УЧЕБНОЕ ПОСОБИЕ

О. Г. КРУПЕННИКОВ
Д. В. КРАВЧЕНКО

КУРС ЛЕКЦИЙ ПО ОСНОВАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ ЗАДАЧ МАШИНОСТРОЕНИЯ

Ульяновск 2006

Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

О. Г. Крупенников, Д. В. Кравченко

КУРС ЛЕКЦИЙ ПО ОСНОВАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ ЗАДАЧ МАШИНОСТРОЕНИЯ

Учебное пособие

для студентов, обучающихся по специальностям 151001 – «Технология машиностроения», 150201 – «Машины и технология обработки металлов давлением», 190201 – «Автомобиле- и тракторостроение», 190601 – «Автомобили и автомобильное хозяйство»

Ульяновск 2006

УДК 681.3:519.68(075)
ББК 22.18я7
К84

Рецензенты: канд. техн. наук, зам. начальника производства – главный специалист ОАО «Автодеталь-Сервис» С. Е. Ведров; кафедра «Математическое моделирование технических систем» Ульяновского государственного университета.

Научный редактор: канд. техн. наук, профессор Карев Е. А.

Утверждено редакционно-издательским советом университета в качестве учебного пособия.

Крупенников О. Г.

К84 Курс лекций по основам алгоритмизации и программирования задач машиностроения: учебное пособие / О. Г. Крупенников, Д. В. Кравченко. – Ульяновск: УлГТУ, 2006. – 144 с.

ISBN 5-89146-803-4

Пособие разработано в соответствии с типовой и рабочей программами дисциплины «Информатика» для студентов первого курса всех форм обучения специальностей 151001, 150201, 190201, 190601.

Освещены общие правила построения блок-схем алгоритмов различных структур (линейной, ветвления, цикла) на примерах решения различных технологических задач, вопросы реализации ряда численных методов применительно к нахождению значений выходных параметров технологических систем. Рассмотрены примеры практической реализации этих методов для решения задач машиностроения с применением интегрированной среды программирования TURBO PASCAL 7.0.

Пособие написано на кафедре «Технология машиностроения» УлГТУ.

УДК 681.3:519.68(075)
ББК 22.18я7

ISBN 5-89146-803-4

Учебное издание

КРУПЕННИКОВ Олег Геннадьевич
КРАВЧЕНКО Дмитрий Валерьевич

КУРС ЛЕКЦИЙ ПО ОСНОВАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ ЗАДАЧ МАШИНОСТРОЕНИЯ

Учебное пособие

Редактор *О. А. Фирсова*

Подписано в печать 16.01.2006. Формат 60×84/16.
Бумага офсетная. Печать трафаретная.
Усл. печ. л. 8,37. Уч.-изд. л. 8,10.
Тираж 300 экз. Заказ
Ульяновский государственный технический университет
432027, Ульяновск, ул. Сев. Венец, 32.
Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Данное учебное пособие написано в соответствии с типовой и рабочей программами дисциплин «Информатика» для студентов первого и второго курсов машиностроительных специальностей всех форм обучения.

Изучение дисциплины «Информатика» дает возможность решения большого спектра задач в области сбора, накопления, передачи, обмена и обработки информации. При этом основной акцент делается на необходимость использования средств вычислительной техники.

Информатика является одной из базовых дисциплин кафедры «Технология машиностроения» и необходима для изучения таких специальных дисциплин, как, например, «Основы технологии машиностроения», «Технология машиностроения», «Автоматизация производственных процессов в машиностроении», «Компьютерное обеспечение машиностроительного производства» и др.

Как показывает многолетняя практика, у многих студентов возникают определённые трудности в понимании ряда теоретических вопросов дисциплины. Это вопросы связанные с построением алгоритмических структур решения технологических задач (правила построения) и реализации с помощью ЭВМ ряда численных методов нахождения приближенных значений выходных параметров технологических систем с заданной точностью при решении машиностроительных задач.

Учитывая вышесказанное, работа студентов с этим пособием позволит снять обозначенные трудности и дать четкое представление о правилах алгоритмизации и практической реализации методов нахождения выходных параметров технологических систем с нелинейной и линейной характеристиками, численного интегрирования и дифференцирования, оптимизации и аппроксимации этих параметров в задачах машиностроения с применением интегрированной среды программирования TURBO PASCAL 7.0.

1. АЛГОРИТМИЗАЦИЯ ЗАДАЧ МАШИНОСТРОЕНИЯ

1.1. Алгоритм и его свойства

Алгоритм – это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

Алгоритм должен содержать конечную последовательность шагов или операций, однозначно определяющих процесс переработки исходных и промежуточных данных в искомый результат.

При разработке алгоритма решения той или иной задачи необходимо учитывать обеспечения этим алгоритмам следующих свойств:

- определенность;
- результативность;
- массовость;
- дискретность.

Определенность – это свойство алгоритма, обеспечивающее его однозначность, исключение произвольности толкования любого из предписаний и заданного порядка исполнения.

Результативность – это свойство алгоритма, обеспечение которого позволит через определенное число шагов получить конечный результат или сообщение о невозможности по тем или иным причинам решения поставленной задачи.

Массовость – это свойство алгоритма, обеспечивающее возможность использования этого алгоритма для решения n-го количества типовых задач.

Дискретность – это свойство алгоритма, обеспечивающее возможность разбиения вычислительного процесса на отдельные самостоятельные этапы.

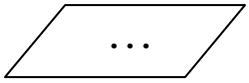
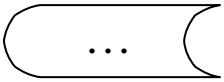
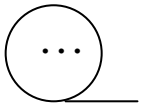
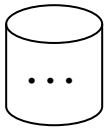
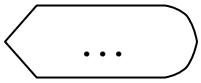
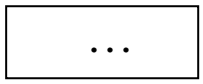
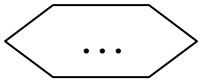
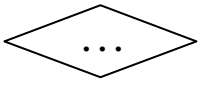
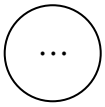
Алгоритм может быть представлен в описательном виде (на словах) или с помощью зарезервированных графических символов: данных, процесса, линий, специальных основных и специфических.

1.2. Графические символы, используемые для построения алгоритмических структур

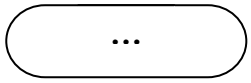
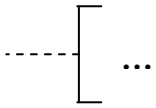
Для построения алгоритмов можно использовать следующие графические символы, представленные в таблице 1.1.

Таблица 1.1

Символы языка графических символов

№п/п	Наименование символа	Обозначение символа
1	2	3
1	Данные (основной символ данных)	
2	Запоминаемые данные (основной символ данных)	
3	Запоминающее устройство с последовательной выборкой (специфический символ данных)	
4	Запоминающее устройство с прямым доступом (специфический символ данных)	
5	Документ (специфический символ данных)	
6	Ручной ввод (специфический символ данных)	
7	Дисплей (специфический символ данных)	
8	Процесс (основной символ процесса)	
9	Подготовка (специфический символ процесса)	
10	Решение (специфический символ процесса)	
11	Линия (основной символ линий)	
12	Пунктирная линия (специфический символ линий)	
13	Соединитель (специальный символ)	

Окончание табл. 1.1

1	2	3
14	Терминатор (специальный символ)	
15	Комментарий (специальный символ)	
Примечание: полный список графических символов представлен в ГОСТ 19.701-90 (ИСО 5807-85) ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения		

Графический символ «Данные» отображает данные, носитель которых не определен. В этот графический символ вписываются, например, имена переменных входных данных при вводе или имена переменных выходных данных при выводе.

Графический символ «Запоминаемые данные» отображает хранимые данные в виде, пригодном для обработки, носитель данных не определен.

Графический символ «Запоминающее устройство с последовательной выборкой» отображает данные, хранящиеся в запоминающем устройстве с последовательным доступом (магнитная лента, кассета с магнитной лентой, магнитофонная кассета).

Графический символ «Запоминающее устройство с прямым доступом» отображает данные хранящиеся в запоминающем устройстве с прямым доступом (магнитный диск, магнитный барабан, гибкий магнитный диск).

Графический символ «Документ» отображает данные, представленные на носителе в удобочитаемом виде (микрофильм, рулон ленты с итоговыми данными, бланки ввода данных).

Графический символ «Ручной ввод» отображает данные, вводимые вручную во время обработки с устройства любого типа (клавиатура).

Графический символ «Дисплей» отображает данные (входные или выходные), представленные в человекочитаемой форме на носителе в виде отображающего устройства (экран).

Графический символ «Процесс» отображает функцию обработки данных любого вида. В этот графический символ, например, может быть вписана расчетная зависимость (формула).

Графический символ «Подготовка» используется в алгоритмических структурах повторения (цикла) и указывает на начало цикла. В этот графический символ вписывается имя переменной параметра цикла, который будет из-

меняться с шагом приращения равным «+ 1» или «– 1», начальное и конечное значение параметра цикла. Например: К = 12; 30, где К – имя переменной параметра цикла, а цифры 12 и 30, соответственно, начальное и конечное значение параметра цикла. С учетом того, что начальное значение параметра цикла меньше конечного, К будет принимать значения от 12 до 30 с шагом приращения «+ 1»: К = 12; 13; 14; 15, …, 30.

Графический символ «Решение» отображает решение, имеющее один вход и несколько альтернативных выходов, один и только – один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Условия могут быть представлены в виде ограничений равенств или ограничений неравенств. Символ используют для построения структур ветвления и повторения (цикла).

Графический символ «Линия» отображает поток данных или управления. Соединяет между собой другие графические символы.

Графический символ «Пунктирная линия» отображает альтернативную связь между двумя или более символами.

Графический символ «Соединитель» отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы – соединители должны содержать одно и то же уникальное обозначение (цифра, буква, буквенно-цифровое значение).

Графический символ «Терминатор» отображает выход во внешнюю среду и вход из внешней среды (начало или конец алгоритма, пуск или останов). В случае выхода во внешнюю среду в этот символ вписывается слово «Конец», а в случае входа из внешней среды – «Начало».

Графический символ «Комментарий» используется для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры.

Кроме отдельно взятых графических символов (см. табл. 1.1), для построения алгоритмических структур используются блочные символы в виде базовых управляющих конструкций (рис. 1.1).

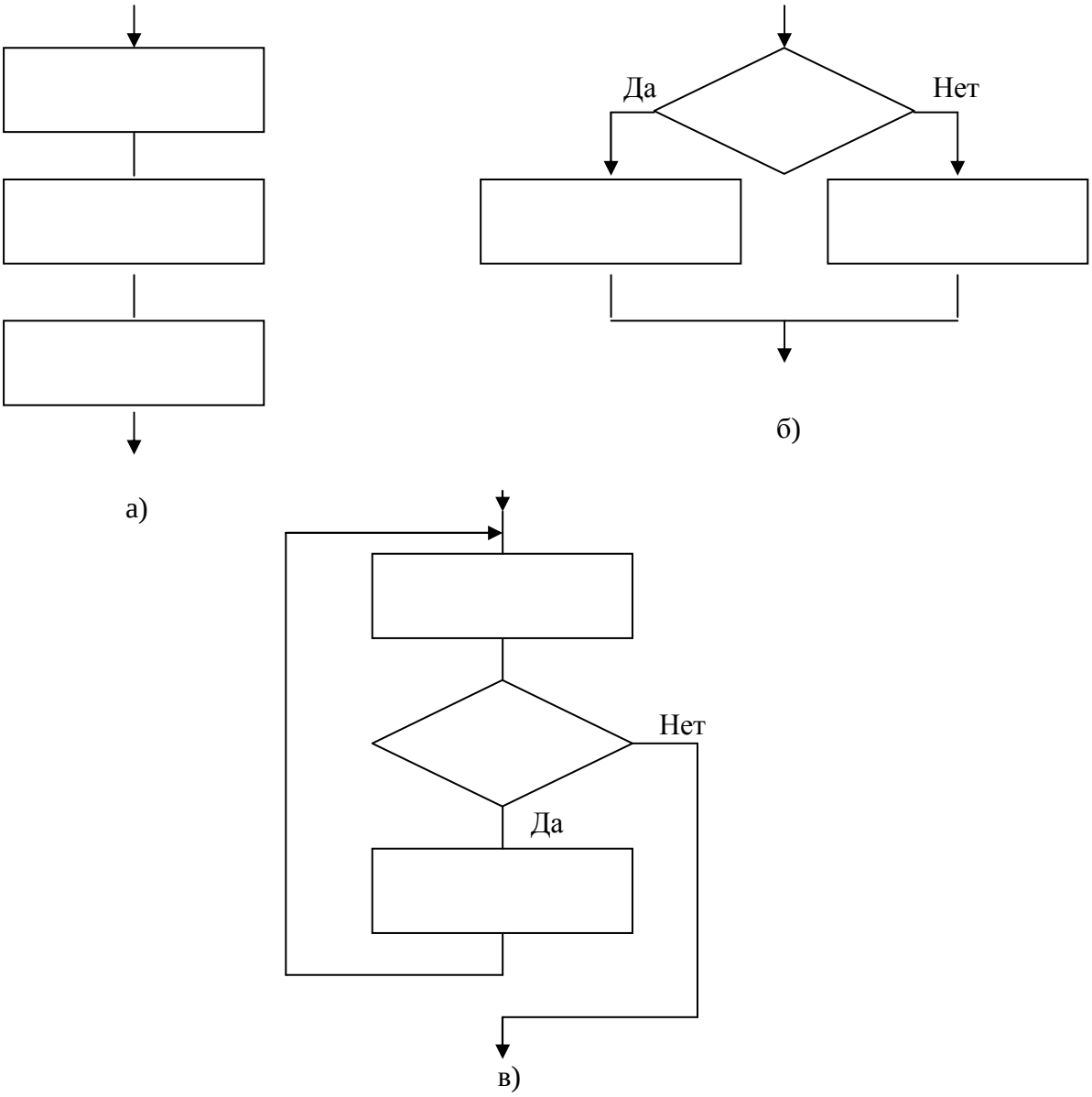


Рис. 1.1. Базовые управляющие конструкции:
а, б, в – соответственно следования, ветвления, повторения

1.3. Разновидности алгоритмических структур для решения задач машиностроения

Типовыми для решения задач, в том числе и задач машиностроения, являются алгоритмические структуры, такие как:

– алгоритм линейной структуры;

– алгоритм разветвленной структуры;

– алгоритм циклической структуры (с вложениями и без вложений);

– алгоритм итерационного цикла.

Алгоритм линейной структуры – это точное предписание, определяющее вычислительный процесс, в котором все действия от ввода варьируемых начальных данных до вывода искомого результата осуществляются последовательно одно за другим.

Пример 1. Составить алгоритм вычисления массы заготовки выполненной в виде пластины, если известны длина, ширина, толщина и плотность материала заготовки. Алгоритм решения задач представлен на рис. 1.2.

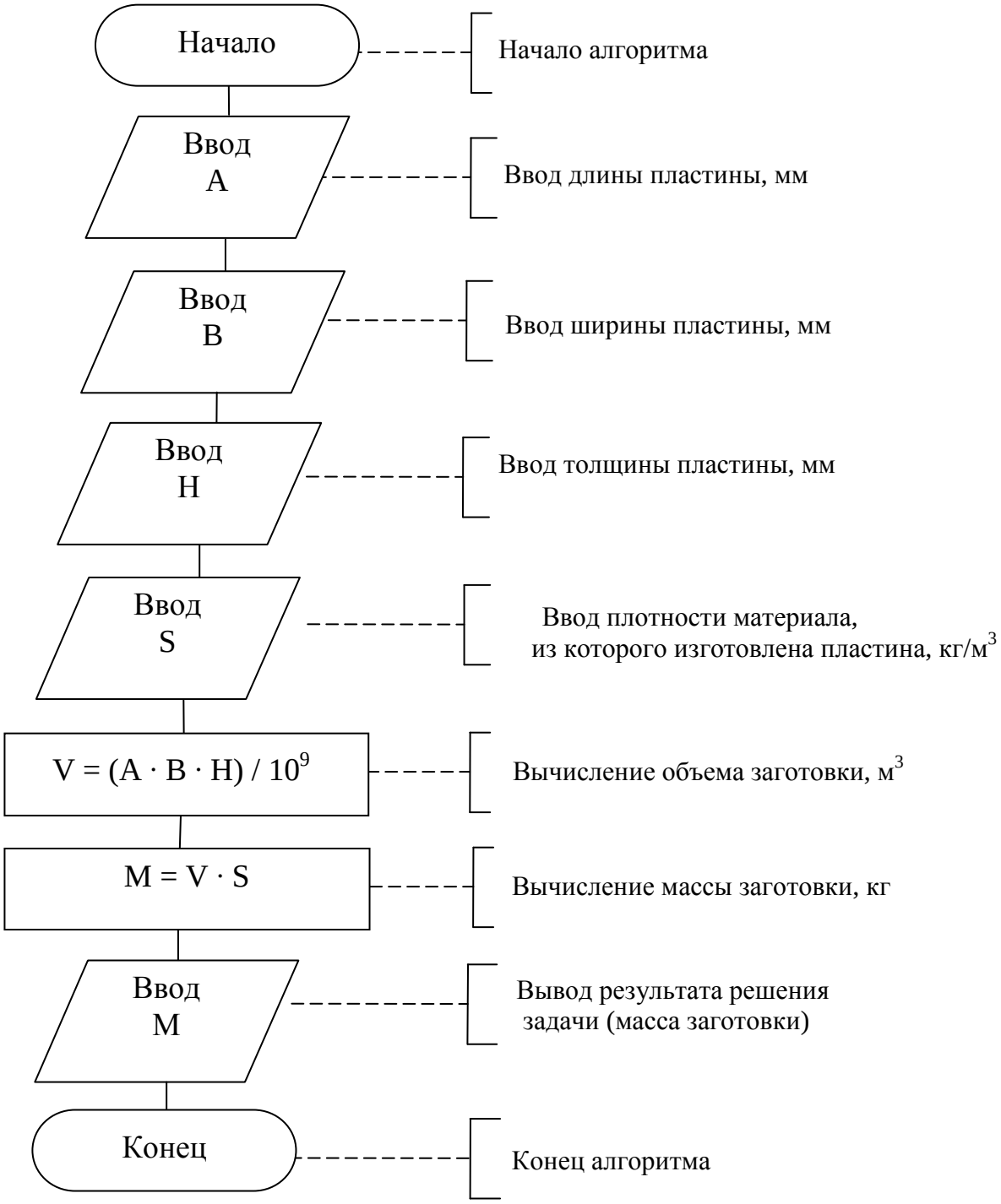


Рис. 1.2. Блок-схема алгоритма линейной структуры (постановка задачи см. пример 1)

Алгоритм разветвленной структуры – это точное предписание, определяющее вычислительный процесс, в котором в зависимости от выполнения или не выполнения ограничивающего условия или условий последовательность действий может разветвляться на два или более направлений.

Пример 2. Составить алгоритм вычисления коэффициента Z_V, учитывающего влияние окружной скорости V шестерни на ее износ в зубчатой передаче, если твердость H_0 зубьев колес может быть меньше, равна или больше 350 единиц по шкале НВ: $Z_V = 0{,}85 \cdot V^{0,1}$ при $H_0 \leq 350$НВ; $Z_V = 0{,}925 \cdot V^{0,05}$ при $H_0 > 350$НВ, где V – окружная скорость, м/мин; H_0 – твердость поверхности зубьев зубчатых колес.

Алгоритм решения задачи представлен на рис. 1.3.

