,p+1}^{(p-1)}, \ldots, a_{n,\,p+1}^{(p-1)} = PX$ С/П $PX = a_{n-1,\,p+1}^{(p)}$, $a_{1,\,p+2}^{(p-1)} = PX$ С/П $PX = a_{n,\,p+1}^{(p)}, \ldots, a_{n,\,n+1}^{(p-1)} = PX$ С/П $PX = a_{n-1,\,n+1}^{(p)}$ С/П $PX = a_{n,\,n+1}^{(p)}$.

Для проверки можно воспользоваться данными табл. 12.

Программа 171/21. Вычисление цепного произведения матриц второго порядка $C = A \times B$

```
↑  F5  ×  P2 F8  ×  ←  ↑  F5  ×  P3 F8
×  ←  ↑  F7  ×  P6 F4  ×  ←  ↑  F7  ×
←  F4  ×  ↑  F3  +  P4  →  P7  →  ↑  F2
+  P5  →  ↑  F7  +  P7  →  ↑  F6  +  P8
С/П БП  P0
```

Инструкция: $a_{11} = P7$, $a_{12} = P8$, $a_{21} = P4$, $a_{22} = P5$, $b_{11} = C3$, $b_{12} = C2$, $b_{21} = C1$, $b_{22} = PX$ (В/О) С/П $PX = P8 = c_{12}$, $P7 = c_{11}$, $P4 = c_{21}$, $P5 = c_{22}$; коэффициенты b_{ij} следующего множителя ввести в регистры $C1$, $C2$, $C3$ и X и повторить выполнение программы.

Контрольный пример:

$$\begin{bmatrix} 5 & -2 \\ -3 & 4 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 8{,}2 & 2 \\ 1 & 8{,}1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 24{,}6 & 6 \\ 8{,}2 & 2 \end{bmatrix}.$$

Программа 172/21. Вычисление определителя Δ и алгебраических дополнений Δ_{11}, Δ_{12} и Δ_{13} матрицы A третьего порядка

```
→ F6  ↑  F8  ×  P2 F5    ↑  F3  ×  ↑  F2
—  ↑  ←  ×  P2 F4  ↑    F3  ×  P3 F7  ↑
F6  ×  ↑  F3  —  ↑  ←    ×  ↑  F2  +  P2
F7  ↑  P5  ×  P3 F4  ↑    F8  ×  ↑  F3  —
↑  ←  ×  ↑  F2  +  С/П
```

Инструкция: ($a_{21} = P4$, $a_{22} = P5$, $a_{23} = P6$, $a_{31} = P7$, $a_{32} = P8$) $a_{33} = P3$, $a_{13} = C2$, $a_{12} = C1$, $a_{11} = PX$ В/О С/П $PX = \Delta$, $C6 = \Delta_{13}$, $C5 = \Delta_{12}$, $C4 = \Delta_{11}$.

Контрольный пример: для матрицы с элементами $a_{11} = 4$, $a_{12} = 0$, $a_{13} = 1$, $a_{21} = 2$, $a_{22} = 5$, $a_{23} = 3$, $a_{31} = 4$, $a_{32} = 1$, $a_{33} = 6$ получим $\Delta = 90$, $\Delta_{13} = -18$, $\Delta_{12} = 0$, $\Delta_{11} = 27$ (время счета около 15 с).

Программа 173/21. Вычисление определителя Δ матрицы A второго порядка с комплексными элементами

```
↑  F7  ×  P2 F8  ×  ←  ↑  F8  ×  P3 F7
×  ←  ↑  F4  ×  P6  F5  ×  ←  ↑  F4  ×
←  F5  ×  ↑  F6  +  /—/  ↑  F3  +  ↑  F2
+  P2  →  ↑  →  XY  —  ↑  →  +  ↑  →
—  С/П
```

Инструкция: ($\mathrm{Re}\,a_{11} = P7$, $\mathrm{Im}\,a_{11} = P8$, $\mathrm{Re}\,a_{12} = P4$, $\mathrm{Im}\,a_{12} = P5$) $\mathrm{Re}\,a_{21} = C3$, $\mathrm{Im}\,a_{21} = C2$, $\mathrm{Re}\,a_{22} = C1$, $\mathrm{Im}\,a_{22} = PX$ В/О С/П $PX = \mathrm{Re}\,\Delta$, $P2 = \mathrm{Im}\Delta$.

Контрольный пример: для матрицы с элементами $a_{11}= 8 + j1$, $a_{12}= 1 + j2$, $a_{21}= 2 + j1$, $a_{22}= 10 + j3$ получим $\Delta = 77 + j29$.

Программа 174/21. Вычисление определенного интеграла I по составной формуле Симпсона

↑	F4	÷	P3	ПП		FBП	P8	ПП	/—/	4	×	↑
F8	+	P8	F4	2		—	P4	ПП	/—/	↑	F8	+
XY	+	P8	F4	$x=0$	F1	F3	×	3	÷	С/П	F3	
↑	F2	+	P2	⋯		В/О						

Инструкция: четное число разбиений $n = P4$, нижний предел интегрирования $a = P2$, ширина интервала $b - a = PX$ В/О С/П $PX = I$.