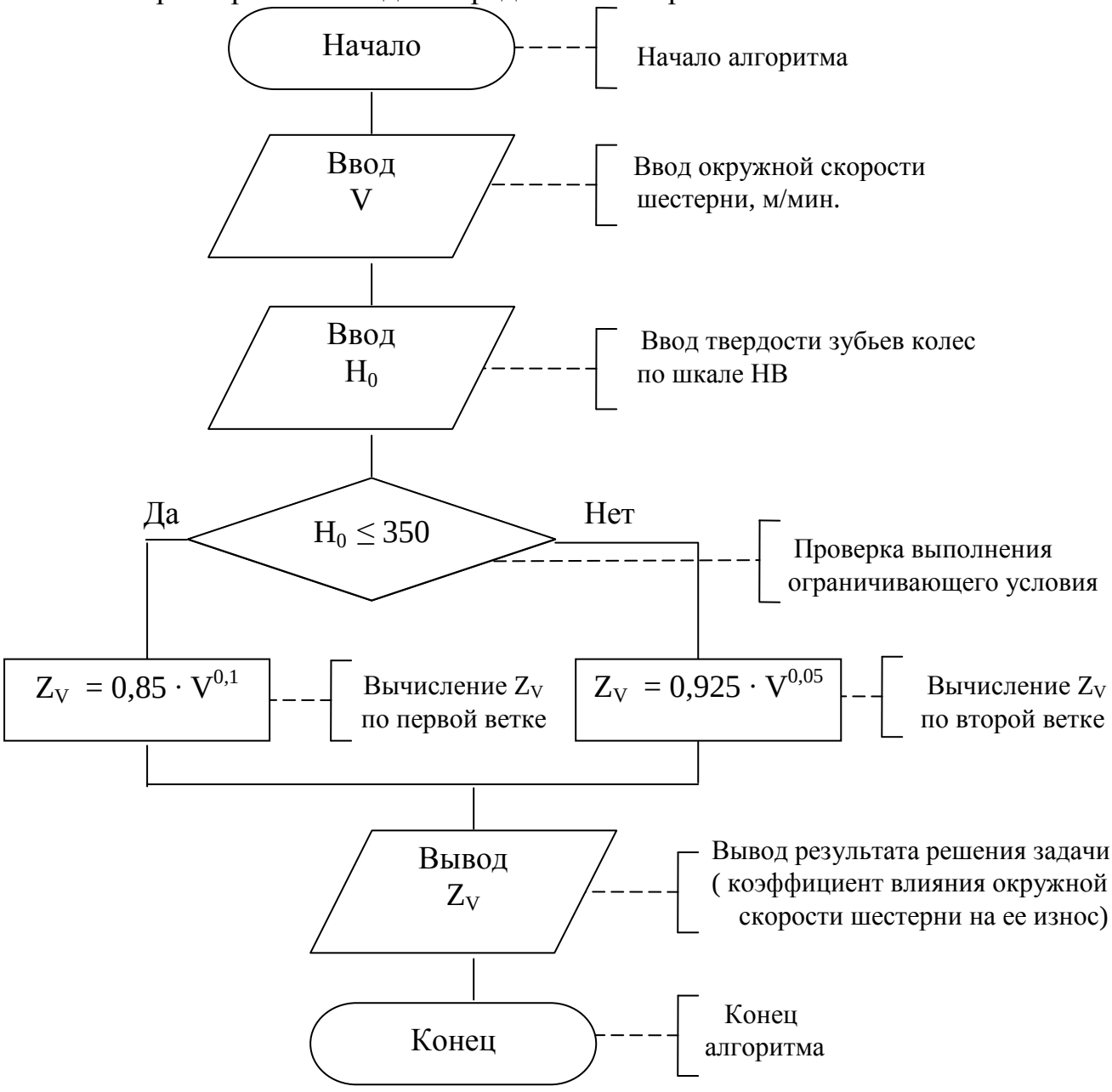


Рис. 1.3. Блок-схема алгоритма разветвленной структуры на два направления (постановка задачи см. пример 2)

Пример 3. Составить алгоритм, с помощью которого по величине среднего арифметического отклонения профиля R_a (мкм) микронеровностей поверхности рассчитывалась высота микронеровностей поверхности по 10-ти точкам R_z (мкм) детали: $R_z = 4 \cdot R_a$, при $R_a = 1{,}25 \ldots 2{,}5$; $R_z = 5 \cdot R_a$, при $R_a = 0{,}32 \ldots 0{,}63$; $R_z < 0{,}1$, при $R_a < 0{,}04$.

Алгоритм решения задачи представлен на рис. 1.4.

Алгоритм циклической структуры это точное предписание, определяющее вычислительный процесс, в котором одна и та же последовательность действий выполняется многократно с заранее известным числом повторений.

Пример 4. Составить алгоритм по определению интенсивности изнашивания фрезы U, если ширина фрезерования B изменяется от начального значения $B_0 = 40$ мм до конечного значения $B_n = 60$ мм с шагом $h_B = 5$ мм: $U = (1 + 100/B) \times \times U_0$, где $U_0 = 16$ мкм/км – начальный износ фрезы.

Алгоритм решения задачи представлен на рис. 1.5.

Результатом решения поставленной задачи в соответствии с алгоритмом (см. рис. 1.5) является формирование массива табличных значений выходного параметра – интенсивность изнашивания фрезы U в зависимости от одного входного параметра (параметра цикла) – ширина фрезерования B. Тем самым выявляется закономерность влияния ширины фрезерования B на интенсивность износа фрезы U. По получаемым табличным значениям U и B может быть построен график этой зависимости. Количество выводимых табличных значений U и B при известных исходных данных примера 4, а также и количество повторений однотипной последовательности действий (многократное обращение к одной и той же расчетной зависимости U= f(B)) можно рассчитать по формуле:

$$N = \frac{(B_n - B_0)}{h_B} + 1. \qquad (1)$$

При $B_0 = 40$ мм, $B_n = 60$ мм, $h_B = 5$ мм:

$$N = \frac{60 - 40}{5} + 1 = 5.$$

Пример 5. Составить алгоритм по определению скорости резания V при сверлении отверстия сверлом диаметром D, изменяющимся от начального значения $D_0 = 1$ мм до конечного значения $D_n = 4$ мм с шагом $h_D = 1$ мм, и подачей S, изменяющейся от начального значения $S_0 = 0{,}1$ мм/об до конечного значения $S_n = 0{,}4$ мм/об с шагом $h_s = 0{,}1$ мм/об: $V = C_v \cdot D^q \cdot K_v / T^m \cdot S^y$, где $C_v = 400$; $q = m = y = 0{,}5$; $K_v = 1$; $T = 60$ мин – период стойкости сверла.

Алгоритм решения задачи представлен на рис. 1.6.

Результатом решения поставленной задачи в соответствии с алгоритмом (см. рис. 1.6) является формирование массива табличных значений выходного параметра – скорость резания при сверлении V в зависимости от двух входных параметров (параметров внешнего и вложенного цикла) – диаметра сверла D и подачи S.

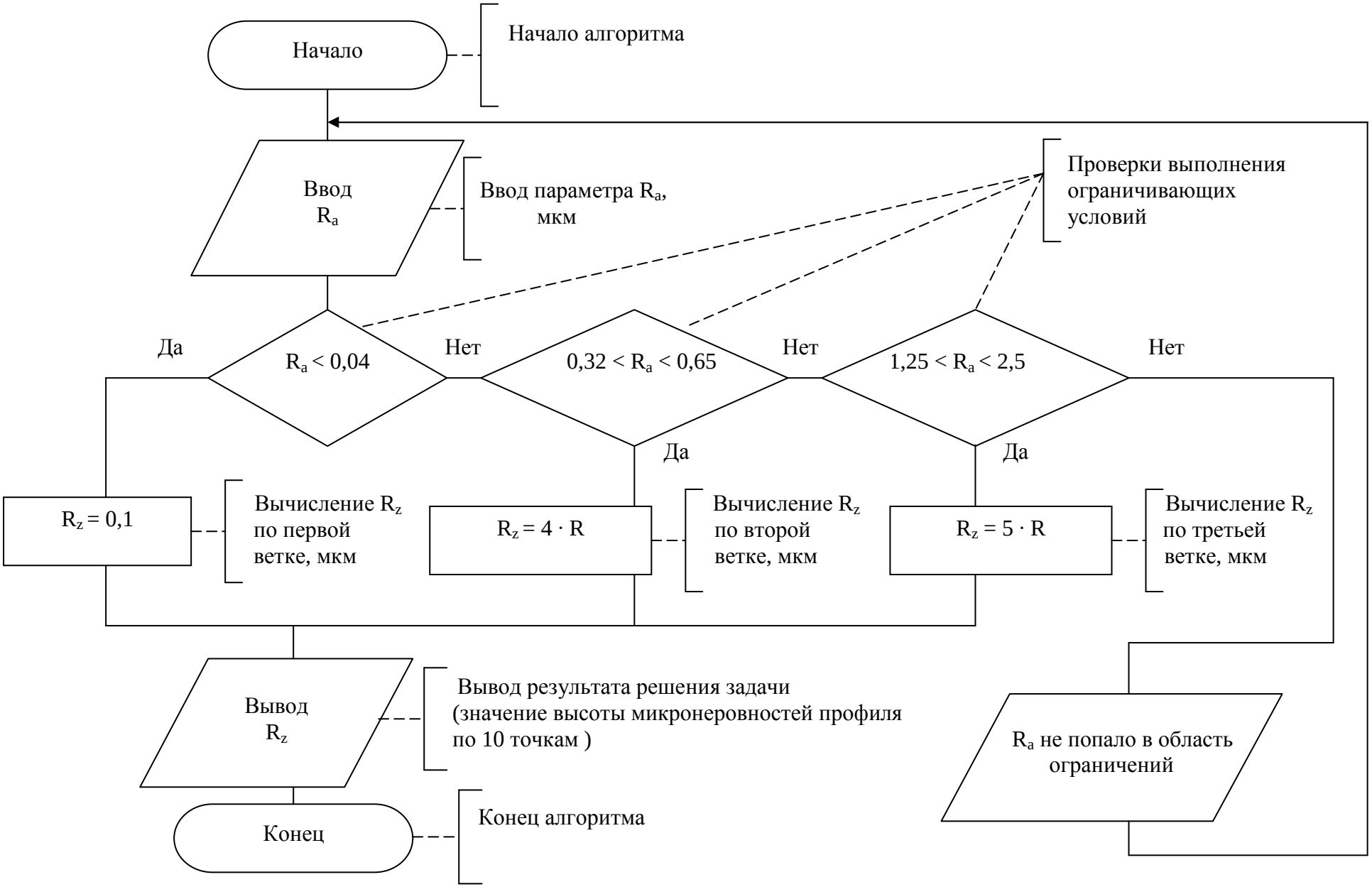


Рис. 1.4. Блок-схема алгоритма разветвленной структуры на три направления (постановка задачи см. пример 3)

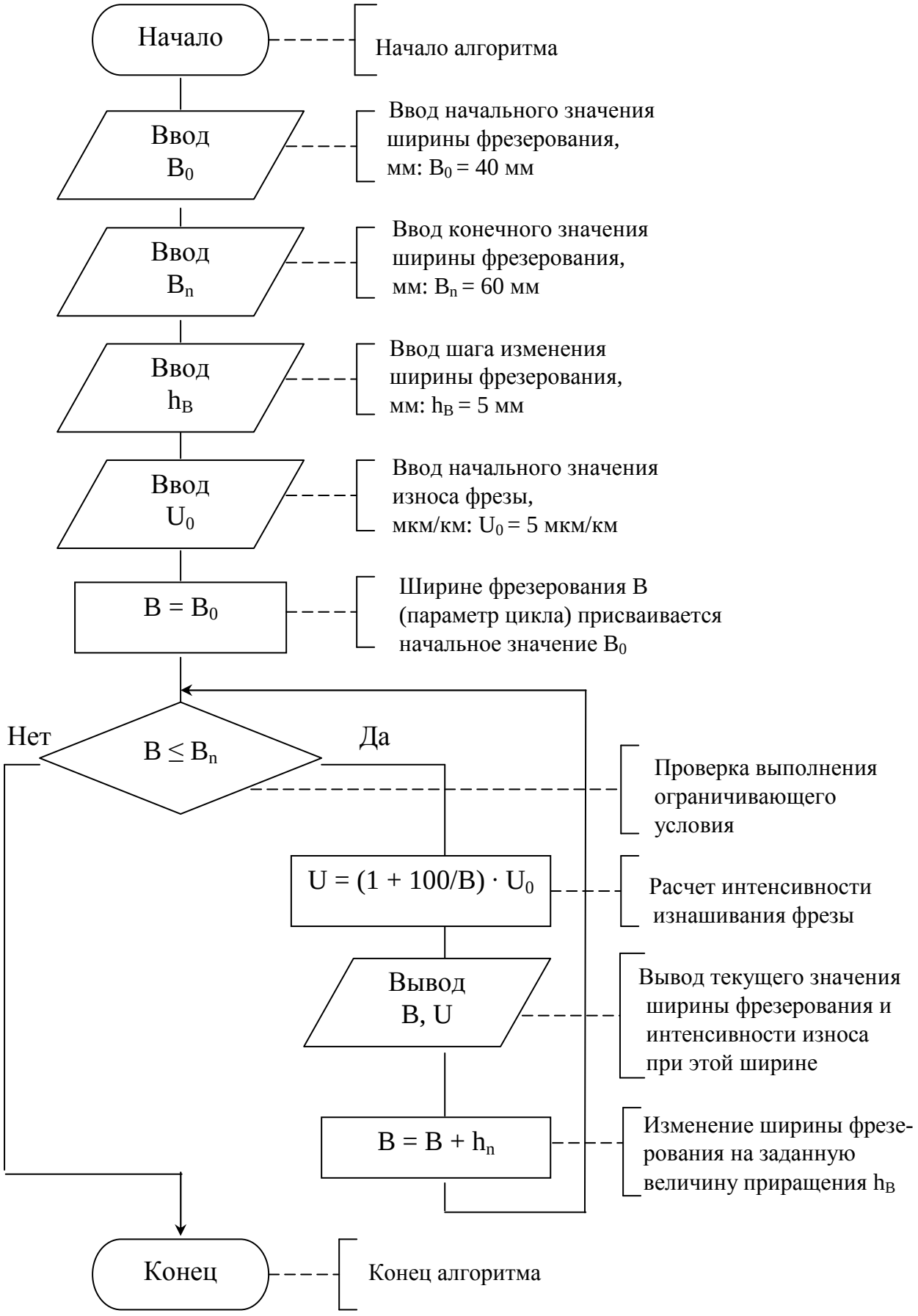


Рис. 1.5. Блок-схема алгоритма циклической структуры без вложений (постановка задачи см. пример 4)

Значение скорости резания V определяется при всех возможных из указанного диапазона значениях D и S и при всех возможных сочетаниях этих значений. Количество выводимых табличных значений V, D и S при известных исходных данных примера 5, а следовательно, и количество повторений однотипной последовательности действий (многократное обращение к одной и той же расчётной зависимости) V = f(D, S)) можно рассчитать по формуле:

$$N = \left(\frac{D_n - D_0}{h_D} + 1 \right) \cdot \left(\frac{S_n - S_0}{h_S} + 1 \right). \quad (2)$$

При D_0 = 1 мм, D_n = 4 мм, h_D = 1 мм, S_0 = 0,1 мм/об, S_n = 0,4 мм/об, h_S = 0,1 мм/об:

$$N = \left(\frac{4-1}{1} + 1 \right) \cdot \left(\frac{0,4-0,1}{0,1} + 1 \right) = 16.$$

Алгоритм итерационного цикла – это точное предписание, определяющее вычислительный процесс, в котором одна и та же последовательность действий выполняется многократно с заранее неизвестным числом повторений.

Пример 6. Составить алгоритм нахождения такого значения скорости резания V при сверлении отверстия, при котором шероховатость обработанной поверхности по параметру R_Z – высота микронеровностей профиля по десяти точкам будет меньше 20 мкм: $R_Z = 48,7 \cdot \frac{D^{0,17} \cdot D^{0,46}}{V^{0.04}}$, где D – диаметр сверла, мм: D = 12 мм; S – подача, мм/об: S = 0,1 мм/об; V – скорость резания, м/мин. Начальное значение скорости резания V_0 = 12 м/мин, шаг изменения скорости резания h_V = 0,1 м/мин.

Алгоритм решения задачи представлен на рис. 1.7.

Результатом решения поставленной задачи в соответствии с алгоритмом (см. рис. 1.7) является нахождение скорости резания V, при которой шероховатость поверхности R_Z будет меньше заданного значения R_{Zmin}. Для этого V изменяют от начального значения V_0 (при V_0 условие $R_Z < R_{Zmin}$ не выполняется) с минимальным шагом приращения h_v. При каждом новом значении скорости резания рассчитывают R_Z и проверяют выполнение условия $R_Z < R_{Zmin}$. Как только это условие выполнится осуществляется выход из тела итерационного цикла на вывод результата, того последнего значения V, при котором и обеспечилось выполнение условия $R_Z < R_{Zmin}$.

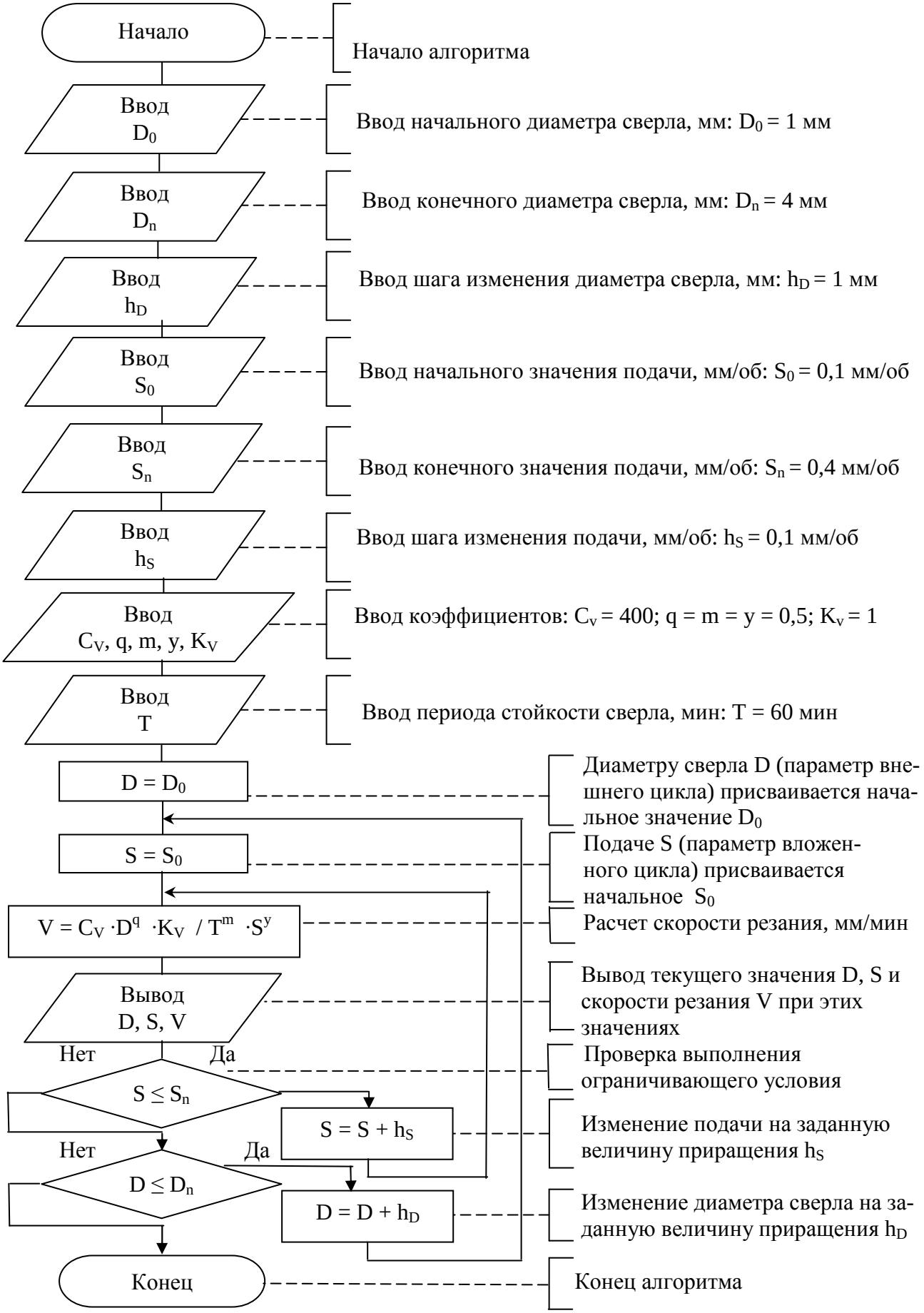


Рис. 1.6. Блок-схема алгоритма циклической структуры с вложениями (постановка задачи см. пример 5)

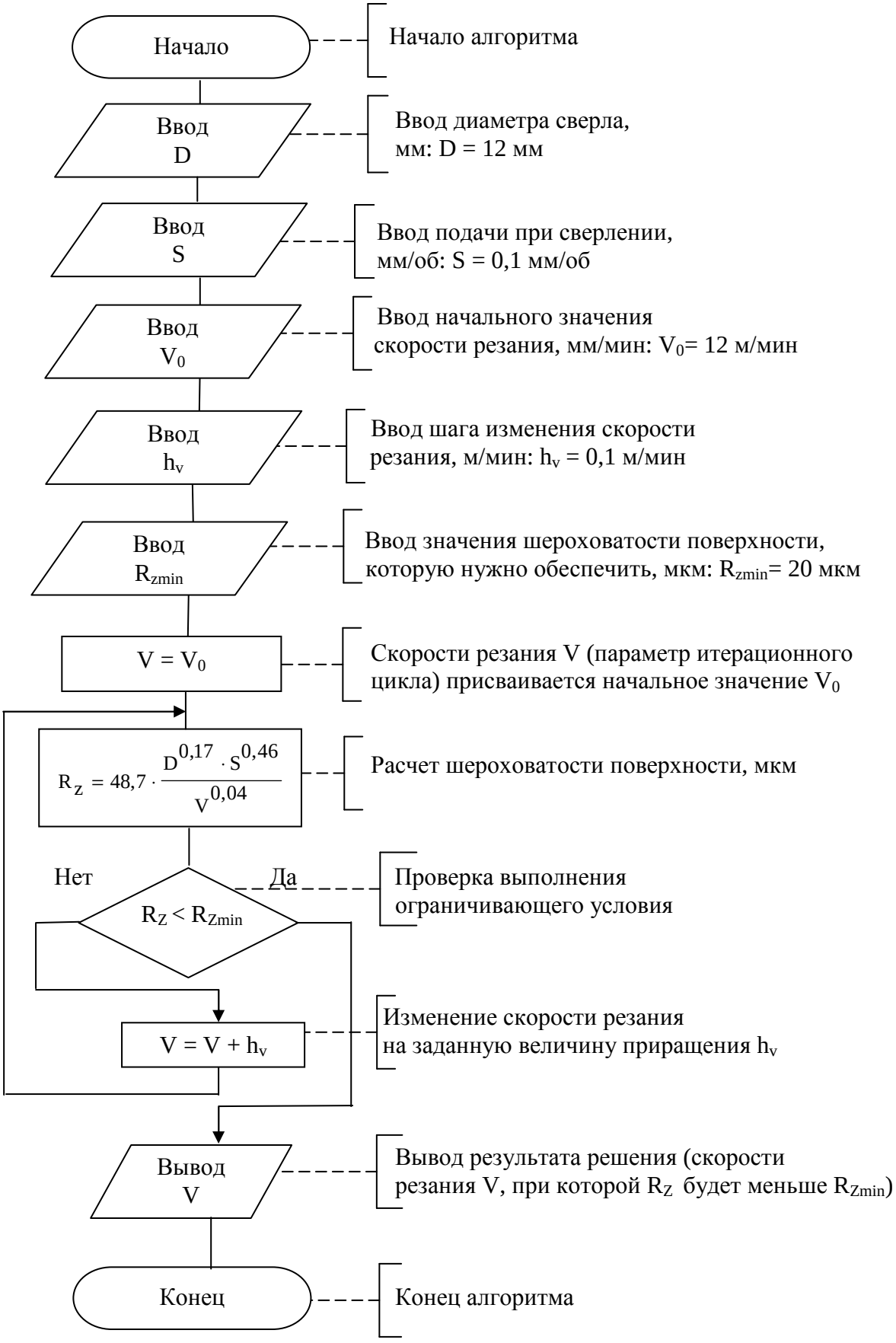


Рис. 1.7. Блок-схема алгоритма итерационного цикла
(постановка задачи см. пример 6)

1.4. Понятие программы. Связь программы с алгоритмической структурой

В общем понимании программой называется набор команд, манипулирующих данными. Для реализации любой программы служат языки программирования, как одна из разновидностей программного обеспечения, с которым может работать пользователь электронно-вычислительной машины (ЭВМ).

Рассмотренные ранее способы описания алгоритмических структур обладают существенным недостатком. Записи предписаний в этих алгоритмических структурах, представленные в виде связанных друг с другом графических символов, не могут непосредственно быть воспринятыми ЭВМ. Они служат для предварительной работы с алгоритмом в расчете на то, что существуют средства описания алгоритмов. Таким средством описания алгоритмов и являются языки программирования. Они позволяют на основе строгих правил формировать последовательность предписаний, однозначно отражающих смысл и содержание частей алгоритма, с целью их последовательного выполнения с помощью ЭВМ. Выбор языка программирования, как средства описания алгоритмической структуры, зависит от характера решаемых задач, возможностей ЭВМ, наклонностей самого пользователя ЭВМ.

В основе всех языков программирования лежит понятие – оператор. Оператор – это самостоятельный этап алгоритмической структуры, представляющий самостоятельную единицу языка. Выделяется фиксированное количество операторов, каждый из которых указывает на выполнение определенных действий. Операторы бывают следующих типов: присваивания, перехода, процедур, функций, пустые операторы, условные операторы, операторы повторения (циклов), ввода данных, вывода данных, выбора, ветвления. Операторы обычно выполняются в той последовательности, в которой они вписаны в программе, за исключением некоторых случаев, когда эта последовательность может прерываться операторами перехода или сокращаться условными операторами. В построении операторов участвуют более мелкие единицы – выражения. Выражения по определенным правилам соединяются между собой ограничителями (выделенными словами). Выражения в свою очередь строятся с помощью знаков, операций, скобок, чисел, переменных, указателей, функций, логических значений и т. д.

Установим связь программы написанной на языке Turbo Pascal 7.0 с алгоритмической структурой при решении задачи в соответствии с примером 7.

Пример 7. Составить алгоритм и программу для определения усилия P, изгибающего оправку с торцовой фрезой под действием составляющих сил резания P_z и P_y:

$$P = \sqrt{P_y^2 + P_z^2},$$

где P, P_z, P_y в Н.

Структурная схема связи алгоритма и программы представлена на рис. 1.8.

На структурной схеме (см. рис. 1.8): по линии связи 1 происходит замена графического символа «Терминатор» зарезервированным языком программирования словом «Begin»; по линии связи 2 графический символ «Данные» заменяется зарезервированными структурой оператора вывода «Write» и ввода «Readln» силы резания P_y; по линии связи 3 графический символ «Данные» заменяется зарезервированными структурой оператора вывода «Write» и ввода «Readln» силы резания P_z; по линии связи 4 графический символ «Процесс» заменяется зарезервированным оператором присваивания (расчет выходного параметра P); по линии связи 5 графический символ «Данные» заменяется зарезервированной структурой оператора вывод «Writeln» результата решения задачи; по линии связи 6 происходит замена графического символа «Терминатор» зарезервированным словом «End». Таким образом, каждый из выше обозначенных операторов и зарезервированных слов языка Turbo Pascal 7.0 представляет самостоятельный этап алгоритмической структуры. Последовательность обращения к этим операторам в программе установлена в строгом соответствии с последовательностью следования графических символов в алгоритмической структуре, относящейся к линейной и предписывающей поэтапное выполнение действий, следующих одно за другим.

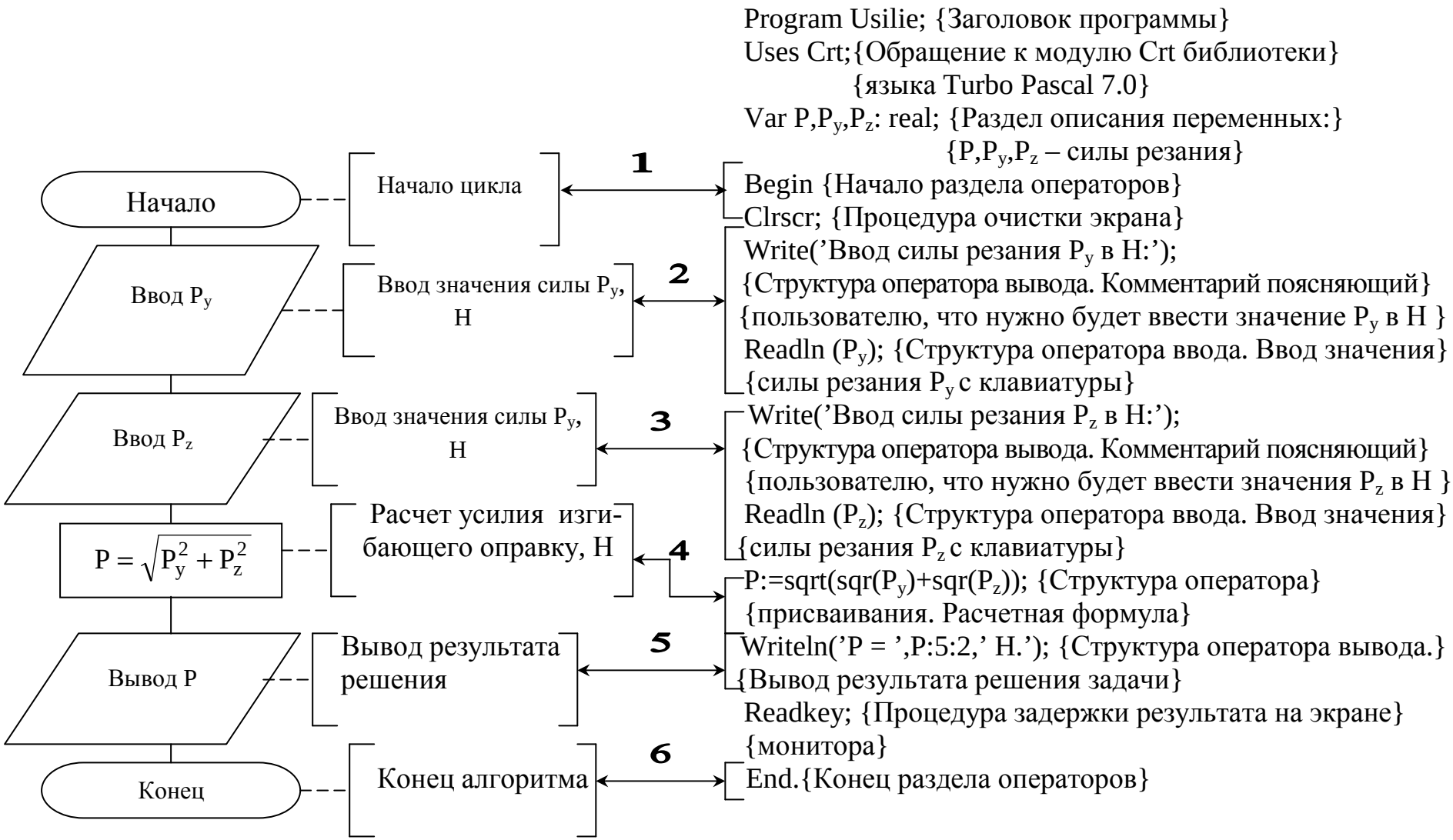


Рис. 1.8. Структурная схема связи алгоритма, представленного в виде блок-схемы, и программы (постановка задачи см. пример 7)

2. МЕТОДЫ НАХОЖДЕНИЯ ПАРАМЕТРОВ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ С НЕЛИНЕЙНОЙ ХАРАКТЕРИСТИКОЙ

2.1. Классификация и общий подход к реализации методов нахождения параметров технологических систем с нелинейной характеристикой

В большинстве случаев на практике выходные параметры технологических систем изменяются по законам, которые представлены в виде уравнений с нелинейной характеристикой (нелинейные уравнения). Как правило, при решении таких уравнений необходимо находить корни. Для нахождения корней используют прямые и итерационные методы.

Реализация прямых методов основывается на нахождении корней уравнений с помощью конечных формул. Известны методы решения простых алгебраических, тригонометрических, логарифмических и показательных уравнений.

Реализация итерационных методов (методов пошагового приближения к искомым значениям корней или корня нелинейного уравнения) позволяет решить практически любые уравнения независимо от их сложности. Итерационные методы легко программируются и реализуются с помощью ЭВМ. К ним можно отнести:

- метод деления отрезка пополам;
- метод хорд;
- метод касательных;
- метод простых итераций.

На первом этапе осуществляется поиск либо приближенного значения корня (метод касательных и простых итераций), либо отрезка, содержащего этот корень (метод деления отрезка пополам и метод хорд). Поиск приближенного значения корня уравнения с нелинейной характеристикой можно осуществить графически, путем построения графика рассматриваемой функции $y = f(x)$ и нахождения такого значения входного параметра x, при котором выполняется условие – $y = f(x) \approx 0$, где y – выходной параметр технологической системы. Установить отрезок [a, b], содержащий корень уравнения – это значит доказать, что на концах этого отрезка функция $y = f(x)$ меняет свой знак. Для проверки выполнения этого условия необходимо, чтобы произведение значений функции на концах отрезка [a, b] было числом отрицательным, а именно $f(a) \cdot f(b) < 0$.

На втором этапе шаг за шагом осуществляется уточнение значения корня с заданной степенью точности E ($E \to 0$: E=0,1; 0,01; 0,001;…), используя какой-либо метод из вышеперечисленных.

Наиболее распространенными из итерационных методов являются метод деления отрезка пополам и метод простых итераций.

2.2. Метод деления отрезка пополам в нахождении корней уравнений с нелинейной характеристикой

Пусть нам известно уравнение с нелинейной характеристикой y = f(x), описывающее закон изменения выходного параметра технологической системы y от входного параметра x. Необходимо найти такое значение x, при котором выполняется условие y = f(x) = 0.

В соответствии с первым этапом реализации итерационных методов (см. п. 2.1) установим отрезок [a, b], содержащий решение (корень) нелинейного уравнения. Для этого подберем такие значения левой a и правой b границу отрезка [a, b] изменения входного параметра x, при которых выполняется условие – f(a) × × f(b) < 0, т. е. на концах отрезка [a, b] функция y = f(x) будет менять свой знак. Проверку правильности выбора границ отрезка [a, b] можно также подтвердить путем построения графика рассматриваемой функции y = f(x) (рис. 2.1).

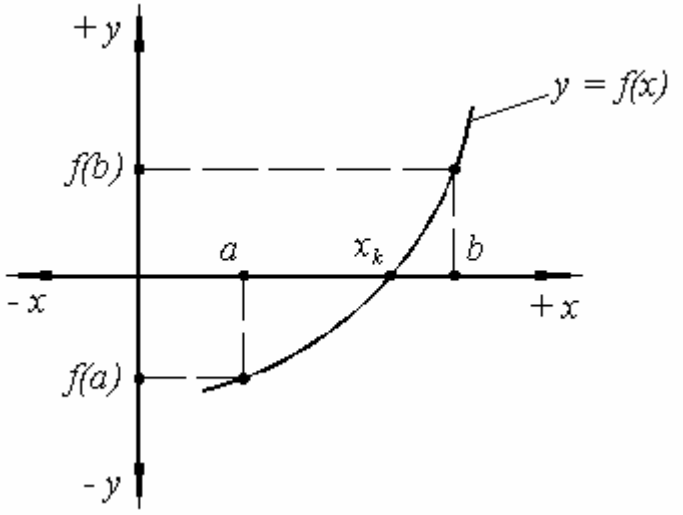


Рис. 2.1. Схема к установлению границ отрезка, содержащего корень нелинейного уравнения: a – левая граница отрезка со значением функции в этой точке f(a) < 0 (число отрицательное); b – правая граница отрезка со значением функции в этой точке f(b) > 0 (число положительное); x_k – искомый корень уравнения y = f(x), который будет находиться на отрезке [a, b]

Определившись с отрезком, содержащим корень уравнения с нелинейной характеристикой, перейдем к реализации второго этапа – уточнения значения корня с заданной степенью точности E. Для этого реализуется итерационный процесс по методу деления отрезка пополам. В рамках первой итерации (шага) найдем начальное приближение x_0 к корню x_k нелинейного уравнения, поделив отрезок [a, b] пополам – $x_0 = (a + b) / 2$. Рассчитаем значение функции в точке начального приближения $f(x_0)$ и проверим выполнение условия $|f(x_0)| \le E$. Если это условие выполняется, то искомое значение корня x_k будет соответствовать начальному приближению x_0 ($x_k = x_0$), в противном случае нужно продолжить процесс уточнения корня нелинейного уравнения, реализовав следующую

итерацию (шаг). Предположим, что условие $\left| f(x_0) \right| \le E$ не выполнилось. Реализуем вторую итерацию. В рамках второй итерации сначала исследуем значения функции y = f(x) на концах отрезков [a, x_0] и [x_0, b], которые появились в результате деления отрезка [a, b] пополам точкой начального приближения x_0 с первой итерации. Отрезок, на концах которого функция y = f(x) меняет свой знак, принимается содержащим корень нелинейного уравнения. Предположим, что условие $f(a) \cdot f(x_0) < 0$ не выполняется, а условие $f(x_0) \cdot f(b) < 0$ выполняется, следовательно дальнейшее уточнение значения корня нужно вести на отрезке [x_0, b]. Для этого отрезок [x_0, b], содержащий корень нелинейного уравнения, делим пополам – $x_1 = (x_0 + b) / 2$. Тем самым, найдена точка нового приближения со второй итерацией. Рассчитаем значение функции в этой точке $f(x_1)$ и проверим выполнение условия $\left| f(x_1) \right| \le E$. Предположим, что условие выполнилось, следовательно, искомое значение корня x_k будет соответствовать приближению x_1 со второй итерации ($x_k = x_1$). Если бы условие $\left| f(x_1) \right| \le E$ не выполнилось, то необходимо было бы реализовать следующую (третью) итерацию по уточнению значения корня уравнения с нелинейной характеристикой x_k аналогичную по своей структуре первой и второй.

Таким образом, итерационный процесс по методу деления отрезка пополам будет продолжаться до тех пор, пока не выполнится условие $\left| f(x_n) \right| \le E$, где x_n – уточнение корня нелинейного уравнения с n-ой итерации. Количество итераций зависит от степени точности E и длины исходного отрезка [a, b], который содержит корень x_k нелинейного уравнения. Чем выше степень точности E и больше длина отрезка [a, b], тем больше число шагов для уточнения искомого корня нужно выполнить, что предопределяет необходимость использования ЭВМ для автоматизации процесса и сокращения затрат времени на решение поставленной задачи. В соответствии с вышеизложенным, алгоритм метода деления отрезка пополам может быть представлен в виде блок-схемы на рис. 2.2.