Контрольный пример: по данным контрольного примера к программе 132/34 тот же результат получим за 25 с.

Программа 175/21. Интегрирование табличной модели с постоянным шагом по формуле трапеций

P8	Cx	P4	С/П	↑	F8	XY	P8	+	2	÷	↑
F7	×	↑	F4	+	P4	БП	0				

Инструкция: $h = P7$, $y_0 = PX$ В/О С/П $PX = 0$, $y_1 = PX$ С/П $PX = I_1$, $y_2 = PX$ С/П $PX = I_2$ ... $y_n = PX$ С/П $PX = I_n$.

Контрольный пример: для табличной модели $y_i = 1$; 0,91; 0,83; 0,77; 0,71; 0,67; 0,63; 0,59; 0,55; 0,52; 0,5 с шагом $h = 0{,}1$ получим $I_{10} = 0{,}693$.

Программа 176/21. Решение дифференциального уравнения $y'(x) = f(x, y)$ модифицированным методом Эйлера

P8	ПП	1	ПП	1	+	С/П	БП	P0	P7	F2	↑
F3	+	P3	⋯	↑	F2	×	↑	F8	+	В/О		

Инструкция: заменить многоточие операторами вычисления функции $f(x, y)$ (при $x = P3$, $y = P7$, свободных регистрах *4*, *5*, *6* и стека памяти), $h/2 = P2$, $x_0 - h/2 = P3$, $y_0 = PX$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Контрольный пример: для уравнения $y'(x) = y - 4x + 3$, заменив в программе многоточие операторами F3 /—/ 4 × 3 + ↑ F7 +, при $x_0 = 0$, $y_0 = 3$, $h = 0{,}1$ получим $y_1 = 3{,}61$; $y_2 = 4{,}24205$; $y_3 = 4{,}898465$.

Программа 177/21. Решение дифференциального уравнения $y'(x) = f(x, y)$ методом Эйлера—Коши

P8	P7	ПП	P÷	P7	F2	↑	F3	+	P3	ПП	P÷
↑	F8	+	2	÷	С/П	БП	P0	⋯	↑	F2	×	
↑	F7	+	В/О									

Инструкция: заменить многоточие в программе операторами вычисления функции $f(x, y)$ (при $x = P3$, $y = P7$, свободных регистрах *4*, *5*, *6* и стека памяти), $h = P2$, $x_0 = P3$, $y_0 = PX$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$.

Для проверки можно использовать данные контрольного примера к предыдущей программе.

Программа 178/21. Решение дифференциального уравнения $y'(x) = f(x, y)$ методом Рунге—Кутта четвертого порядка

```
P7  P8  ПП F4  ПП  P÷  +  P4  ПП F4  +  P4
F7  +   P7 ПП  P÷  3   ÷  С/П F2  ↑  F3  +
P3  ··· ↑  F2  ×   ↑   F8 +  P7  F4  +  P4
В/О
```

Инструкция: заменить многоточие в программе операторами вычисления функции $f(x, y)$ (при $x = P3$, $y = P7$, свободных регистрах *5*, *6* и стека памяти), $h/2 = P2$, $x_0 = P3$, $3y_0 = P4$, $y_0 = PX$ В/О С/П $PX = y_1$ В/О С/П $PX = y_2$ В/О С/П $PX = y_3$ ...

Контрольный пример: для уравнения $y'(x) = y - 4x + 3$ при начальных условиях $x_0 = 0$, $y_0 = 3$, заменив многоточие операторами F3 /—/ 4 × 3 + ↑ F7 +, получим (время выполнения программы около 20 с) $y_i = 3{,}610341$; 4,242805; 4,899717; ...

Программа 179/21. Решение системы из двух дифференциальных уравнений $y'(x) = f_1(x, y, z)$ и $z'(x) = f_2(x, y, z)$ методом Эйлера

```
···  ↑  F2 ×  ↑  F7 +  P6 ···  ↑  F2 ×
↑..  F8 +  P8 F2 ↑  F3 +  P3 F6 P7 С/П
БП P0
```

Инструкция: заменить многоточие в программе операторами вычисления $f_1(x, y, z)$ и $f_2(x, y, z)$ соответственно (при $x = P3$, $y = P7$, $z = P8$, свободных регистрах *4*, *5* и стека памяти), $h = P2$, $x_0 = P3$, $y_0 = P7$, $z_0 = P8$ В/О С/П $PX = y_1$ С/П $PX = y_2$ С/П $PX = y_3$ ...

Контрольный пример: для решения дифференциального уравнения $y''(x) + 4y'(x) + 3y = 1$ при начальных условиях $x_0 = 0$, $y_0 = 0$, $y'_0 = 3$ представим его системой уравнений $y'(x) = z$, $z'(x) = 1 - 4z - 3y$, заменим многоточия в программе соответственно операторами F8 и F8 4 × P4 F7 3 × ↑ F4 + 1 XY — и при $h = 0{,}1$ получим $y = 0{,}3$; 0,49; 0,605 ... (при точных значениях $y_i = 0{,}2504$; 0,4203; 0,5321).

Программа 180/21. Решение системы из двух дифференциальных уравнений $y'(x) = f_1(x, y, z)$ и $z'(x) = f_2(x, y, z)$ модифицированным методом Эйлера

```
ПП  F÷  ×  P2 F3 +  P3  ПП  F÷  ÷  P2  ↑
F3  +   P3 F7 P8 F5 P6  С/П БП  P0 ··· ↑
F2  ×   ↑  F6 +  ←  ··· ↑   F2  ×  ↑  F8
+   P7  →  P5 F2 2  В/О
```

Инструкция: заменить многоточие операторами вычисления функций $f_1(x, y, z)$ и $f_2(x, y, z)$ соответственно (при $x = P3$, $y = P7$, $z = P8$ и свободных регистрах *4*, *C2*, ..., *C6*); $h/2 = P2$, $x_0 = P3$, $y_0 = P5 = P6$, $z_0 = P7 = P8$ В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$, $P3 = x_n$, $P8 = z_n$.

Контрольный пример: по данным контрольного примера к предыдущей программе получим y_i= 0,3; 0,43; 0,49115; ...

Программа 181/21. Решение системы из трех дифференциальных уравнений $y'(t) = f_1(t, y, z, x)$, $z'(t) = f_2(t, y, z, x)$ и $x'(t) = f_3(t, y, z, x)$ методом Эйлера

...	↑	F2	×	↑	F6	+	←	...	↑	F2	×
↑	F7	+	←	...	↑	F2	×	↑	F8	+	P8
F2	↑	F3	+	P3	→	P7	→	P6	С/П	БП	P0

Инструкция: заменить многоточия в программе операторами вычисления функций $f_1(t, y, z, x)$, $f_2(t, y, z, x)$ и $f_3(t, y, z, x)$ соответственно (при t = *P3*, y = *P6*, z = *P7*, x = *P8* и свободных регистрах *4*, *5*, *C3*, ..., *C6*); h = *P2*, t_0= *P3*, y_0= *P6*, z_0= *P7*, x_0= *P8* В/О С/П $PX = y_1$ С/П $PX = y_2$ ... С/П $PX = y_n$, *P3* = t_n, *P7* = z_n, *P8* = x_n.

Контрольный пример: для решения дифференциального уравнения $d^3y/dt^3 + 6d^2y/dt^2 + 11dy/dt + 6y = 0$ при начальных условиях $t_0 = 0$, $y_0 = 0$, $y'_0 = 3$, $y''_0 = -11$ представим уравнение системой уравнений первого порядка $y'(t) = z$, $z'(t) = x$, $x'(t) = -6x - 11z - 6y$ и при $h = 0{,}1$ получим y_i= 0,3; 0,49; 0,603; ... (точные значения y_i = = 0,2501; 0,4183; 0,5263).

Программа 182/21. Вычисление гиперболических и обратных гиперболических функций вещественного аргумента x

e^x	↑	$1/x$	—	2	÷	С/П	e^x	↑	$1/x$	+	2
÷	С/П	↑	2	×	e^x	1	—	2	+	÷	С/П
P8	x^2	1	+	$\sqrt{\ }$	↑	F8	+	ln	С/П	P8	x^2
1	—	$\sqrt{\ }$	↑	F8	+	ln	С/П	↑	1	+	2
—	/—/	÷	ln	2	÷	С/П					

Инструкция: $x = PX$ В/О С/П PX = sh$x$ или БП F1 С/П PX = = ch$x$ или БП P× С/П PX = th$x$ или БП P4 С/П PX = arsh$x$ или БП F /—/ С/П PX = arch$x$ или БП PCx С/П PX = arth$x$.

Контрольный пример: sh0,5 = 0,5210951; arsh (sh0,5) = 0,499998; ch0,5 = 1,127625; arch (ch0,5) = 0,499998; th0,5 = 0,462117; arth (th0,5) = 0,4999992.

Программа 183/21. Вычисление гамма-функции Г (x)

P2	1	+	×	P3	F2	2	+	P4	1	e^x	÷
X^y	↑	F3	÷	P3	π	2	×	↑	F4	÷	$\sqrt{\ }$
↑	F3	×	P3	0	,	7	↑	F4	÷	1	XY
—	2	4	÷	↑	F4	÷	1	+	1	2	÷
↑	F4	÷	1	+	↑	F3	×	С/П	БП	P0	

Инструкция: $x = PX$ (В/О) С/П PX = Г (x), точность результата около пяти верных цифр.

Контрольный пример: Г (1) = 1, Г (8) = 5039,945.