Пример 8. Определить величину внешнего усилия N, воздействующего на крепежный болт, изготовленный из стали 45, которое может выдержать этот болт, если его диаметр d_1 = 3 мм:

$$d_1 = \sqrt{\frac{4 \cdot N}{\pi \cdot \left[\sigma_p \right]}},$$

где N – внешнее усилие (усилие растяжения болта), кгс; $[\sigma_p]$ – допустимое напряжение при растяжении, кгс/мм^2: $[\sigma_p]$ = 7,2 кгс/мм^2 – для стали 45; d_1 – внутренний диаметр резьбы болта, мм. Степень точности E = 0,01.

РЕШЕНИЕ. При известных $[\sigma_p]$ = 7,2 кгс/мм^2 и заданном в качестве ограничения d_1 = 3 мм, искомое значение N должно удовлетворять условию:

$$\sqrt{0{,}18 \cdot N} = 3$$

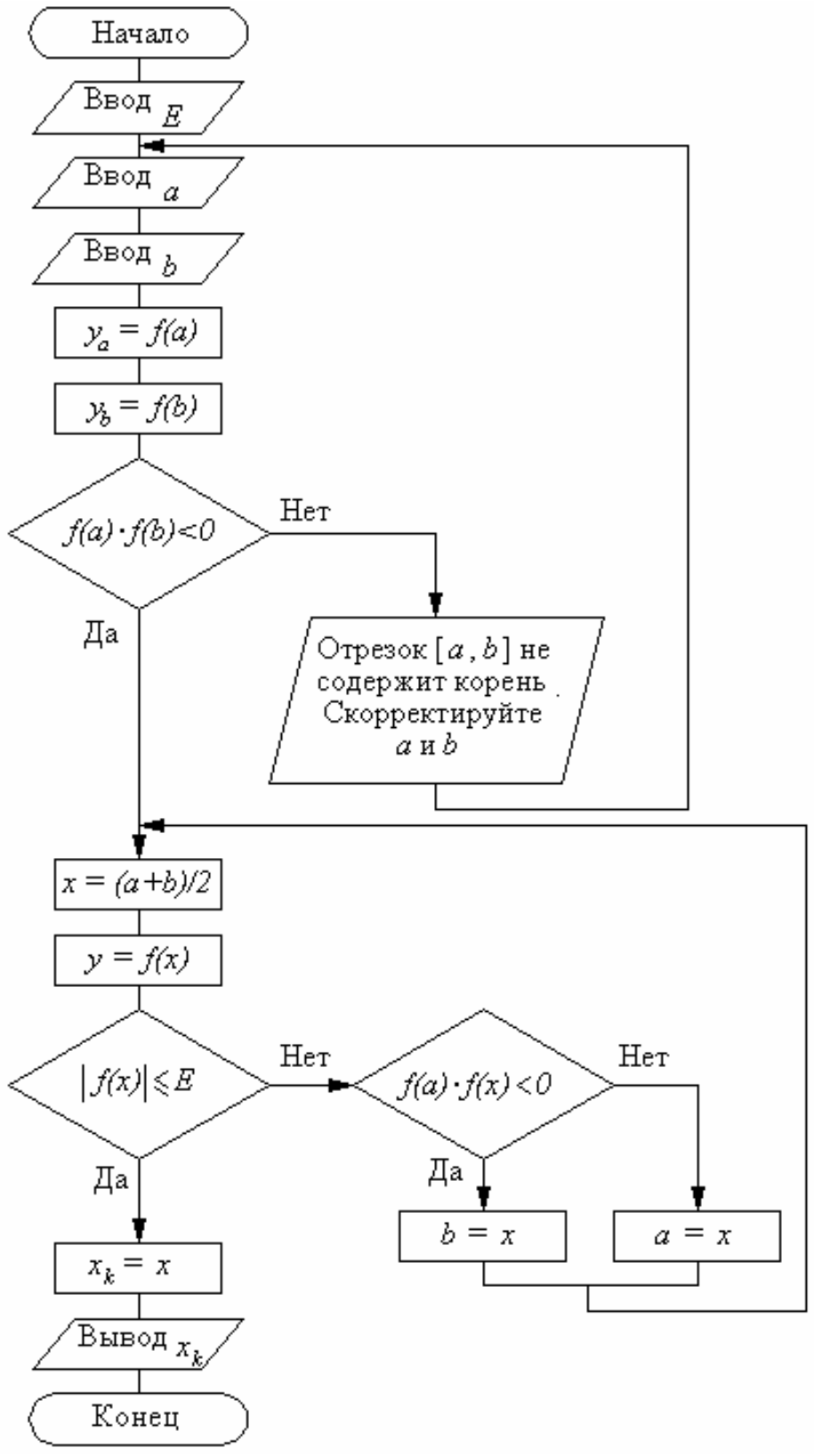


Рис. 2.2. Блок-схема алгоритма метода деления отрезка пополам

или

$$\sqrt{0{,}18 \cdot N} - 3 = 0.$$

Определим границы отрезка, содержащего корень этого нелинейного уравнения. Для этого табулируем исходную функцию и построим ее график (рис. 2.3).

Опираясь на рис. 2.3, в качестве отрезка, содержащего корень (решение) уравнения, можно выбрать отрезок $[N_Л, N_П] = [45, 65]$, где $N_Л$ – левая граница отрезка, кгс; $N_П$ – правая граница, кгс. На границах этого отрезка функция $y = \sqrt{0{,}18 \cdot N} - 3$ меняет свой знак относительно границы 2: $y_Л = f(N_Л) = \sqrt{0{,}18 \cdot 45} - 3 = -0{,}154$; $y_П = f(N_П) = \sqrt{0{,}18 \cdot 65} - 3 = 0{,}421$. Следовательно, условие $f(N_Л) \cdot f(N_П) < 0$ выполняется:

$$-0{,}154 \cdot 0{,}421 < 0;$$

$$-0{,}064 < 0.$$

Реализуем первую итерацию. Поделим отрезок $[N_Л, N_П]$ пополам:

$$N_0 = \frac{N_Л + N_П}{2};$$

$$N_0 = \frac{45 + 65}{2} = 55 \text{ кгс}.$$

Рассчитаем значение функции при усилии растяжения $N_0 = 55$ кгс:

$$y = f(N_0) = \sqrt{0{,}18 \cdot 55} - 3 = 0{,}15.$$

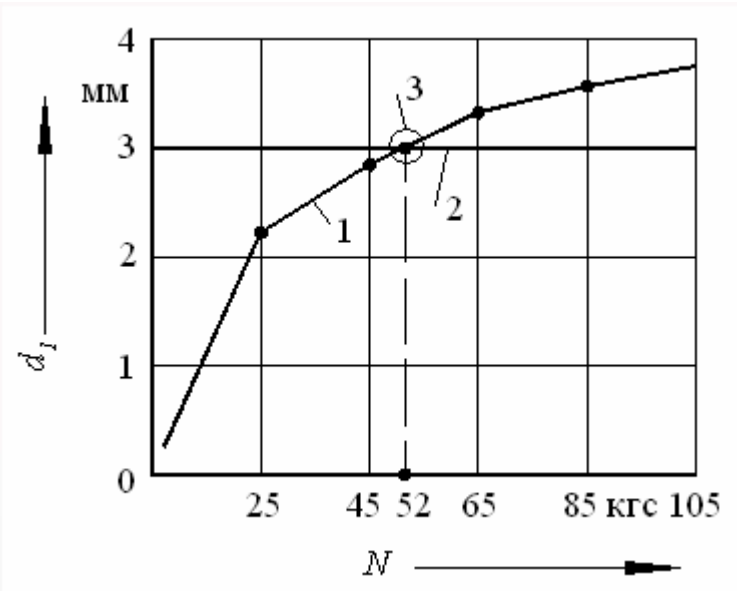


Рис. 2.3. Зависимость внутреннего диаметра резьбы болта d_1 от усилия растяжения болта N:
1 – кривая $d_1 = f(N)$; 2 – граница внутреннего диаметра резьбы болта ($d_1 = 3$ мм);
3 – точка пересечения кривых 1 и 2 (примерное решение задачи)

Проверим выполнение условия $|f(N_0)| \le E$. Условие не выполняется ($|0{,}15| \not\le 0{,}1$). Следовательно, N_0 не может быть принято за искомое решение.

Реализуем вторую итерацию. Предварительно проверим, на каком из отрезков $[N_Л, N_0]$ или $[N_0, N_П]$ функция $y = \sqrt{0{,}18 \cdot N} - 3$ меняет свой знак:

$$f(N_Л) = -0{,}154;$$

$$f(N_0) = 0{,}15;$$

$$f(N_П) = 0{,}421, \text{ следовательно,}$$

$$f(N_Л) \cdot f(N_0) < 0;\ -0{,}023 < 0;$$

$$f(N_0) \cdot f(N_П) \not< 0;\ -0{,}063 \not< 0.$$

Новым отрезком, содержащим корень, является отрезок $[N_Л, N_0]=[45, 55]$. Разделим отрезок $[N_Л, N_0]$ пополам:

$$N_1 = \frac{45+55}{2},$$

$$N_1 = 50 \text{ кгс}.$$

Рассчитаем значение функции при усилии растяжения N_1=50 кгс:

$$y = f(N_1) = \sqrt{0{,}18 \cdot 50} - 3 = 0.$$

Проверим выполнение условия $|f(N_1)| \le E$. Условие выполняется ($|0| \le 0{,}01$). Следовательно, искомое значение корня N_K будет соответствовать приближению N_1 со второй итерации ($N_K = N_1 = 50$ кгс). Таким образом, усилие растяжения, которое может выдержать болт с внутренним диаметром резьбы $d_1 = 3$ мм, изготовленный из стали 45, не должно превышать 50 кгс или 500 Н.

В соответствии с поставленной задачей (см. пример 8) разработаем алгоритм (рис. 2.4) и программу USILIE ее решения на языке TURBO PASCAL.

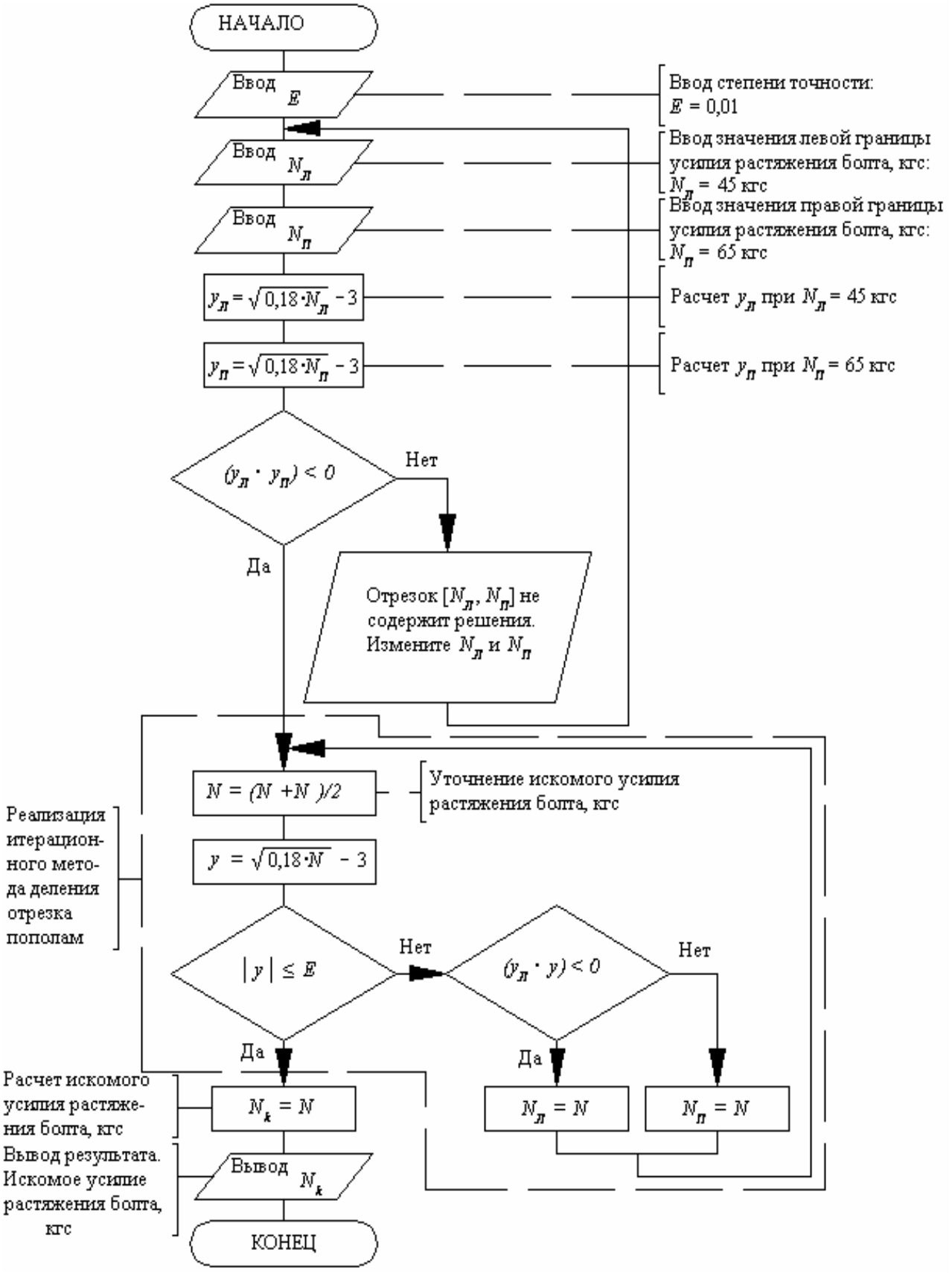


Рис. 2.4. Блок-схема алгоритма нахождения усилия растяжения болта методом деления отрезка пополам (постановка задачи см. пример 8)

ПРОГРАММА USILIE

(постановка задачи см. пример 8, алгоритм см. рис. 2.4)

```
    Program Usilie;
    Uses Crt;
    Label 10,20,30;
    Var E,Nl,Np,yl,yp,N,y,Nk:Real;
    Begin
    Clrscr;
    Write('Задайте степень точности E:');
    Redln(E);
10: Write('Введите значение левой границы усилия');
    Writeln('растяжения болта Nл в кгс:');
    Readln(Nl);
    Write('Введите значение правой границы усилия');
    Writeln('растяжения болта Nп в кгс:');
    Readln(Np);
    Begin
    yl:=Sqrt(0.18*Nl)-3;
    yp:=Sqrt(0.18*Np)-3;
    If (yl*yp)<0 then Goto 20;end Else
    Write('Отрезок [Nл,Nп] не содержит');
    Write(' решения. Скорректируйте значения');
    Writeln('Nл и Nп.');Goto 10;
20: N:=(Nl+Np)/2;
    y:=Sqrt(0.18*N)-3;
    If Abs(y)<=E then Goto 30;
    If (yl*y)<0 then Nl:=N Else
    Np:=N; Goto 20;
30: Nk:=N;
    Write('Искомое усилие растяжения болта');
    Writeln(' Nк = ',Nk:5:2,' кгс.');
    Readkey;
    End.
```

2.3. Метод простых итераций в нахождении корней уравнения с нелинейной характеристикой

Пусть нам известно уравнение с нелинейной характеристикой y = f(x) для нахождения параметров технологической системы. Требуется найти такое значение входного параметра *x*, при котором выполняется условие y = f(x) = 0.

В соответствии с первым этапом реализации итерационных методов (см. п. 2.1) установим точку начального приближения x_0 к искомому решению (корню) x_K уравнения с нелинейной характеристикой. Для этого строим график рассматриваемой функции y = f(x) и графически установим значение входного параметра x_0, при котором выполняется условие – $y = f(x_0) \approx 0$.

Для реализации второго этапа (уточнения значения корня с заданной степенью точности E) необходимо от исходного нелинейного уравнения f(x) = 0 перейти к эквивалентному уравнению вида x = φ(x). Переход осуществляется умножением левой и правой части уравнения f(x) = 0 на произвольную постоянную K и прибавлением к обеим частям уравнения значения x:

$$x + K \cdot f(x) = x + 0 \cdot K.$$

После преобразований в итоге получаем:

$$x = \varphi(x), \qquad (3)$$

где $\varphi(x) = x + K \cdot f(x)$. Значение произвольной постоянной K устанавливается подбором, исходя из выполнения условия: $0 < |\varphi'(x)| < 1$, при этом сходимость итерационного процесса будет двухсторонней. Производную φ′(x) можно определить по формуле:

$$\varphi'(x) = 1 + K \cdot f'(x). \qquad (4)$$

После приведения исходного уравнения f(x) = 0 к эквивалентному $x = x + K \cdot f(x)$, с применением последнего реализуется итерационный процесс. Для нахождения уточненного значения корня в рамках первой итерации в правую часть уравнения, эквивалентного исходному, вместо x подставим x_0 (начальное приближение):

$$x_1 = x_0 + K \cdot f(x_0),$$

где x_1 – уточненное значение корня с первой итерации. Если выполняется условие $|f(x_1)| \le E$ и $|f'(x_1)| < 1$, то искомое значение корня x_K, будет соответствовать приближению x_1 с первой итерации ($x_K = x_1$), в противном случае находят новое уточнение корня нелинейного уравнения, продолжая итерационный процесс. Предположим, что $|f(x_1)| \le E$ и $|f'(x_1)| < 1$ не выполнились, тогда реализуется вторая итерация, в рамках которой:

$$x_2 = x_1 + K \cdot f(x_1),$$

где x_2 – уточненное значение корня со второй итерации. Проверяется выполнение условий $|f(x_2)| \le E$ и $|f'(x_2)| < 1$. Предположим, условия выполнялись, следовательно, искомое значение корня x_K будет соответствовать приближению x_2 со второй итерации ($x_K = x_2$). Итерационный процесс завершен. Количество итераций будет зависеть от правильности нахождения точки начального приближения x_0 и произвольной постоянной K. Таким образом, итерационный процесс метода простых итераций по зависимости $x_n = x_{n-1} + K \cdot f(x_{n-1})$, где x_n – уточненное значение корня с n-ой итерации, а x_{n-1} – уточненное значение корня с предыдущей (n – 1)-ой итерации, будет продолжаться до тех пор, пока не выполнятся условия $|f(x_n)| \le E$ и $|f'(x_n)| < 1$. В соответствии с вышеизложенным, алгоритм метода простых итераций может быть представлен в виде блок-схемы на рис. 2.5. Осуществим практическую реализацию метода простых итераций по постановке задачи (см. пример 8).