Программа 184/21. Вычисление нормальных эллиптических интегралов Лежандра первого F (k, φ) и второго E (k, φ) рода

P4	1	P6	F8	P3	↑	F4	÷	P2	ПП	P7	1
ПП	4	4	ПП	4	2	БП	P2	F6	3	÷	↑
F2	×	С/П	×	↑	F6	+	P6	F4	1	—	P4
$x \neq 0$	P÷	↑	F2	×	P3	F3	sin	↑	F7	×	x^2
1	XY	—	$\sqrt{\ }$	$1/x$	В/О						

Инструкция: для вычисления интеграла первого рода $k = P7$, $\varphi = P8$, четное число разбиений интервала интегрирования $n = PX$ (В/О) С/П $PX = F$ (k, φ); для вычисления интеграла второго рода исключить оператор $1/x$ в конце подпрограммы; для вычисления полных интегралов принять $\varphi = \pi/2$.

Контрольный пример: при $n = 10$ время вычисления интегралов F (sin 45°, π/2) = 1,854074 или E (sin 45°, π/2) = 1,350643 составляет около 70 с.

Программа 185/21. Вычисление функций Бесселя вещественного аргумента $J_n(x)$ порядка n

P8	XY	P7	Cx	P2	P4	4	$1/x$	—	↑	π	×
9	÷	P5	sin	↑	F8	×	cos	P6	F5	↑	F7
×	cos	↑	$\overline{\text{F6}}$	×	↑	F2	$\overline{+}$	P2	F4	1	+
9	—	$x \neq 0$	F7	XY	БП	↑	F2	9	÷	С/П	

Инструкция: при четном порядке $n = P7$, $x = PX$ В/О С/П $PX = J_n(x)$; при нечетном порядке n предварительно поменять местами в программе подчеркнутые операторы.

Контрольный пример: время вычисления $J_4(2) = 0{,}03399572$ составляет около 110 с.

Программа 186/21. Вычисление коэффициентов ортогональных многочленов Лежандра $P_n(x)$

P2	1	P6	P8	+	P3	2	P7	×	3	—	P5
P4	↑	F8	×	↑	F6	÷	P8	F6	1	+	P6
F4	2	—	$x < 0$	P2	F8	С/П	F3	1	—	1	—
P3	×	↑	F8	×	↑	F7	÷	↑	F5	÷	/—/
P8	F7	2	+	P7	F5	2	—	P5	БП	,	

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-2}$ С/П $PX = a_{n-4}$ ... С/П $PX = 0$, $P2 = n$.

Контрольный пример: $P_7(x) = 26{,}8125x^7 - 43{,}3125x^5 + 19{,}687x^3 - 2{,}1875x$.

Программа 187/21. Вычисление коэффициентов ортогональных многочленов Эрмита $H_n(x)$

P7	1	P8	+	P4	2	P5	F8	С/П	F4	1	—
1	—	P4	×	↑	F8	×	↑	F5	÷	/—/	P8
F5	2	+	P5	БП	F1						

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-2}$ С/П $PX = a_{n-4}$ ... С/П $PX = 0$, $P7 = n$.

Контрольный пример: $H_{10}(x) = x^{10} - 45x^8 + 630x^6 - 3150x^4 + 4725x^2 - 945$.

Программа 188/21. Вычисление коэффициентов ортогональных многочленов Лагерра $L_n(x)$

```
P7 0  P8 π  ×  cos P6 С/П F8 1  +  P8
F7 —  x²  ↑ F6 ×  ↑ F8  ÷  /—/ БП P1
```

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-1}$ С/П $PX = a_{n-2}$ ... С/П $PX = 0$, $P7 = n$.

Контрольный пример: $L_7(x) = -x^7 + 49x^6 - 882x^5 + 7350x^4 - 29400x^3 + 52920x^2 - 35280x + 5040$.

Программа 189/21. Вычисление коэффициентов ортогональных многочленов Чебышева $T_n(x)$ степени $n > 2$

```
P2  2 P4  P8  —  P3  F4  2  ×  P4 F3 1
P5  —  P3 x = 0 P1 Г4 С/П /—/  ↑  F2 × 4
÷  ↑ F5  ×  С/П F8 ÷  P4  F8 1  P5 +
P8 P7 2  —  P6  F7 1  +  P7 F2 —  ↑
F5 × P5  F6  1  —  P6 x = 0 ВП F4 БП ÷
```

Инструкция: $n = PX$ В/О С/П $PX = a_n$ С/П $PX = a_{n-2}$ ... С/П $PX = 0, P2 = n$ (дробные результаты округлить до целых).

Контрольный пример: $T_{10}(x) = 512x^{10} - 1280x^8 + 1120x^6 - 400x^4 + 50x^2 - 1$.

Программа 190/21. Вычисление корней x_0 и ординат x_{max} и x_{min} экстремумов ± 1 многочленов Чебышева степени $n > 2$

```
P7 Cx  P4  P5  π  ↑  F7  —  P3 F3  2  ×
↑  F4  ×  cos  1 С/П ПП ВП F4 2  ×  1
+  ↑  F3  ×  cos  1  /—/ С/П F4 1  +  P4
ПП ВП БП  ↑  F5  2  ×  1  +  ↑  F3  ×
 2  ÷  cos P6  F5  1  +  P5  F6 0  С/П В/О
```

Инструкция: $n = PX$ В/О С/П $PX = 1, PY = 1$ С/П $PX = 0, PY = x_{10}$ С/П $PX = -1, PY = x_{1max}$ С/П $PX = x_{20}$ С/П $PX = 1$, $PY = x_{1min}$ ... С/П $PX = 0, PY = x_{n0}$ С/П $PX = \pm 1$, $PY = \pm 1$.

Контрольный пример: для $n = 7$ получим $x_{max} = 1, x_{10} = 0{,}951056$; $x_{1min} = 0{,}8090917$; $x_{20} = 0{,}587785$; $x_{1max} = 0{,}309017$; $x_{30} = 0$, $x_{2min} = -0{,}309017$; $x_{40} = -0{,}587785$; $x_{2max} = -0{,}809016$; $x_{50} = -0{,}951056$; $x_{min} = -1$.

Программа 191/21. Вычисление функции ошибок erf x с четырьмя верными знаками в интервале $0 \leqslant x \leqslant 0{,}51$

```
x²  P7 0  ,  0  9  ×  /—/ 1  ,  6  2
1  +  √  /—/  ↑  F7 ×  eˣ  1  XY  —  √
С/П БП P0
```

Инструкция: $x = PX$ (В/О) С/П $PX = \text{erf}\, x$, $P7 = x^2$.

Контрольный пример: erf 0,01 = 0,01127829 и erf 0,4 = 0,4284389 при табличных значениях [6] соответственно 0,0113 и 0,4284.

Программа 192/21. Вычисление коэффициентов c_i бинома Ньютона степени $n < 70$

```
P7 1  +  P7  Cx P4 1  P5  P6 С/П F4   1
+  P4 ↑  F5   ×  P5 F7 1   —  P7  x≠0 P1
↑  F6 ×  P6 ↑ F5    ÷  БП 1
```

Инструкция: $n = PX$ В/О С/П $PX = c_1 = 1$ С/П $PX = c_2 \ldots$ С/П $PX = c_n$ С/П $PX = c_{n+1} = 1$.

Контрольный пример: для $n = 15$ получим коэффициенты разложения $(a + b)^{15} = a^{15} + 15a^{14}b + 105a^{13}b^2 + 455a^{12}b^3 + 1365a^{11}b^4 + + 3003a^{10}b^5 + 5005a^9b^6 + 6435a^8b^7 + 6435a^7b^8 + \ldots + b^{15}$.

Программа 193/21. Преобразование позиционных представлений чисел N_n в N_m для систем счисления с основаниями $n \leqslant 10$, $m = 10$ или $n = 10$, $m \leqslant 10$*

```
P2 1  P4 Cx P5 F2 ↑  F8 ÷   (9 × 5
— 9 ÷) 1  ВП 7  XY + XY   — P6 ↑
F8 × ↑  F2 XY —  ↑  F4 ×   ↑  F5 +
P5 F4 ↑  F7 ×  P4 F6 P2 x=0 ↑  F5 С/П
```

Инструкция: $n = P7$, $m = P8$, $N_n = PX$ В/О С/П $PX = N_m$. Для преобразования представлений чисел с $n < 10$, $m < 10$ следует найти десятичное представление числа, а затем преобразовать его в представление в заданной системе счисления.

Контрольный пример: для преобразования $N_2 = 10101011$ в N_8 находим $(2 = P7, 10 = P8, 10101011 = PX) N_{10} = 171$, а затем (при $10 = P7, 8 = P8, 171 = PX$) искомое $N_8 = 253$ (время счета соответственно 60 и 25 с).

Программа 194/21. Формирование последовательности квазилинейных целых чисел z_i в интервале $(0,b)$ и дробных x_i в интервале $(0,1)$ с равномерным распределением**

```
F7 ↑  F2  ×  P4  ↑  F3 ÷  (9   ×  5  —
9  ÷) 1   ВП 7   XY + XY —   ↑  F3 ×
↑  F4 XY  —  P7  ↑  F3 ÷  С/П БП P0
```

Инструкция: целые числа $a = P2$, $b = P3$, $z_0 = P7$ В/О С/П $PX = x_1$, $PY = z_1$ С/П $PX = x_2$, $PY = z_2$ С/П ...

Контрольный пример: при выборе по методу Коробова $a = 1298$, $b = 2027$ и $z_0 = 1$ получим $x_i = 0{,}6403552$; $0{,}1810557$; $0{,}01036013$; $0{,}4474592$; ...; $z_i = 1298{,}367{,}21{,}907$, ... (время счета для микрокалькуляторов с округлением по дополнению и отбрасыванием около 8 и 5 с соответственно).

Программа 195/21. Формирование последовательности квазислучайных дробных x_i и целых z_i с равномерным распределением в интервалах $(0,1)$ и $(0,b)$, q_i с распределением Релея и y_i с нормальным распределением при заданной дисперсии σ^{2}**

---

* Фрагмент в круглых скобках $(9 \times 5 - 9 \div)$ блока выделения целой части чисел программы необходим только для микрокалькуляторов с округлением по дополнению результатов сложения.