РЕШЕНИЕ. Из графика (см. рис. 2.3) видно, что точкой начального приближения является точка 3 пересечения кривых 1 и 2. Значение усилия растяжения болта в этой точке, решая задачу графически, соответствует 52 кгс (N_0 = 52 кгс). Таким образом, первый этап метода простых итераций реализован. Перейдем к реализации второго этапа. Приведем исходное нелинейное уравнение $\sqrt{0{,}18 \cdot N} - 3 = 0$ к эквивалентному $N = N + K \cdot (\sqrt{0{,}18 \cdot N} - 3)$. Установим значение коэффициента K. Для этого первоначально определим производную первого порядка эквивалентного уравнения $\varphi'(N) = 1 + K \cdot f'(N)$, где f’(N) – производная первого порядка исходного нелинейного уравнения:

$$f'(N) = (\sqrt{0{,}18 \cdot N} - 3)' = \sqrt{0{,}18 \cdot N}' - 3' = ((0{,}18 \cdot N)^{0{,}5})' - 3' =$$

$$(0{,}18^{0{,}5} \cdot N^{0{,}5})' - 3' = (0{,}424 \cdot N^{0{,}5})' - 3' = 0{,}424' \cdot N^{0{,}5} + 0{,}424 \cdot (N^{0{,}5})' - 3' =$$

$$= 0 + 0{,}212 \cdot N^{-0{,}5} + 0 = 0{,}212 \cdot N^{-0{,}5}.$$

Следовательно, $\varphi'(N) = 1 + K \cdot (0{,}212 \cdot N^{-0{,}5})$. Выберем значение произвольной постоянной K равным «-10» ($K = -10$) и при $N = N_0 = 52$ рассчитаем $\varphi'(N)$: $\varphi'(52) = 1 + (-10) \cdot (0{,}212 \cdot 52^{-0{,}5}) = 0{,}706$. Проверим выполнение условия $0 < |\varphi'(N)| < 1$, т. е. $0 < |\varphi'(52)| < 1$. Условие выполняется ($0 < |0{,}706| < 1$), следовательно, произвольная постоянная выбрана правильно. Реализуем первую итерацию, в рамках которой $N_1 = N_0 + K \cdot (\sqrt{0{,}18 \cdot N_0} - 3)$. После подстановки в правую часть уравнения $N_0 = 52$, $K = -10$ получаем: N_1 = 51,406 кгс. Проверим выполнение условий $|f(N_1)| \le E$ и $|f'(N_1)| < 1$, где E = 0,01. Рассчитаем значение функции и ее производной при N_1=51,406 кгс: f(51,406) = 0,0418, f′(51,406) = 0,0295, следовательно, условие $|0{,}0418| \le 0{,}01$ не выполняется, а

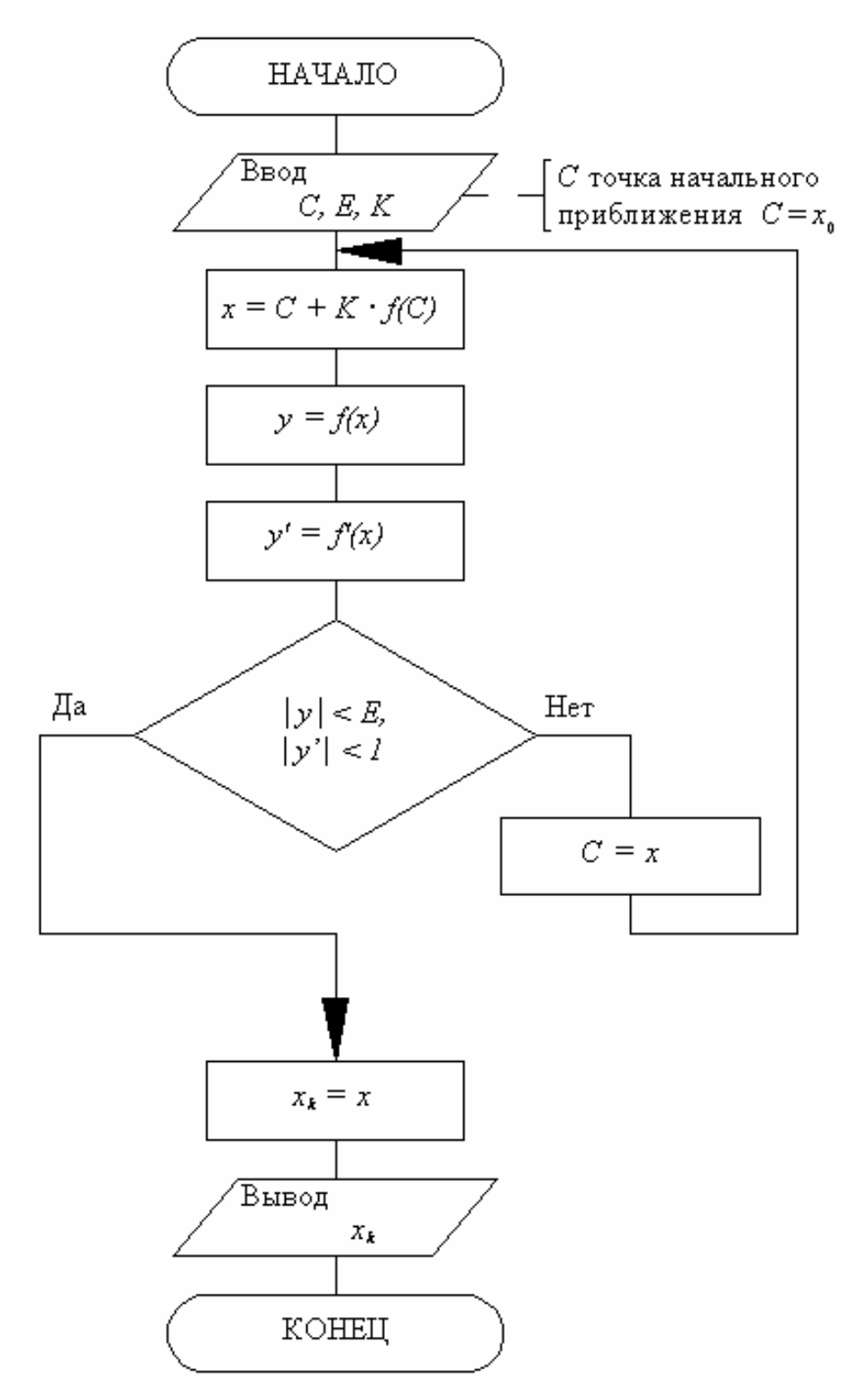


Рис. 2.5. Блок-схема алгоритма метода простых итераций

условие $|0{,}0295| < 1$ выполнилось. Реализуем вторую итерацию, в рамках которой $N_2 = N_1 + K \cdot (\sqrt{0{,}18 \cdot N_1} - 3)$. После подстановки в правую часть уравнения $N_0 = 51{,}406$, $K = -10$ получаем: $N_2 = 50{,}987$ кгс. Проверим выполнение условий $|f(N_2)| \le E$ и $|f'(N_2)| < 1$. Рассчитаем значение функции и ее производной при $N_2 = 50{,}987$: f(50,987) = 0,0294, f′(50,987) = 0,0296, вновь условие $|0{,}0294| \le 0{,}01$ не выполнилось. Реализуем третью итерацию, в рамках которой $N_3 = N_2 + K \cdot (\sqrt{0{,}18 \cdot N_2} - 3)$, $N_3 = 50{,}692$ кгс. Рассчитаем значение функции и ее производной при $N_3 = 50{,}692$ кгс: f(50,692) = 0,0206, f′(50,692) = = 0,0297, вновь условие $|0{,}0206| \le 0{,}01$ не выполнилось. Реализуем четвертую итерацию, в рамках которой $N_4 = N_3 + K \cdot (\sqrt{0{,}18 \cdot N_3} - 3)$, $N_4 = 50{,}485$ кгс. Рассчитаем значение функции и ее производной при $N_4 = 50{,}485$ кгс: f(50,485) = = 0,0145, f′(50,485) = 0,0298. Условие $|0{,}0145| \le 0{,}01$ не выполнилось. Реализуем пятую итерацию, $N_5 = N_4 + K \cdot (\sqrt{0{,}18 \cdot N_4} - 3)$, $N_5 = 50{,}339$ кгс. Рассчитаем значение функции и ее производной при $N_5 = 50{,}339$ кгс: f(50,339) = 0,010, f′(50,339) = 0,0298. Оба условия выполняются $|f(N_5)| = |f(50{,}339)| = |0{,}010| \le 0{,}01$, $|f'(N_5)| = |f'(50{,}339)| = |0{,}0298| \le 1$. Следовательно, искомое значение корня N_K будет соответствовать приближению N_5 с пятой итерации ($N_K = N_5 = 50{,}339$ кгс). При точном решении ($N_K = 50$ кгс), относительная погрешность реализации метода простых итераций составляет равной 0,7 % при заданной степени точности E = 0,01.

В соответствии с постановкой задачи (см. пример 8) разработаем алгоритм (рис. 2.6) и программу USILIE1 ее решения на языке TURBO PASCAL.

ПРОГРАММА USILIE1

(постановка задачи см. пример 8, алгоритм см. рис. 2.6)

```
Program Usilie1;
Uses Crt;
Var N,E,K,Nut,y,ypr,Nk:Real;
Begin
Clrscr;
Write(’Задайте степень точности E:’);
Readln(E);
Write(’Введите начальное приближение’);
Writeln(’усилия растяжения болта N0 в кгс:’);
Readln(N);
```

```
Write(’Введите значение произвольной постоянной ’);
Writeln(’K:’);
Readln(K);
Repeat
Nut:=N+(K*(Sqrt(0.18*N)-3));
y:=Sqrt(0.18*Nut)-3;
ypr:=0.212*Nut;
N:=Nut
Until (Abs(y)<=E) And (Abs(ypr)<1);
Nk:=Nut;
Write(’Искомое усилие растяжения болта ’);
Writeln(’ Nk = ’,Nk:5:2,’ кгс.’);
Readkey;
End.
```

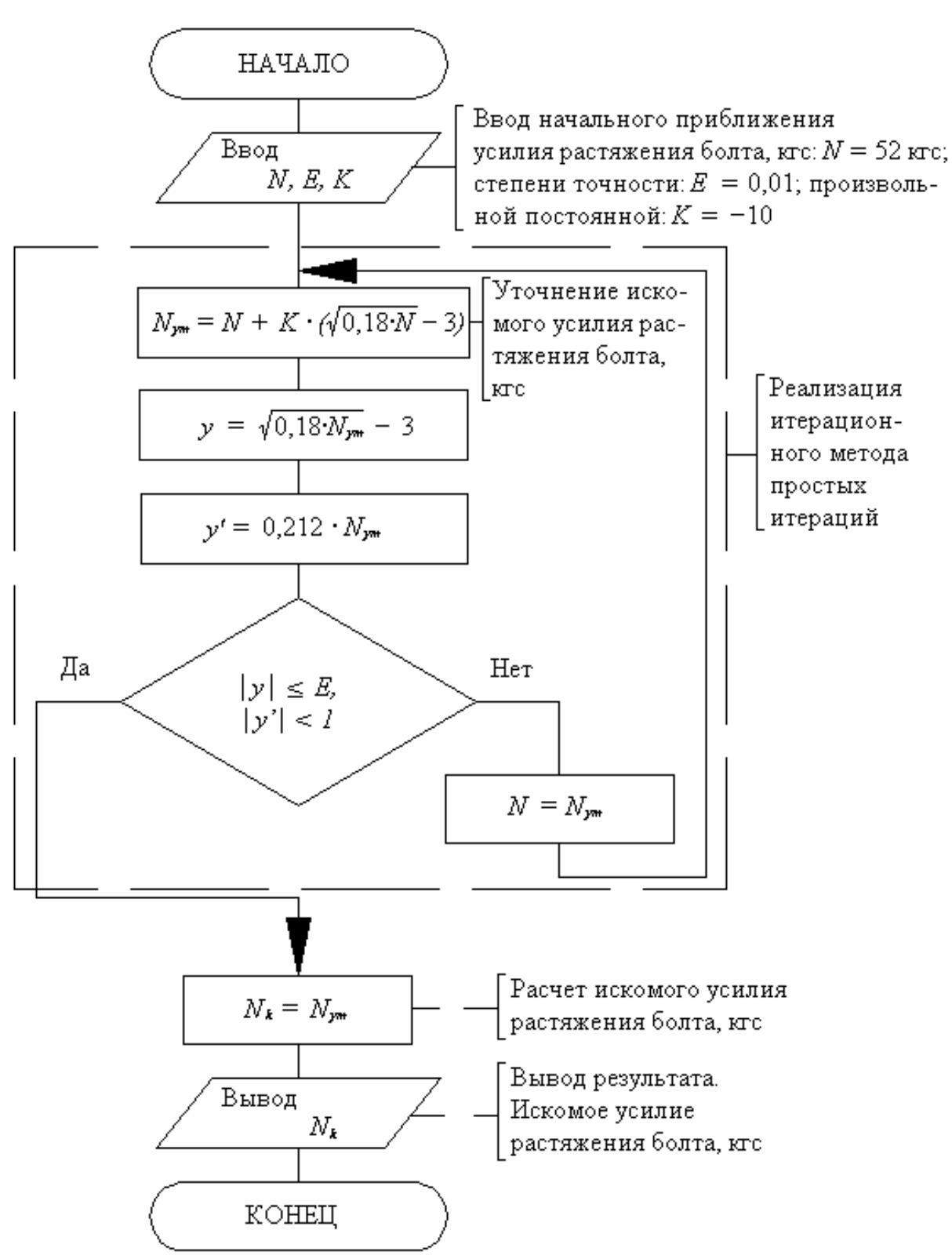


Рис. 2.6. Блок-схема алгоритма нахождения усилия растяжения болта методом простых итераций (постановка задачи см. пример 8)

3. МЕТОДЫ НАХОЖДЕНИЯ ПАРАМЕТРОВ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ С ЛИНЕНОЙ ХАРАКТЕРИСТИКОЙ

3.1. Основные понятия, классификация и общий подход к реализации методов нахождения параметров технологических систем с линейной характеристикой

В ряде случаев на практике в области технологии машиностроения можно столкнуться с необходимостью нахождения таких значений входных параметров технологических систем, которые будут являться общим решением сразу для нескольких уравнений, устанавливающих связь этих входных параметров с выходными. Причем, сами уравнения могут рассматриваться, как уравнения с линейной характеристикой (линейные уравнения). Нахождение значений входных параметров предполагает решение системы, состоящей из нескольких линейных уравнений.

Рассмотрим систему n линейных уравнений:

$$\left.\begin{array}{l} a_{11}\cdot x_1 + a_{12}\cdot x_2 + ... + a_{1j}\cdot x_n = b_1 \\ a_{21}\cdot x_1 + a_{22}\cdot x_2 + ... + a_{2j}\cdot x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i1}\cdot x_1 + a_{i2}\cdot x_2 + ... + a_{ij}\cdot x_n = b_n \end{array}\right\} \qquad (5)$$

или в векторно-матричной форме:

$$A\cdot x = b, \qquad (6)$$

где A – матрица совокупности коэффициентов a_{ij}:

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1j} \\ a_{21} & a_{22} & ... & a_{2j} \\ ..... & & & \\ a_{i1} & a_{i2} & ... & a_{ij} \end{bmatrix};$$

x – вектор-столбец неизвестных входных параметров:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix};$$

b – вектор-столбец свободных членов:

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}.$$

Решение этой системы предполагает нахождение таких значений x, при подстановке которых в каждое из уравнений при известных A и b будет обеспечиваться тождество (равенство левой и правой частей каждого из уравнений). При определенных условиях может обеспечиваться либо единственное решение системы, либо бесчисленное множество решений, либо отсутствие решения, либо плохая обусловленность в получении решения.

Установить факт возможности или невозможности решить систему линейных уравнений можно, например, по величине определителя D матрицы A совокупности коэффициентов a_{ij}. Для систем состоящих из двух линейных уравнений матрица A имеет вид:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

а ее определитель (определитель второго порядка):

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{21} \cdot a_{12}.$$

Для систем, состоящих из трех линейных уравнений, матрица A имеет вид:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix},$$

а ее определитель (определитель третьего порядка):

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} =$$

$$= a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{21} \cdot a_{32} \cdot a_{13} - a_{31} \cdot a_{22} \cdot a_{13} - a_{21} \cdot a_{12} \cdot a_{33} - a_{32} \cdot a_{23} \cdot a_{11}.$$

Для треугольной матрицы A с a_{ij} ниже главной диагонали равными нулю определитель D можно рассчитать по формуле:

$$D = a_{11} \cdot a_{22} \cdot \dots \cdot a_{ij},$$

где i = j.

Для единичной матрицы A с a_{ij} по главной диагонали равными единице определитель D так же равен единице (D = 1). Для нулевой матрицы A с a_{ij},

равными нулю (частный случай), определитель D равен нулю (D = 0). Если $D \neq 0$, то система линейных уравнений будет иметь единственное решение. При D = 0 система линейных уравнений либо не имеет решения, либо их будет бесчисленное множество. Когда $D \approx 0$, система линейных уравнений называется плохо обусловленной.

Факт возможности или не возможности решить систему линейных уравнений можно установить также по соотношениям между коэффициентами системы. Например, пусть нам дана система:

$$\left.\begin{aligned} a_{11} \cdot x + a_{12} \cdot y = b_1 \\ a_{21} \cdot x + a_{22} \cdot y = b_2 \end{aligned}\right\}.$$

Решением этой системы являются координаты точки пересечения двух прямых на плоскости. Возможны четыре случая:

1) прямые пересекаются (единственное решение);

2) прямые проходят параллельно друг другу (решения нет);

3) прямые совпадают (бесчисленное множество решений);

4) прямые проходят почти параллельно друг другу (нет определенности в решении, координаты точки пересечения прямых чувствительны к изменениям коэффициентов системы).

Первый случай реализуется при выполнении условия: $(a_{11}/a_{21}) \neq (a_{12}/a_{22})$, при этом D получается не равным нулю $(D \neq 0)$. Второй случай реализуется при выполнении условия: $(a_{11}/a_{21}) = (a_{12}/a_{22}) \neq (b_1/b_2)$, при этом D получается равным нулю (D = 0). Третий случай характерен при выполнении условия: $(a_{11}/a_{21}) = (a_{12}/a_{22}) = (b_1/b_2)$, при этом D также получается равным нулю (D = 0). Последний случай возникает, когда определитель матрицы A совокупности коэффициентов a_{ij} получается примерно равным (близким) нулю ($D \approx 0$).

Таким образом, перед тем как реализовать тот или иной метод решения систем линейных уравнений, необходимо убедиться в возможности или невозможности решения системы либо по значению D, либо через соотношения между коэффициентами a_{ij} матрицы A.

Методы решения систем линейных уравнений по принципам организации вычислений делятся на два класса: прямые (точные) и итерационные (приближенные). Прямыми называются методы, которые в предположении, что вычисления ведутся без округлений, позволяют получить точное решение за конечное число арифметических операций по известным расчетным формулам. К числу таких методов относятся метод Крамера и метод Гаусса. Итерационные методы даже в предположении, что вычисления ведутся без округлений, дают приближенное решение системы с наперед заданной точностью. Точное решение в данном случае теоретически может быть получено как результат бесконечного процесса. Характерным представителем этого класса является метод Зейделя.

3.2. Метод Крамера в нахождении технологических параметров системы уравнений с линейными характеристиками

Пусть нам дана система двух линейных уравнений, как наиболее часто реализуемый случай:

$$\left.\begin{aligned} a_{11}\cdot x_1 + a_{12}\cdot x_2 = b_1 \\ a_{21}\cdot x_1 + a_{22}\cdot x_2 = b_2 \end{aligned}\right\}.$$

Если матрица A совокупности коэффициентов a_{ij} невырожденная, т. е. если ее определитель D не равен нулю, то система уравнений имеет единственное решение, или если выполняется условие: $(a_{11}/a_{21}) \neq (a_{12}/a_{22})$. Решением системы является вектор-столбец x совокупности коэффициентов x_i. Для нахождения неизвестных x_i можно воспользоваться конечными формулами:

$$x_1 = \frac{D_1}{D}, \tag{7}$$

$$x_2 = \frac{D_2}{D}, \tag{8}$$

где D – определитель матрицы совокупности коэффициентов a_{ij}:

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}\cdot a_{22} - a_{21}\cdot a_{12};$$

D_1 – определитель матрицы совокупности коэффициентов b_i и a_{i2}:

$$D_1 = \begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix} = b_1\cdot a_{22} - b_2\cdot a_{12};$$

D_2 – определитель матрицы совокупности коэффициентов b_i и a_{i1}:

$$D_2 = \begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix} = a_{11}\cdot b_2 - a_{21}\cdot b_1.$$

Таким образом, реализация метода Крамера сводится к нахождению неизвестных через значения определителей второго порядка совокупности коэффициентов a_{ij}; b_i и a_{ij}. Как правило, метод Крамера используется для решения систем, состоящих не более чем из двух уравнений. С вычислительной точки зрения метод Крамера неэффективен для решения систем с большим чем два числом уравнений, так как его реализация потребует выполнения значительного количества арифметических операций и соответственно больших затрат машинного времени для нахождения определителей более высоких порядков. Как показывает практика, метод Крамера весьма чувствителен к ошибкам округления. Алгоритм метода Крамера в нахождении технологических параметров системы уравнений с линейными характеристиками представлен в виде блок-схемы на рис. 3.1.