** См. примечание к программе 193/21.

```
F6 P5 ↑ F2 × ↑ ← F3 ÷ (9 × 5
— 9 ÷) 1 ВП 7 XY + XY — ↑ F3
× ↑ → XY — ↑ P6 F3 ÷ P7 1/x ln
2 × ↑ F4 × √ P8 F5 ↑ F3 ÷ 2
× ↑ π × e^x F8 × С/П БП P0
```

Инструкция: целые числа $a = P2$, $b = P3$, $z_0 = P6$, дисперсия $\sigma^2 = P4$ (В/О) С/П $PX = y_i$, $P6 = z_i$, $P7 = x_i$, $P8 = q_i$.

Контрольный пример: для $a = 1298$, $b = 2027$, $z_0 = \sigma^2 = 1$ получим $y_i = 0{,}00292665$; $-1{,}427117$; $2{,}743927$; $0{,}0824948$; ..., $z_i =$ $= 1298{,}367{,}21{,}907$, ..., $x_i = 0{,}6403552$; $0{,}1810557$; $0{,}01036013$; $0{,}4474592$; ..., $q_i = 0{,}9441728$; $1{,}848756$; $3{,}023173$; $1{,}268201$; ... (время счета около 15 с).

Программа 196/21. Ускоренное формирование последовательности квазислучайных чисел x_i с равномерным распределением в интервале (0,1)*

```
F7 ↑ π + e^x P6 (9 × 5 — 9 ÷)
1 ВП 7 XY + XY — ↑ F6 XY — P7
С/П БП P0
```

Инструкция: $x_0 = P7$ (В/О) С/П $PX = x_i$.

Контрольный пример: при $x_0 = 0{,}5$ получим $x_i =$ 0,152546; 0,954182; 0,085857; 0,215263; 0,698806; ... (время счета около 6 с).

Программа 197/21. Моделирование символами $y_i \in (0,1)$ двух исходов случайного события с вероятностями $p(1)$ и $p(0) = 1 - p(1)$*

```
F7 ↑ π + e^x P6 (9 × 5 — 9 ÷)
1 ВП 7 XY + XY — ↑ F6 XY — P7
↑ F8 — x<0 Р/—/ Сх БП 5 1 С/П БП P0
```

Инструкция: $x_0 = P7$, $p(1) = P8$ (В/О) С/П $PX = y_i$.

Контрольный пример: для $x_0 = p(1) = 0{,}5$ получим $y_i = 0$, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, ... (время счета около 9 с).

Программа 198/21. Моделирование цифрами $y_i \in (1, 2, \ldots, k)$ последовательности равновероятных k исходов случайного события*

```
F8 1/x P5 F8 1 P4 + P3 F7 ↑ π +
e^x P6 (9 × 5 — 9 ÷) 1 ВП 7 XY
+ XY — ↑ F6 XY — P7 F3 1 — P3
F4 ↑ F5 — P4 ↑ F7 XY — x≥0 Р/—/ F3
С/П БП 0
```

Инструкция: $x_0 = P7$, $k = P8$ (В/О) С/П $PX = y_i$.

Контрольный пример: для $x_0 = 0{,}5$; $k = 6$ получим $y_i = 1$, 6, 1, 2, 5, 4, 6, 2, 4, 5, 3, 3, ... (время счета от 10 до 23 с).

Программа 199/21. Оценка среднего $\tilde{m}_1$ и дисперсии $\tilde{\sigma}^2$ выборки из n случайных чисел x_i с учетом постоянной составляющей

---

* См. примечание к программе 193/21.

P6	F7	$x\neq 0$	F4	1	—	←	F4	XY	÷	P2	×	
↑	F5	XY	—	↑	→	÷	P3	F2	↑	F8	+	
С/П	P4	P5	F6	↑	F8	—	↑	F4	+	P4	F1	
×	↑	F5	+	P5	F7	1	+	P7	С/П	БП	F,	

Инструкция: $0 = P7$, приближенное среднее $a = P8$, $x_1 = PX$ В/О С/П $PX = 1, x_2 = PX$ С/П $PX = 2 \ldots x_n = PX$ С/П $PX = n$ В/О С/П $PX = \tilde{m}_1$, $P3 = \sigma^2$, $P2 = \tilde{m}_1$ (среднее для $\Delta x_i = x_i - a$).

Контрольный пример: для $x_i = 1000{,}8$; 1001,2; 998,5; 999,4; 1000,5 при $a = 1000$ получим $\tilde{m}_1 = a + \tilde{m}_1 = 1000{,}8$; $\tilde{\sigma}^2 = 1{,}227$; $\tilde{m}_1 = 0{,}08$; при $a = 0$ получим $\tilde{m}_1 = 1000{,}08$; $\tilde{\sigma}^2 = 1{,}175$ с погрешностью около 5% (время счета около 5 с).