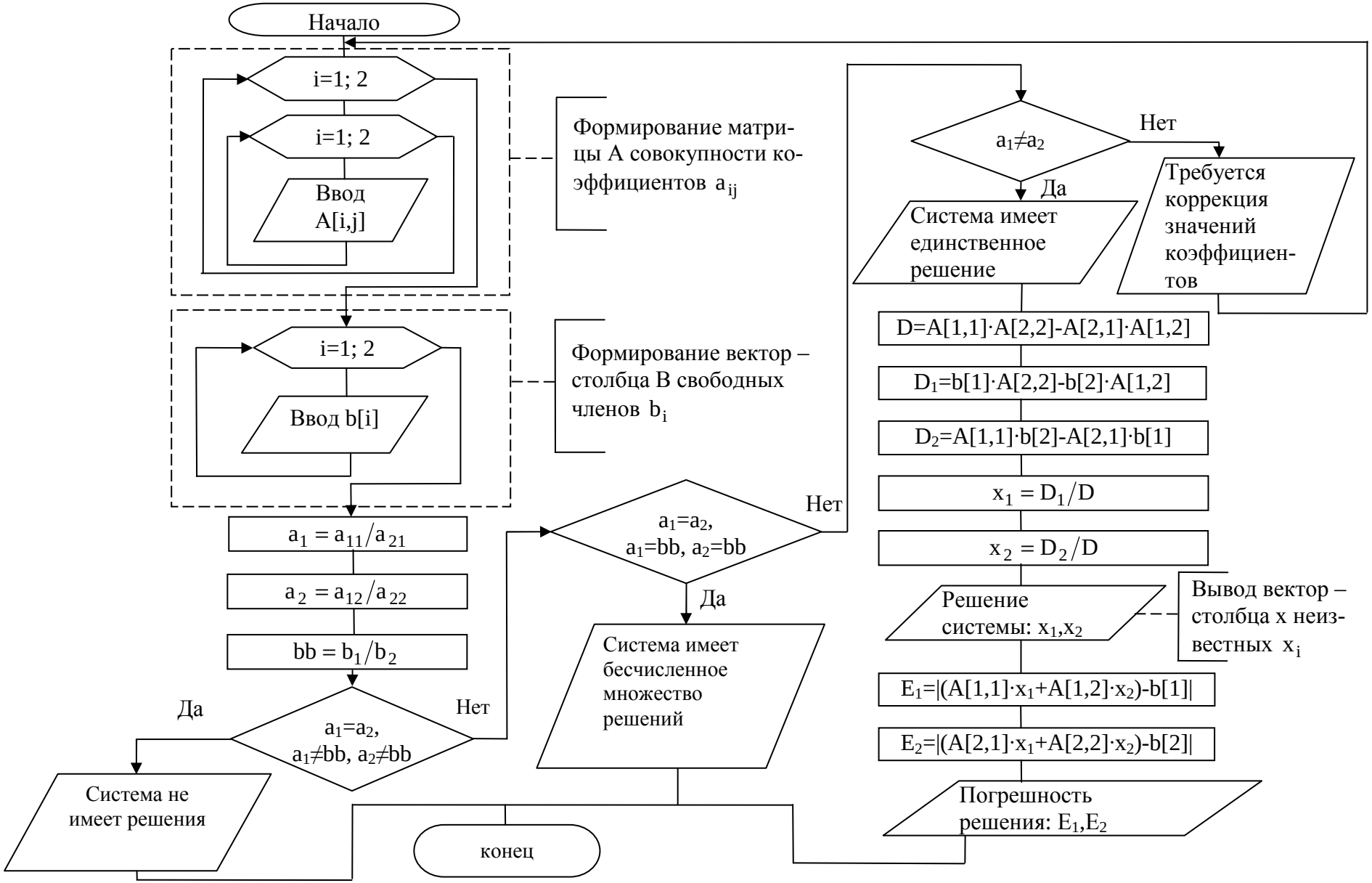


Рис. 3.1. Блок-схема алгоритма метода Крамера (решение системы $A \cdot x = b$ из двух линейных уравнений)

Пример 9. В рамках технологической подготовки операции электроэрозионного вырезания на станке с ЧПУ для разработки управляющей программы формообразования исполнительной поверхности шаблона необходимо определить положение узловой точки М (ее координаты), опираясь на исходные данные, представленные на рис. 3.2.

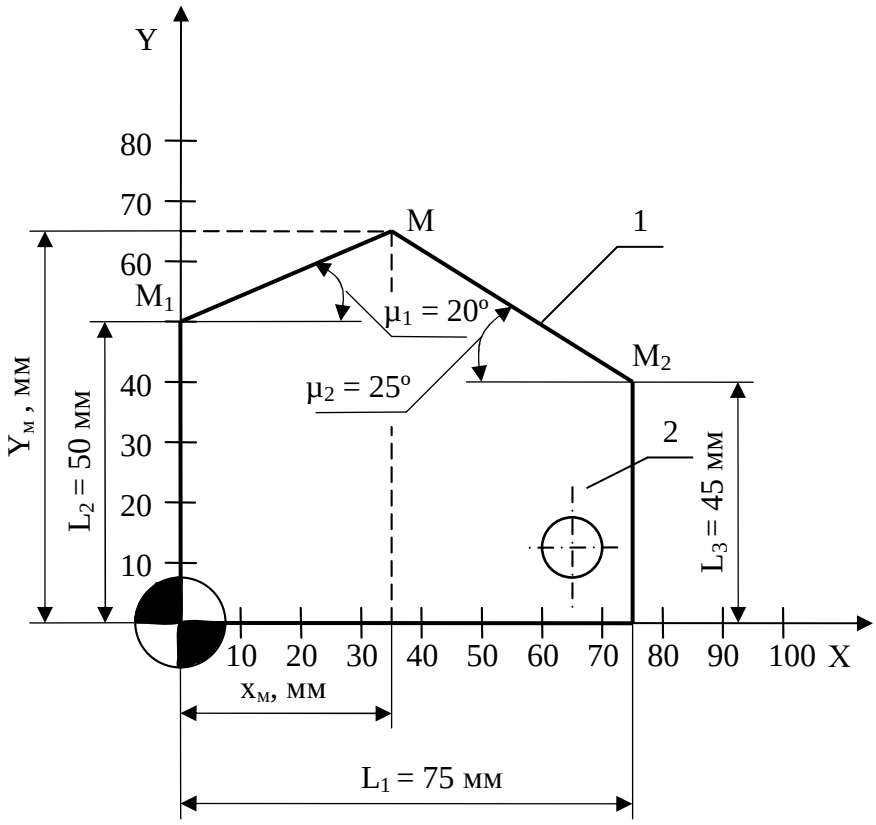


Рис. 3.2. Расчетная схема к определению местоположения узловой точки М:
1 – торцовый профиль исполнительной поверхности шаблона; 2 – шаблон

РЕШЕНИЕ. Из расчетной схемы (см. рис. 3.2) видно, что торцовый профиль 1 исполнительной поверхности шаблона 2 представляет из себя ломаную, состоящую из двух прямолинейных отрезков M_1M и MM_2. Каждый из отрезков описывается уравнением прямой. Точка М – это точка пересечения отрезков. Следовательно, если выявить уравнение прямой для M_1M и уравнение прямой для MM_2 и решить систему этих двух линейных уравнений, то будет найдено искомое положение точки $M(x_м,y_м)$.

Система двух линейных уравнений имеет следующий вид:

$$\left.\begin{aligned} y_м &= к_1 \cdot x_м + b_1 \\ x_м &= к_2 \cdot x_м + b_2 \end{aligned}\right\},$$

где $x_м, y_м$ – искомые координаты точки М, мм; $к_1$ – угловой коэффициент прямой, проходящей через точки M_1 и М; $к_2$ – угловой коэффициент прямой, проходящей через точки М и M_2; b_1 – коэффициент смещения прямой, прохо-

дящей через точки M_1 и M; b_2 – коэффициент смещения прямой, проходящей через точки M и M_2.

Величину углового коэффициента $к_1$ можно определить по формуле:

$$к_1 = tg(\mu_1),$$

где μ_1 – угол наклона рассматриваемой прямой, проходящей через точки M_1 и M, к оси x: $\mu_1 = 20^o$. Следовательно, $к_1 = 0{,}364$.

Величину углового коэффициента $к_2$ определим по формуле:

$$к_2 = tg(180° - \mu_2),$$

где μ_2 – угол наклона рассматриваемой прямой, проходящей через точки M и M_2, к оси x: $\mu_2 = 25^o$. Следовательно, $к_2 = -0{,}466$.

Величину коэффициента смещения прямой b_1 рассчитаем по формуле:

$$b_1 = y_{м_1} - к_1 \cdot x_{м_1},$$

где $x_{м_1}, y_{м_1}$ – координаты точки M_1, мм; $x_{м_1} = 0$, а $y_{м_1} = L_2 = 50$мм. Следовательно, $b_1 = 50$.

Величину коэффициента смещения прямой b_2 рассчитаем по зависимости:

$$b_2 = y_{м_2} - к_2 \cdot x_{м_2},$$

где $x_{м_2}, y_{м_2}$ – координаты точки M_2: $x_{м_2} = L_1 = 75$ мм, а $y_{м_2} = L_3 = 45$ мм. Следовательно, $b_2 = 79{,}950$. В результате, система имеет вид:

$$\left.\begin{aligned} y_м &= 0{,}364 \cdot x_м + 50 \\ y_м &= -0{,}466 \cdot x_м + 79{,}950 \end{aligned}\right\}.$$

Условно обозначим $y_м$ через x_2, а $x_м$ через x_1 и перенесем $x_м$ в левую часть каждого уравнения. В итоге получим:

$$\left.\begin{aligned} -0{,}364 \cdot x_1 + x_2 &= 50 \\ 0{,}466 \cdot x_1 + x_2 &= 79{,}950 \end{aligned}\right\}.$$

В этой системе матрица A совокупности коэффициентов имеет вид:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} -0{,}364 & 1 \\ 0{,}466 & 1 \end{bmatrix}.$$

Вектор-столбец свободных членов B имеет вид:

$$B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 50 \\ 79{,}950 \end{bmatrix}.$$

Вектор-столбец неизвестных x имеет вид:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Убедимся в том, что система имеет единственной решение и матрица A невырожденная. Проверим выполнение условия $(a_{11}/a_{21}) \neq (a_{12}/a_{22})$: $-0{,}364/0{,}466 \neq 1/1$.

Условие выполняется. Найдем определитель D матрицы A совокупности коэффициентов a_{ij}:

$$D = a_{11} \cdot a_{22} - a_{21} \cdot a_{12};$$
$$D = -0{,}364 \cdot 1 - 0{,}466 \cdot 1 = -0{,}83.$$

Найдем определитель D_1 матрицы совокупности коэффициентов b_i и a_{i2}:

$$D_1 = b_1 \cdot a_{22} - b_2 \cdot a_{12};$$
$$D_1 = 50 \cdot 1 - 79{,}950 \cdot 1 = -29{,}95.$$

Найдем определитель D_2 матрицы совокупности коэффициентов b_i и a_{i1}:

$$D_2 = a_{11} \cdot b_2 - a_{22} \cdot b_1;$$
$$D_2 = -0{,}364 \cdot 79{,}950 - 0{,}466 \cdot 50 = -52{,}402.$$

По найденным D, D_1 и D_2 установим значения x_1 и x_2:

$$x_1 = \frac{D_1}{D}; \qquad x_1 = \frac{-29{,}95}{-0{,}83} = 36{,}084;$$

$$x_2 = \frac{D_2}{D}; \qquad x_2 = \frac{-52{,}402}{-0{,}83} = 63{,}135.$$

Таким образом, искомое положение точки M (см. рис. 3.2) задано координатами $x_м$ = 36,084 мм, а $y_м$ = 63,125 мм. Осуществим проверку правильности решения системы. Подставим $x_м$ и $y_м$ в систему

$$\left.\begin{aligned} y_м &= 0{,}364 \cdot x_м + 50 \\ y_м &= -0{,}466 \cdot x_м + 79{,}950 \end{aligned}\right\}$$

и проверим выполнение тождеств:

$$\left.\begin{aligned} 63{,}135 &= 0{,}364 \cdot 36{,}084 + 50 \\ 63{,}135 &= -0{,}466 \cdot 36{,}084 + 79{,}950 \end{aligned}\right\};$$

$$\left.\begin{aligned} 63{,}135 &= 63{,}13457 \\ 63{,}135 &= 63{,}13485 \end{aligned}\right\}.$$

Тождества обеспечиваются с погрешностью $|E_1|$ = 0,00043 для первого уравнения и $|E_2|$ = 0,00015 для второго уравнения в системе.

В соответствии с постановкой задачи (см. пример 9) разработаем программу SLU_KRAMER ее решения на языке TURBO PASCAL.

ПРОГРАММА SLU_KRAMER
(постановка задачи см. пример 9, алгоритм см. рис. 3.1)

```
Program SLU_KRAMER;
Uses Crt;
Label 1;
Var i, j: Byte;
```

```
    a1, a2, bb, D, D1, D2, x1, x2, xm, ym, E1, E2: Real;
    A, B : Array[1..5] of Real;
Begin
Clrscr;
1:Writeln('Введите матрицу A совокупности коэффициентов a_ij');
For i:=1 To 2 Do Begin
For i:=1 To 2 Do Begin
Read(' ',a[i,j],' ');{a11 = -0,364, a12 = 1, a21 = 0,466, a22 = 1}
End;End;
Writeln('Введите вектор-столбец B совокупности коэффициентов b_i.');
For i:=1 To 2 Do Begin
Read(' ',b[i],' ');{b1=50, b2=79,950}
End;
a1:=a[1,1]/a[2,1];
a2:=a[1,2]/a[2,2];
bb:=b[1]/b[2];
If (a1=a2) And (a1<> bb) And (a2<>bb) Then
Begin Writeln('Система не имеет решения.');
End;
If (a1=a2) And (a1=bb) And (a2=bb) Then Begin
Writeln('Система имеет бесчисленное множество решений .');
End;
If (a1<>a2) Then Begin Writeln('Система имеет единственное решение.');
D:=a[1,1]*a[2,2]-a[2,1]*a[1,2];
D1:=b[1]*a[2,2]-b[2]*a[1,2];
D2:=a[1,1]*b[2]-a[2,1]*b[1];
x1:=D1/D;
x2:=D2/D;
xm:=x1;
ym:=x2;
Writeln('Координаты точки M: ');
Writeln('хм =',xm:5:3,', ум =',ym:5:3,'.');
Writeln('Погрешность расчета:');
E1:=Abs((a[1,1]*x1+a[1,2]*x2)-b[1]);
E2:=Abs((a[2,1]*x1+a[2,2]*x2)-b[2]);
Writeln('E1 = ',E1:6:5,', E2 = ',E2:6:5,'.');
End Else Begin
Writeln('Требуется коррекция значений коэффициентов.');
Goto 1; End;
Readkey;
End.
```

3.3. Метод Гаусса в нахождении технологических параметров системы уравнений с линейными характеристиками

Метод Гаусса реализуется в два этапа. На первом этапе (прямой ход) последовательным исключением неизвестных составляется преобразованная эквивалентная система уравнений с треугольной матрицей А совокупности коэффициентов a_{ij}, в которой все элементы ниже главной диагонали обращены в нуль. При этом одно из уравнений содержит только одну неизвестную x_n, а в каждом следующем добавляется еще по одной переменной. На втором этапе (обратный ход) решается преобразованная система, последовательным определением с помощью элементарных вычислений значений неизвестных, начиная с x_n и заканчивая x_1.

В описательной форме алгоритм реализации метода Гаусса имеет следующую последовательность предписаний:

1) Среди элементов a_{ij} матрицы А, где i, j =1 , …, n, выбирается наибольший по абсолютной величине элемент a_{pq}. Этот элемент называется главным. При этом строка матрицы А с номером p называется главной. Если предположить, что наибольшим по абсолютной величине является элемент a_{11}, то $a_{pq} = a_{11}$. Если это условие не выполняется, то необходимо поменять местами первую строку со строкой p и первый столбец со столбцом q и осуществить соответствующую перенумерацию коэффициентов и неизвестных.

2) Полученное на предыдущем этапе уравнение системы делят на $a_{11} = a_{pq}$:

$$x_1 + C_{12} \cdot x_2 + ... + C_{1n} = d_1,$$

где $C_{1j} = a_{1j} / a_{11}$, а $d_1 = b_1 / a_{11}$.

3) Осуществляется исключение неизвестной переменной x_1 из каждого уравнения исходной системы, начиная со второго, путем вычитания уравнения со второго этапа, умноженного на коэффициент a_{i1}, где i = 2, …, n, при x_1 в соответствующем уравнении. Отбрасывается главная строка и первый столбец матрицы А и получается преобразованная система:

$$\left.\begin{array}{l} C_{22} \cdot x_2 + C_{23} \cdot x_3 + ... + C_{2n} \cdot x_n = d_2 \\ ..... \\ C_{n2} \cdot x_2 + C_{n3} \cdot x_3 + ... + C_{nn} \cdot x_n = d_n \end{array}\right\},$$

где $C_{ij} = a_{ij} - C_{1j} \cdot a_{i1}$; $d_i = b_i - d_1 \cdot a_{i1}$ при I = 2, …, n.

4) Над системой, полученной на третьем этапе, повторяют операции этапов 1–3, в результате чего получается система уравнений, содержащая неизвестные x_3, …, x_n. Такие преобразования необходимо продолжать до тех пор, пока не получится уравнение с одним неизвестным $C_{nn} \cdot x_n = d_n$.

5) После объединения всех уравнений, получается система с треугольной матрицей, эквивалентная исходной:

$$\left.\begin{array}{l} x_1 + C_{12} \cdot x_2 + C_{13} \cdot x_3 + ... + C_{1n} \cdot x_n = d_1 \\ x_2 + C_{23} \cdot x_3 + ... + C_{2n} \cdot x_n = d_2 \\ ..... \\ C_{nn} \cdot x_n = d_n \end{array}\right\}.$$

6) Реализуется обратный ход, в рамках которого из системы, полученной на пятом этапе, неизвестные x_1, …, x_n определяются в обратном порядке по формулам:

$$x_n = d_n / C_{nn};$$
$$x_{n-1} = d_{n-1} - C_{n-1,n} \cdot x_n;$$
$$.....$$
$$x_1 = d_1 - C_{12} \cdot x_2 - C_{13} \cdot x_3 - ... - C_{1n} \cdot x_n.$$

В сравнении с методом Крамера метод Гаусса можно безболезненно применять для решения систем, состоящих как из двух, так и более линейных уравнений. Он является простым в своей реализации и легко программируемым.

Осуществим практическую реализацию метода Гаусса по постановке задачи (см. пример 9).

РЕШЕНИЕ. В результате ранее выполненных преобразований (см. п. 3.2, пример 9) имеем систему, состоящую из двух линейных уравнений:

$$\left.\begin{array}{l} -0{,}364 \cdot x_1 + x_2 = 50 \\ 0{,}466 \cdot x_1 + x_2 = 79{,}950 \end{array}\right\},$$

в которой

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} -0{,}364 & 1 \\ 0{,}466 & 1 \end{bmatrix},$$

$B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 50 \\ 79{,}950 \end{bmatrix}$ и $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ – вектор столбец неизвестных.