Программа 200/21. Вычисление оценок $\tilde{m}_k$ четырех начальных моментов выборки из n случайных чисел x_i

P2	F8	$x\neq 0$	P3	$1/x$	↑	←	×	С/П	←	×	С/П	
←	×	С/П	←	×	С/П	←	0	←	0	←	0	
←	←	←	F2	↑	×	P2	×	P3	×	P4	←	
+	←	↑	F2	+	←	↑	F3	+	←	↑	F4	
+	←	←	←	F8	1	+	P8	С/П	БП	F,		

Инструкция: $0 = P8$, $x_1 = PX$ В/О С/П $PX = 1, x_2 = PX$ С/П $PX = 2 \ldots x_n = PX$ С/П $PX = n$ В/О С/П $PX = \tilde{m}_1$ С/П $PX = \tilde{m}_1$ С/П $PX = \tilde{m}_3$ С/П $PX = \tilde{m}_4$ (время обработки отсчета около 8 с).

Контрольный пример: для $x_i = 1000{,}8$; 1001,2; 998,5; 999,4; 1000,5 получим $\tilde{m}_1 = 1000{,}8$; $\tilde{m}_2 = 1000160$; $\tilde{m}_3 = 1{,}000242 \cdot 10^9$; $\tilde{m}_4 = 1{,}000325 \cdot 10^{12}$.

Программа 201/21. Вычисление суммы $S = a + b$ слагаемых, содержащих до 28 значащих разрядов одинакового порядка

↑		F5	+	P5	1	ВП	7	P2	—	$x \geqslant 0$	F×	P5
F4		1	+	P4	F4	↑	←	+	P4	↑	F2	—
$x \geqslant 0$		F5	P4	F8	1	+	P8	F8	↑	←	+	P8
↑		F2	—	$x \geqslant 0$	FCx	P8	F7	1	+	P7	F7	↑
←		+	P7	С/П	БП	P0						

Инструкция: дробные слагаемые предварительно уравнять по порядку и представить в показательной форме с целочисленными мантиссами и отрицательным порядком; разбить разряды мантисс (начиная с младших) на блоки $a_3\ a_2\ a_1\ a_0$ и $b_3\ b_2\ b_1\ b_0$, содержащие по 7 разрядов; $a_3 = P7$, $a_2 = P8$, $a_1 = P4$, $a_0 = P5$, $b_3 = C3$, $b_2 = C2$, $b_1 = C1$, $b_0 = PX$ (В/О) С/П $PX = P7 = S_3$, $P8 = S_2$, $P4 = S_1$, $P5 = S_0$ (S_i — семиразрядные блоки суммы).

Контрольный пример: для сложения 98765432102234567,89 и 1234567890987,654321 выполнить $98 = P7$; $7654321 = P8$; $123456 = P4$; $7890000 = P5$; $0 = C3$; $12345 = C2$; $6789098 = C1$; $7654321 = PX$ В/О С/П $PX = 98$; $P8 = 7666666$; $P4 = 6912555$; $P5 = 5544321$ или (с учетом порядка $n = -6$) $S = 98766666669125555{,}544321$.

Программа 202/21. Вычисление частных произведений при умножении двух чисел, содержащих до 12 значащих цифр.

↑	F6	×	→	F5	×	P7	F4	×	P8	F3	↑
F6	×	↑	F7	+	→	F3	↑	F4	×	P7	F5
×	↑	F8	+	P8	F2	↑	F4	×	↑	F8	+
→	F2	↑	F5	×	↑	F7	+	→	F2	↑	F4
×	С/П	←	С/П	←	С/П	←	С/П	←	С/П		

Инструкция: дробные множители a и b представить в показательной форме с целыми мантиссами и показателями порядка n_a и n_b, разбить разряды (начиная с младших) мантисс на блоки $a_2\ a_1\ a_0$ и $b_2\ b_1\ b_0$, содержащие по четыре разряда, $a_2 = P4$, $a_1 = P5$, $a_0 = P6$, $b_2 = P2$, $b_1 = P3$, $b_0 = PX$ В/О С/П $PX = S_4 = a_2b_2$ С/П $PX = S_3 = a_2b_1 + a_1b_2$ С/П $PX = S_2 = a_2b_0 + a_1b_1 + a_0b_2$ С/П $PX = S_1 = a_1b_0 + a_0b_1$ С/П $PX = S_0 = a_0b_0$; для получения произведения, содержащего до 24 значащих цифр, сложить S_i со сдвигом на четыре разряда при помощи программы 201/21.

Контрольный пример: для множителей 123456789098 и 98765432101 выполнить $1234 = P4$; $5678 = P5$; $9098 = P6$; $987 = P2$; $6543 = P3$; $2101 = PX$ В/О С/П $PX = 1217958$ С/П $PX = 13678248$ С/П $PX = 40961746$ С/П $PX = 71457692$ С/П $PX = 19114898$; при помощи программы 201/21 находим $S = 12193262344889196{,}034898$.

Программа 203/21. Вычисление числа $A_n^m = n(n-1)\ldots(n-m+1)$ размещений из n по m элементов (n и m — положительные целые числа)

—	P7	XY	→	1	←	1	—	→	P2	F7	—
$x \neq 0$	P3	F2	×	БП	↑	F2	С/П	БП	P0		

Инструкция: $n = PY$, $m = PX$ (В/О) С/П $PX = A_n^m$.

Контрольный пример: $A_4^2 = 12$; $A_5^4 = 120$; $A_{10}^5 = 30240$.

Программа 204/21. Вычисление числа $C_n^m = n!/(m!\,(n-m)!)$ сочетаний из n по m элементов (n и m — положительные целые числа)

P3	÷	P5	XY	P4	F3	1	—	P3	$x \neq 0$	P4	F4
1	—	P4	↑	F5	×	↑	F3	÷	P5	БП	↑
F5	С/П	БП	P0								

Инструкция: $n = PY$, $m = PX$ (В/О) С/П $PX = C_n^m$.

Контрольный пример: $C_4^2 = 6$; $C_5^4 = 5$; $C_{10}^5 = 252$.

СПИСОК ЛИТЕРАТУРЫ

1. **Бахвалов Н. С.** Численные методы.— М.: Наука, 1975.— 632 с.
2. **Воеводин В. В.** Численные методы алгебры.— М.: Наука, 1968.— 246 с.
3. **Годунов С. К., Рябенький В. С.** Разностные схемы.— М.: Наука, 1977.— 440 с.
4. **Демидович Б. П., Марон И. А.** Основы вычислительной математики.— М.: Наука, 1970.— 664 с.
5. **Каширский И. С., Трохименко Я. К.** Обобщенная оптимизация электронных схем.— Киев: Техніка, 1980.— 192 с.
6. **Корн Г., Корн Т.** Справочник по математике.— М.: Наука, 1974.— 831 с.
7. **Мак-Кракен Д., Дорн У.** Численные методы и программирование на ФОРТРАНе.— М.: Мир, 1977.— 584 с.
8. **Периферийное** устройство для калькулятора моделирует СПЗУ.— Электроника: Пер. с англ.— М.: Мир, 1982, № 11, с. 115—116.
9. **Программное** обеспечение ЭВМ МИР-1 и МИР-2.1. Численные методы/ В. М. Глушков, И. Н. Молчанов, Б. Н. Брусникина и др.— Киев: Наук. думка, 1976.— 280 с.
10. **Трохименко Я. К.** Метод обобщенных чисел и анализ линейных цепей.— М.: Сов. радио, 1972.— 312 с.
11. **Трохименко Я. К., Любич Ф. Д.** Инженерные расчеты на микрокалькуляторах.— Киев: Техніка, 1980.— 384 с.
12. **Трохименко Я. К., Любич Ф. Д.** Искусство программирования программируемых микрокалькуляторов. 1. Входные языки.— Изв. вузов. Радиоэлектроника, 1983, № 6, с. 70—74.
13. **Трохименко Я. К., Любич Ф. Д.** Радиотехнические расчеты на микрокалькуляторах.— М.: Радио и связь, 1983.— 256 с.
14. **Трохименко Я. К., Каширский И. С., Ловкий В. К.** Проектирование радиотехнических цепей на инженерных ЭЦВМ.— Киев: Техніка, 1976.— 276 с.
15. **Форсайт Дж., Малькольм М., Моулер К.** Машинные методы математических вычислений.— М.: Мир, 1980.— 320 с.
16. **Хейс А.** Расчет резистивных аттенюаторов на калькуляторе HP-41C.— Электроника: Пер. с англ.— М.: Мир, 1982, № 1, с. 97—99.
17. **Хеминг Р. В.** Численные методы.— М.: Наука, 1968.— 320с.
18. **Химмельблау Д.** Прикладное нелинейное программирование. — М.: Мир, 1976.— 534 с.
19. **Цветков А. Н.** Прикладные программы для микро-ЭВМ «Электроника Б3-21».— М.: Финансы и статистика, 1982.— 128 с.
20. **Численные** методы / Н. И. Данилина, Н. С. Дубровская, О. П. Кваша и др.— М.: Высш. шк., 1976.— 368 с.

ОГЛАВЛЕНИЕ

Глава 7. Статистические расчеты

Глава 8. Оптимизация решений инженерных задач

Приложение

Список литературы

БИБЛИОТЕКА ИНЖЕНЕРА

Ярослав Карпович Трохименко,
д-р техн. наук,
Феликс Дмитриевич Любич,
канд. техн. наук

Инженерные расчеты
на программируемых микрокалькуляторах

Редактор
Л. О. Полянская
Оформление художника
Л. А. Дикарева
Художественный редактор
В. С. Шапошников
Технический редактор
С. В. Иванус
Корректор
Н. В. Тарабан

Информ. бланк № 2678

Сдано в набор 25.05.84. Подписано в печать 21.11.84. БФ 40468.
Формат 60×90¹/₁₆. Бумага типогр. № 2. Гарн. лит.
Печ. выс. Усл. печ. л. 20,5. Усл. кр.-отт. 20,7.
Уч.-изд. л. 22,36. Тираж 30 000 экз. Зак. 4-713.
Цена в переплете тип 5 — 1 р. 30 к. (20 000 экз.), цена в переплете тип 7 — 1 р. 40 к. (10 000 экз.).

Издательство «Тэхника»,
252601, Киев, 1, Крещатик, 5.

Книжная фабрика имени М. В. Фрунзе, 310057, Харьков-57,
ул. Донец-Захаржевского, 6/8.