Приведем систему линейных уравнений к виду с треугольной матрицей A совокупности коэффициентов a_{ij}, т. е.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix}.$$

Исключим из второго уравнения системы x_1, реализовав прямой ход метода Гаусса. Для этого предварительно умножим первое уравнение на коэффициент «$-a_{21}/a_{11}$»:

$$\left(-\frac{a_{21}}{a_{11}} \cdot a_{11}\right) \cdot x_1 + \left(-\frac{a_{21}}{a_{11}} \cdot a_{12}\right) \cdot x_2 = \left(-\frac{a_{21}}{a_{11}}\right) \cdot b_1,$$

в результате при известных A и B умеем уравнение:

$$-0{,}466 \cdot x_1 + 1{,}280 \cdot x_2 = 64{,}011.$$

В общем виде оно имеет вид:

$$a^*_{11} \cdot x_1 + a^*_{12} \cdot x_2 = b^*_1.$$

Полученное уравнение прибавим к второму исходной системы:

$$+\begin{cases} a_{21} \cdot x_1 + a_{22} \cdot x_2 = b_2; \\ a^*_{11} \cdot x_1 + a^*_{12} \cdot x_2 = b^*_1, \end{cases} \text{ т. е.}$$

$$+\begin{cases} 0{,}466 \cdot x_1 + x_2 = 79{,}950; \\ -0{,}466 \cdot x_1 + 1{,}280 \cdot x_2 = 64{,}011. \end{cases}$$

В результате получаем уравнение с исключенной из него переменной x_1:

$$2{,}280 \cdot x_2 = 143{,}961.$$

В общем виде оно имеет вид:

$$a^*_{22} \cdot x_2 = b^*_2.$$

Таким образом, получаем систему уравнений:

$$\left.\begin{aligned} a_{11} \cdot x_1 + a_{12} \cdot x_2 &= b_1 \\ a^*_{22} \cdot x_2 &= b^*_2 \end{aligned}\right\}, \text{ т. е.}$$

$$\left.\begin{aligned} -0{,}364 \cdot x_1 + x_2 &= 50 \\ 2{,}280 \cdot x_2 &= 143{,}961 \end{aligned}\right\},$$

в которой матрица A совокупности коэффициентов a_{ij} является треугольной:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a^*_{22} \end{bmatrix} = \begin{bmatrix} -0{,}364 & 1 \\ 0 & 2{,}280 \end{bmatrix}.$$

В итоге прямой ход метода Гаусса реализован.

Найдем неизвестные x_1, x_2, реализовав обратных ход метода Гаусса. Для этого из второго уравнения, полученной системы с треугольной матрицей A, определим x_2:

$$x_2 = \frac{b^*_2}{a^*_{22}},$$

$$x_2 = \frac{143{,}961}{2{,}280} = 63{,}141.$$

При известном значении x_2 и подстановке x_2 в первое уравнение системы определим x_1:

$$x_1 = \frac{b_1 - a_{12} \cdot x_2}{a_{11}},$$

$$x_1 = \frac{50 - 1 \cdot 63{,}141}{-0{,}364} = 36{,}102.$$

Таким образом, искомое положение точки М (см. рис. 3.2) будет задано координатами: $x_м = x_1 = 36{,}102$ мм, а $y_м = x_2 = 63{,}141$ мм.

Осуществим проверку правильности решения исходной системы (см. п. 3.2, пример 9):

$$\left.\begin{aligned} y_м &= 0{,}364 \cdot x_м + 50 \\ y_м &= -0{,}466 \cdot x_м + 79{,}950 \end{aligned}\right\}.$$

Подставим $x_м$ и $y_м$ в эту систему и проверим выполнение тождеств:

$$\left.\begin{aligned} 63{,}141 &= 0{,}364 \cdot 36{,}102 + 50 \\ 63{,}141 &= -0{,}466 \cdot 36{,}102 + 79{,}950 \end{aligned}\right\};$$

$$\left.\begin{aligned} 63{,}141 &= 63{,}141 \\ 63{,}141 &= 63{,}141 \end{aligned}\right\}.$$

Тождества обеспечиваются с погрешностью $|E_1| = 0$ для первого уравнения и $|E_2| = 0$ для второго уравнения в системе. В сравнении с методом Крамера (см. п. 3.2, пример 9) метод Гаусса обеспечил более высокую точность решения системы.

Разработаем блок-схему алгоритма решения поставленной задачи методом Гаусса (рис. 3.3) и программу SLU_GAUSS на языке TURBO PASCAL.

ПРОГРАММА SLU_GAUSS

(постановка задачи см. пример 9, алгоритм см. рис. 3.3)

```
Program SLU_GAUSS;
Uses crt;
Label 1;
Var i, j: Byte;
    D, a11z, a12z, b1z, a22z, b2z, x2, x1, xm, ym, E1, E2: Real;
    A,B: Array[1..5] Of Real;
Begin
Clrscr;
1: Writeln (‘Введите матрицу А совокупности коэффициентов aij.’);
For i:=1 To 2 Do Begin
For i:=1 To 2 Do Begin
Read (’ ’, a[i,j],’ ’);{a11 = -0,364, a12 = 1, a21 = 0,466, a22 = 1}
End; End;
Writeln (’Введите вектор-столбец В совокупности коэффициентов bi.’);
For i:=1 To 2 Do Begin
Read (’ ’,b[i],’ ’); {b1 = 50, b2 = 79,950}
End;
D:=(a[1,1]*a[2,2])-(a[2,1]*a[1,2]);
If D=0 Then Begin
```

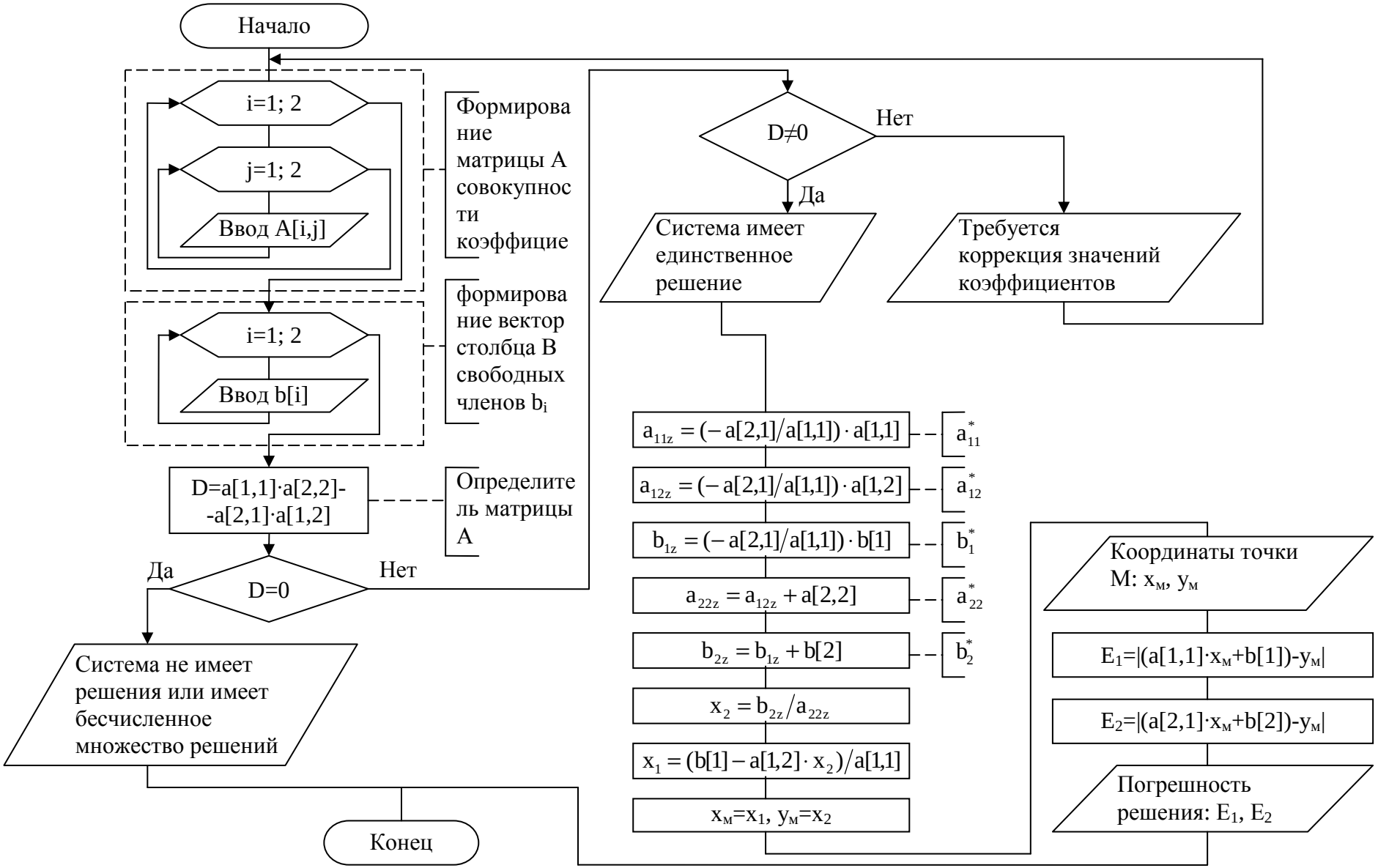


Рис. 3.3. Блок-схема алгоритма метода Гаусса (постановка задачи см. пример 9)

```
Write (’Система не имеет решения или имеет ’);
Writeln (’ бесчисленное множество решений.’);
End;
If D<>0 Then Begin
Writeln (’Система имеет единственное решение.’);
a11z:=(-1*(a[2,1]/a[1,1]))*a[1,1];
a12z:=(-1*(a[2,1]/a[1,1]))*a[1,2];
b1z:=(-1*(a[2,1]/a[1,1]))*b[1];
a22z:=a12z+a[2,2];
b2z:=b1z+b[2];
x2:=b2z/a22z;
x1:=(b[1]-a[1,2]*x2)/a[1,1];
xm:=x1; ym:=x2;
Write (’Координаты точки М: ’);
Writeln (’хм =’, xm:5:3,’,ум =’,ym:5:3,’.’);
Write (’Погрешность расчета: ’);
E1:=Abs((a[1,1]*xm+b[1])-ym);
E2:=Abs((a[2,1]*xm+b[2])-ym);
Writeln(’E1= ’, E1:6:5, ’ ,E2= ’,E2:6:5,’.’);
End Else Begin
Writeln(’Требуется коррекция значений коэффициентов.’);
Goto 1; End;
Readkey;
End.
```

3.4. Метод Зейделя в нахождении технологических параметров системы уравнений с линейными характеристиками

Пусть нам дана система линейных уравнений:

$$\left.\begin{array}{l} a_{11}\cdot x_1 + a_{12}\cdot x_2 + ... + a_{1j}\cdot x_n = b_1 \\ a_{21}\cdot x_1 + a_{22}\cdot x_2 + ... + a_{2j}\cdot x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i1}\cdot x_1 + a_{i2}\cdot x_2 + ... + a_{ij}\cdot x_n = b_n \end{array}\right\}.$$

Необходимо эту систему линейных уравнений разрешить относительно неизвестных x_1, …, x_n, т. е. привести ее к виду:

$$\left.\begin{aligned} x_1 &= \frac{1}{a_{11}} \cdot (b_1 - a_{12} \cdot x_2 - ... - a_{1j} \cdot x_n) \\ x_2 &= \frac{1}{a_{22}} \cdot (b_2 - a_{21} \cdot x_1 - ... - a_{2j} \cdot x_n) \\ &..... \\ x_n &= \frac{1}{a_{ij}} \cdot (b_n - a_{i1} \cdot x_1 - a_{i2} \cdot x_2 - ... - a_{i-1,j-1} \cdot x_{n-1}) \end{aligned}\right\}.$$

После выполнения преобразований устанавливается вектор начального приближения $x^{(0)} = \left\{x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}\right\}$. Этот вектор может быть выбран произвольно, однако необходимо использовать всю имеющуюся информацию о системе уравнений, чтобы $x^{(0)}$ располагался как можно ближе к точному решению системы. При отсутствии такой информации этот вектор можно задать нулевым ($x_1^{(0)} = 0,\ x_2^{(0)} = 0,\ ...,\ x_n^{(0)}$). Определившись с начальным приближением, реализуется первая итерация. В рамках которой находятся уточненные значения неизвестных x_i:

$$x_1^{(1)} = \frac{1}{a_{11}} \cdot (b_1 - a_{12} \cdot x_2^{(0)} - ... - a_{1j} \cdot x_n^{(0)});$$

$$x_2^{(1)} = \frac{1}{a_{22}} \cdot (b_2 - a_{21} \cdot x_1^{(1)} - ... - a_{2j} \cdot x_n^{(0)});$$

.....

$$x_n^{(1)} = \frac{1}{a_{ij}} \cdot (b_n - a_{i1} \cdot x_1^{(1)} - a_{i2} \cdot x_2^{(1)} - ... - a_{i-1,j-1} \cdot x_{n-1}^{(1)})$$

и проверяется выполнение условий:

$$|x_1^{(1)} - x_1^{(0)}| \le E;$$

$$|x_2^{(1)} - x_2^{(0)}| \le E;$$

……

$$|x_n^{(1)} - x_n^{(0)}| \le E,$$

где E – точность вычислений. В случае выполнения этих условий искомое решение системы линейных уравнений будет иметь вид:

$$x_1 = x_1^{(1)};$$

$$x_2 = x_2^{(1)};$$

.....

$$x_n = x_n^{(1)}.$$

В противном случае, необходимо продолжить процесс уточнения значений вектор – столбца неизвестных x_i реализаций последующих итераций.

Предположим, что условия не выполнились, следовательно, реализуем вторую итерацию. В рамках этой итерации в качестве начальных выбирается вектор $x^{(1)} = \left\{x_1^{(1)}, x_2^{(1)}, ..., x_n^{(1)}\right\}$, полученный с предыдущей итерации, и находятся уточненные значения неизвестных x_i:

$$x_1^{(2)} = \frac{1}{a_{11}} \cdot (b_1 - a_{12} \cdot x_2^{(1)} - ... - a_{1j} \cdot x_n^{(1)});$$

$$x_2^{(2)} = \frac{1}{a_{22}} \cdot (b_2 - a_{21} \cdot x_1^{(2)} - ... - a_{2j} \cdot x_n^{(1)});$$

.....

$$x_n^{(2)} = \frac{1}{a_{ij}} \cdot (b_n - a_{i1} \cdot x_1^{(2)} - a_{i2} \cdot x_2^{(2)} - ... - a_{i-1,j-1} \cdot x_{n-1}^{(2)})$$

и вновь проверяется выполнение условий:

$$| x_1^{(2)} - x_1^{(1)} | \le E;$$

$$| x_2^{(2)} - x_2^{(1)} | \le E;$$

……

$$| x_n^{(2)} - x_n^{(1)} | \le E.$$

Предположим, что условия выполняются, тогда искомое решение системы будет определено значениями x_i вектора x, полученных в рамках реализации второй итерации:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ ... \\ x_n^{(2)} \end{bmatrix}.$$

В общем случае, каждое К приближение для К итерации определяют по формулам:

$$x_1^{(к)} = \frac{1}{a_{11}} \cdot (b_1 - a_{12} \cdot x_2^{(к-1)} - ... - a_{1j} \cdot x_n^{(к-1)});$$

$$x_2^{(к)} = \frac{1}{a_{22}} \cdot (b_2 - a_{21} \cdot x_1^{(к)} - ... - a_{2j} \cdot x_n^{(к-1)});$$

.....

$$x_n^{(к)} = \frac{1}{a_{ij}} \cdot (b_n - a_{i1} \cdot x_1^{(к)} - a_{i2} \cdot x_2^{(к)} - ... - a_{i-1,j-1} \cdot x_{n-1}^{(к)}).$$

Итерационный процесс продолжается до тех пор, пока все $x_i^{(к)}$ не станут достаточно близкими к $x_i^{(к-1)}$. Итерации прекращаются при выполнении условий:

$$|x_1^{(к)} - x_1^{(к-1)}| \le E;$$
$$|x_2^{(к)} - x_2^{(к-1)}| \le E;$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$|x_n^{(к)} - x_n^{(к-1)}| \le E.$$

Метод Зейделя является одним из широко применяемых и легко программируемых в группе итерационных методов.

Осуществим практическую реализацию метода Зейделя по постановке задачи (см. пример 9). Разработаем блок-схему алгоритма (рис. 3.4) и программу SLU_ZEIDEL на языке TURBO PASCAL для решения системы:

$$\left.\begin{aligned} -0{,}364 \cdot x_1 + x_2 &= 50 \\ 0{,}466 \cdot x_1 + x_2 &= 79{,}950 \end{aligned}\right\},$$

которая была получена в результате ранее выполненных преобразований (см. п. 3.2, пример 9), где x_1 соответствует координате $x_м$, а x_2 координате $y_м$ положения точки М (см. рис. 3.2).

ПРОГРАММА SLU_ZEIDEL

(постановка задачи см. пример 9, алгоритм см. рис. 3.4)

```
Program SLU_ZEIDEL;
Uses crt;
Label 1;
Var i, j: Byte;
    D, E, E1, E2, C1, C2, x1, x2, xm, ym, Del1,Del2: Real;
    A,B Array[1..5] Of Real;
Begin
Clrscr;
1: Writeln (’Введите матрицу А совокупности элементов a_ij’);
For i:=1 To 2 Do Begin
For j:=1 To 2 Do Begin
Read (’ ’,a[i,j],’ ’); {a11 = -0,364, a12 = 1, a21 = 0,466, a22 = 1}
End; End;
Writeln (’ Введите вектор-столбец В совокупности коэффициентов b_i. ’);
For i:=1 To 2 Do Begin
Read (’ ’,b[i,],’ ’); {b1 = 50, b2 = 79,950}
End;
D:=(a[1,1]*a[2,2])*-(a[2,1]*a[1,2]);
If D=0 Then Begin
Write (’Система не имеет решения или имеет’);
Writeln (’ бесчисленное множество решений.’);
```

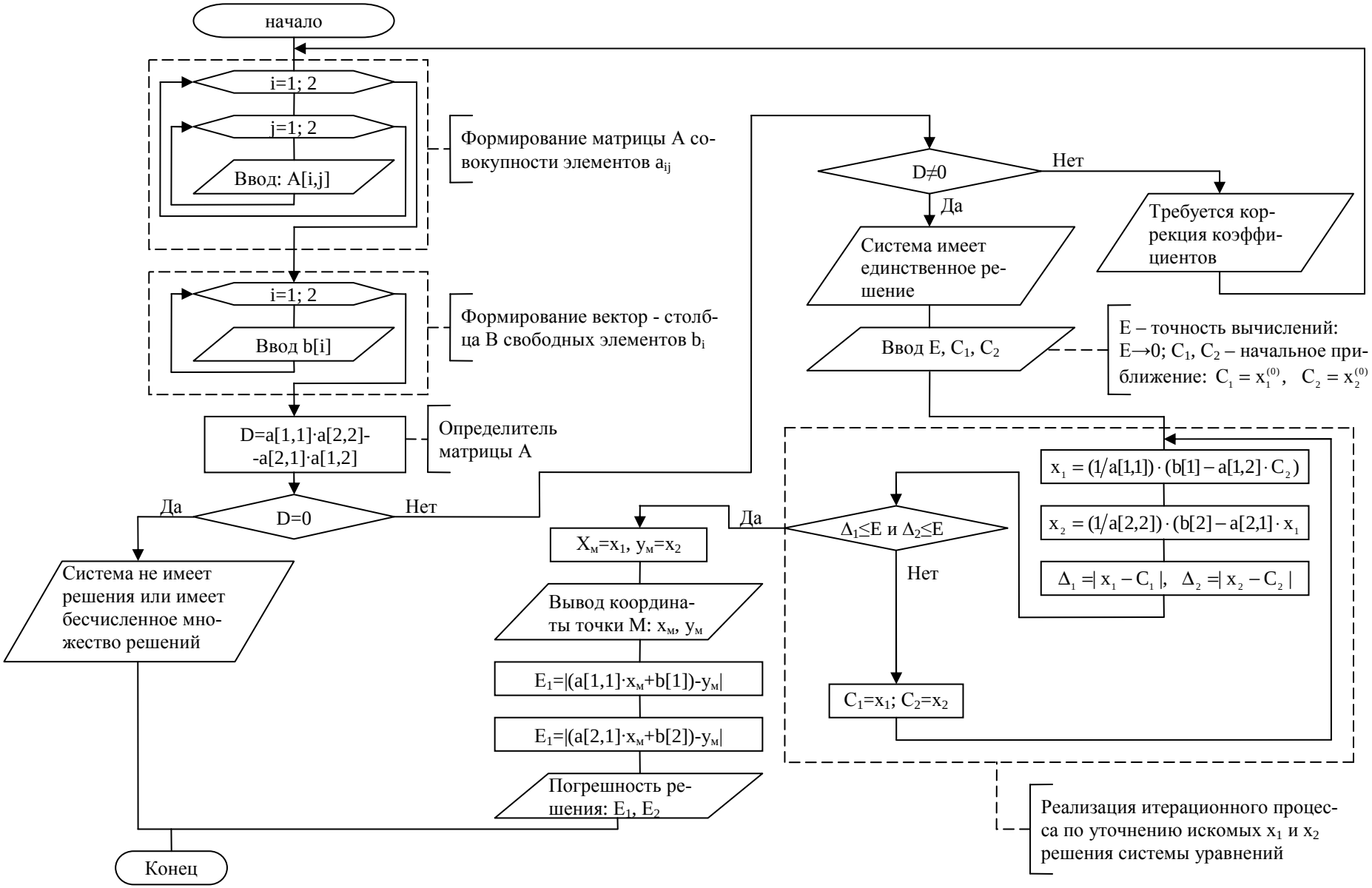


Рис. 3.4. Блок-схема алгоритма метода Зейделя (постановка задачи см. пример 9)

```
End;
If D<>0 Then Begin
Writeln (’Система имеет единственное решение.’);
Write (’Задайте точность расчета E:’);
Readln (E); {E=0,001}
Write(’Введите начальное приближение  x10:’);
Readln(C1); {x10≈xm=36 (см. рис. 3.2)}
Write (’Введите начальное приближение x20:’);
Readln (C2); {x20≈ym=63 (см. рис. 3.2)}
Repeat
x1:=(1/a[1,1])*(b[1]-a[1,2]*C2);
x1:=(1/a[2,2])*(b[2]-a[2,1]*x1);
Del1:=Abs(x1-C1);
Del2:=Abs(x2-C1);
C1:=x1;
C2:=x2;
Until (Del1<=E) And (Del2<=E);
xm:=x1;
ym:=x2;
Write (’Координаты точки M:’);
Writeln (’хм = ’,xm:5:3,’, ум = ’,ym:5:3’.’);
Write(’Погрешность расчета:’);
E1:=Abs((a[1,1]*xm+b[1])-ym);
E2:=Abs((a[2,1]*xm+b[2])-ym);
Writeln(’E1 = ’,E1:6:5,’, E2 = ’,E2:6:5,’.’);
End Else Begin
Writeln (’Требуется коррекция коэффициентов.’);
Goto 1; End;
Readkey;
End.
```

4. АППРОКСИМАЦИЯ ТЕХНОЛОГИЧЕСКИХ ПАРАМЕТРОВ В ЗАДАЧАХ МАШИНОСТРОЕНИЯ

4.1. Основные понятия

Аппроксимация – это замена исходной зависимости $y = f(x)$, устанавливающей связь входного параметра *x* с выходным параметром y, которая может быть задана в явном виде через формулу или в виде таблицы со множеством значений $\{x_i, y_i\}$, приближенной к ней вида $y = \varphi(x)$ с заданной точностью. Зависимость $y = \varphi(x)$, заменяющая исходную $y = f(x)$, называется аппроксимационной. К аппроксимации прибегают в случаях, например, когда исходная зависимость $y = f(x)$ представлена в виде сложной формулы, в записи которой используются нестандартные функции, нахождение значений которых требует больших затрат времени или вообще исключается, исходя из возможностей языка программирования, при решении задачи с помощью ЭВМ, а также, когда зависимость $y = f(x)$ является точечной и задана в виде таблицы со множеством значений $\{x_i, y_i\}$, при этом необходимо найти приближенное значение выходного параметра *y* при x^*, не совпадающем по своему значению не с одним из x_i таблицы, т. е. $x_i < x^* < x_{i+1}$. В случае, когда $y = f(x)$ выражается формулой, для аппроксимации этой формулы могут использоваться различные ряды, например, показательные и степенные или многочленые, например, ортогональные, а при табличном представлении $y = f(x)$ для аппроксимации используется интерполирование с помощью различных интерполяционных многочленов, например, линейных при линейном интерполировании, параболических при параболической (квадратичной) интерполяции, многочленов Лагранжа, многочленов Ньютона. Кроме этого, может осуществляться интерполирование кубическими сплайн-функциями. Если приближение $y = \varphi(x)$ к $y = f(x)$ строится на заданном множестве точек x_i, то аппроксимация называется точечной. Независимо от вида интерполяционного многочлена при точечной аппроксимации интерполяция может быть глобальной и локальной.

При глобальной интерполяции интерполяционный многочлен $y = \varphi(x)$ используется для точечной аппроксимации $y = f(x)$ на всем рассматриваемом интервале изменение аргумента x (рис. 4.1), а при локальной интерполяции интерполяционный многочлен $y = \varphi(x)$ используется для точечной аппроксимации $y = f(x)$ на некотором интервале изменения аргумента x от x_i до x_{i+1}, кроме того интерполяционных многочленов может быть несколько (рис. 4.2). При

прочих равных условиях меньшую погрешность аппроксимации обеспечивает локальная интерполяция. Интерполяционный многочлен $y = \varphi(x)$ должен задаваться таким, чтобы обеспечивалась близость его расположения относительно исходной зависимости $y = f(x)$.

Близость расположения $y = \varphi(x)$ относительно $y = f(x)$ оценивается среднеквадратичным приближением. Мерой отклонения многочлена $y = \varphi(x)$ от заданной зависимости $y = f(x)$ на множестве точек $\{x_i, y_i\}$ при среднеквадратичном приближении является величина *S*:

$$S = \sum_{i=1}^{n} (\varphi(x_i) - f(x_i))^2, \qquad (9)$$

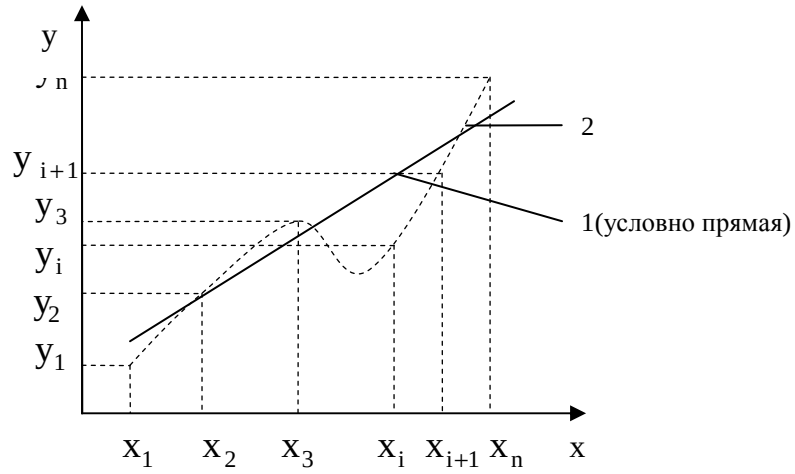


Рис. 4.1. Глобальная интерполяция: 1 – аппроксимационная зависимость, представленная в виде интерполяционного многочлена $y = \varphi(x)$; 2 – исходная зависимость $y = f(x)$, представленная в виде точечной

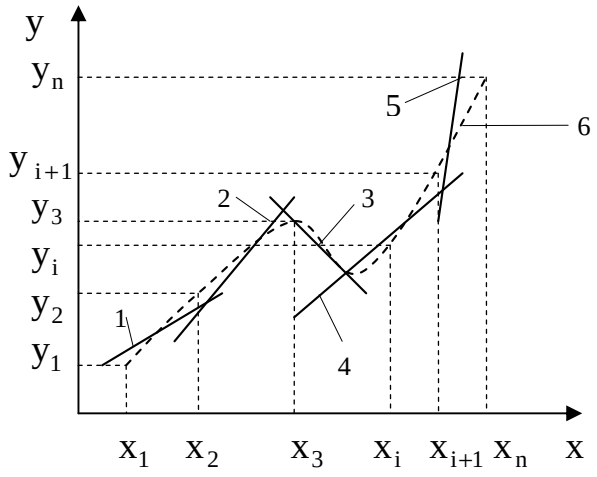


Рис. 4.2. Локальная интерполяция: 1 – 5 – аппроксимационные зависимости, представленные в виде интерполяционных многочленов, соответственно, $y = \varphi_1(x)$, $y = \varphi_2(x)$, $y = \varphi_3(x)$, $y = \varphi_4(x)$, $y = \varphi_5(x)$; 6 – исходная зависимость $y = f(x)$, представленная в виде точечной

т. е. находится сумма квадратов разностей между значениями многочлена и исходной зависимости в данных точках от i = 1 до i = n. Значение S должно стремиться к своему минимуму, т. е. к нулю. Для того, чтобы найти минимальное значение S необходимо задать несколько интерполяционных многочленов на рассматриваемой области значений аргумента x, после чего для каждого случая определить S_1, S_2, S_3, …, S_n. Среди найденных S_i методом сравнения найти S с наименьшим значением и окончательно выбрать тот интерполяционный многочлен, при котором и получено минимальное значение S. Все вышесказанное можно пояснить с помощью рис. 4.3. Предположим, опираясь на рис. 4.3, что наименьшее значение S обеспечивается для второго варианта аппроксимационной зависимости, представленной в виде интерполяционного многочлена $y = \varphi_2(x)$ (кривая 3), т. е. $S_2^{(3)} \approx 0$. Исходя из этого, зависимость $y = \varphi_2(x)$ можно выбрать в качестве искомой. Используя эту аппроксимационную зависимость можно определить приближенные значения исходной зависимости $y = f(x)$, заданной в виде точечной, с наименьшей погрешностью при непрерывном изменении x от x_1 до x_3.

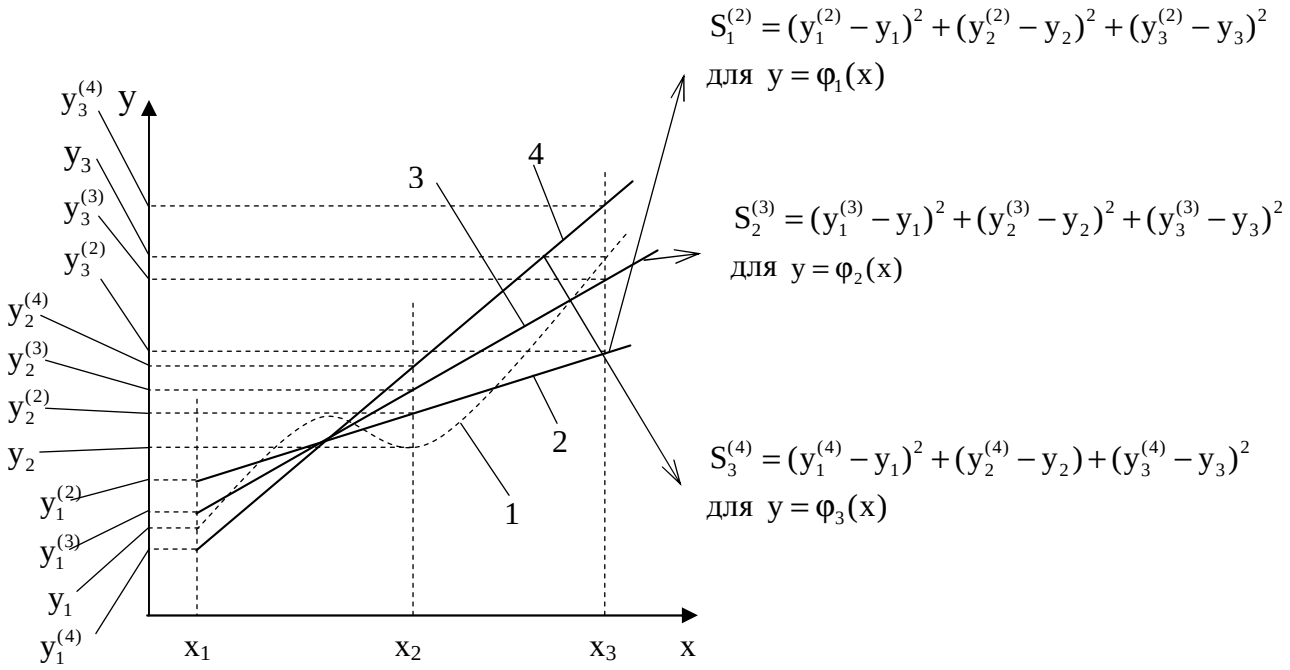


Рис. 4.3. Расчетная схема к нахождению минимального значения S и установления аппроксимационной зависимости: 1 – исходная зависимость, представленная в виде точечной $y = f(x)$: $x = \{x_1, x_2, x_3\}$, $y = \{y_1, y_2, y_3\}$; 2 – первый вариант аппроксимационной зависимости, представленной в виде интерполяционного многочлена $y = \varphi_1(x)$; 3 – второй вариант аппроксимационной зависимости, представленной в виде интерполяционного многочлена $y = \varphi_2(x)$; 4 – третий вариант аппроксимационной зависимости, представленной в виде интерполяционного многочлена $y = \varphi_3(x)$ (реализуется глобальная интерполяция, графики интерполяционных многочленов 2,3,4 условно показаны прямыми)

4.2. Использование рядов и многочленов для аппроксимации технологических параметров

В ряде случаев при решении задач машиностроения возникает необходимость вычисления значений различных тригонометрических, показательных, логарифмических функций. В том случае, если значения этих функций нужно вычислить на ЭВМ, используют аппроксимацию таких функций ортогональными многочленами, степенными и показательными рядами. Рассмотрим несколько примеров аппроксимации функций ортогональными многочленами (таблица 4.1), степенными и показательными рядами (таблица 4.2).

Таблица 4.1

Аппроксимация функций ортогональными многочленами

Исходная зависимость $y = f(x)$	Аппроксимационная зависимость $y = \varphi(x)$	Наибольшая абсолютная погрешность
1	2	3
e^{-x} $(0 \le x \le \ln 2)$	$1 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_4 \cdot x^4$ $(a_1 = -0{,}9998684,\ a_2 = 0{,}4982926,$ $a_3 = -0{,}1595332,\ a_4 = 0{,}0293641)$	$3 \cdot 10^{-5}$
10^x $(0 \le x \le 1)$	$\left(1 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_4 \cdot x^4\right)^2$ $(a_1 = 1{,}1499196,\ a_2 = 0{,}6774323,$ $a_3 = 0{,}2080030,\ a_4 = 0{,}1268089)$	$7 \cdot 10^{-4}$
$\frac{\sin(x)}{x}$ $\left(0 \le x \le \frac{\pi}{2}\right)$	$1 + a_2 \cdot x^2 + a_4 \cdot x^4$ $(a_2 = -0{,}16605,\ a_4 = 0{,}00761)$	$2 \cdot 10^{-4}$
$\cos(x)$ $\left(0 \le x \le \frac{\pi}{2}\right)$	$1 + a_2 \cdot x^2 + a_4 \cdot x^4$ $(a_2 = -0{,}49670,\ a_4 = 0{,}03705)$	$9 \cdot 10^{-4}$
$\frac{\operatorname{tg}(x)}{x}$ $\left(0 \le x \le \frac{\pi}{4}\right)$	$1 + a_2 \cdot x^2 + a_4 \cdot x^4$ $(a_2 = 0{,}31755,\ a_4 = 0{,}20330)$	$1 \cdot 10^{-3}$
$\arcsin(x)\ (0 \le x \le 1)$	$\frac{\pi}{2} - \sqrt{1-x} \cdot \left(a_0 + a_1 \cdot x + \ldots + a_3 \cdot x^3\right)$ $(a_0 = 1{,}5707288,\ a_1 = -0{,}2121144,$ $a_2 = 0{,}0742610,\ a_3 = -0{,}0187293)$	$5 \cdot 10^{-5}$

Окончание табл. 4.1

1	2	3
$\lg(x)$ $\left(\frac{1}{\sqrt{10}} \le x \le \sqrt{10}\right)$	$a_1 \cdot t + a_3 \cdot t^3$ $(t = \frac{x-1}{x+1},\ a_1 = 0{,}86304,\ a_1 = 0{,}36415)$	$6 \cdot 10^{-4}$
$\operatorname{arctg}(x)\ (-1 \le x \le 1)$	$\frac{x}{1 + 0{,}28 \cdot x^2}$	$5 \cdot 10^{-3}$

Таблица 4.2.

Аппроксимация функций степенными и показательными рядами, соотношения между функциями

Исходная зависимость $y = f(x)$	Аппроксимационная зависимость $y = \varphi(x)$	Примечание
1	2	3
$\sin(x)$	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$	$\lvert x \rvert < \infty$
$\cos(x)$	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$	$\lvert x \rvert < \infty$
e^x	$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$	$\lvert x \rvert < \infty$
$\operatorname{sh}(x)$	$x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots$	$\lvert x \rvert < \infty$
$\operatorname{ch}(x)$	$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots$	$\lvert x \rvert < \infty$
$\ln(1 + x)$	$x - \frac{x^2}{2!} + \frac{x^3}{3!} - \frac{x^4}{4!} + \frac{x^5}{5!} - \ldots$	$\lvert x \rvert < 1$
$\arcsin(x)$	$x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{x^5}{5} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{x^7}{7} + \ldots$	$\lvert x \rvert < 1$
$\operatorname{arsh}(x)$	$x - \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{x^5}{5} - \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdot \frac{x^7}{7} + \ldots$	$\lvert x \rvert < 1$
$\operatorname{arctg}(x)$	$x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$	$\lvert x \rvert < 1$
$\operatorname{arth}(x)$	$x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots$	$\lvert x \rvert < 1$
$\operatorname{tg}(x)$	$x + \frac{1}{3} \cdot x^3 + \frac{2}{15} \cdot x^5 + \frac{17}{315} \cdot x^7 + \ldots$	$\lvert x \rvert < \frac{\pi}{2}$

Окончание табл. 4.2

1	2	3
$\mathrm{ctg}(x)$	$\frac{1}{x}-\frac{1}{3}\cdot x-\frac{1}{45}\cdot x^3-\frac{2}{945}\cdot x^5-\ldots$	$\lvert x\rvert<\pi$
$\mathrm{ch}(x)$	$\frac{e^x+e^{-x}}{2}$	–
$\mathrm{sh}(x)$	$\frac{e^x-e^{-x}}{2}$	–
e^x	$\mathrm{ch}(x)+\mathrm{sh}(x)$	–
e^{-x}	$\mathrm{ch}(x)-\mathrm{sh}(x)$	–
$\arccos(x)$	$-i\cdot\ln\left(x+i\cdot\sqrt{1-x^2}\right)$	–
$\arcsin(x)$	$-i\cdot\ln\left(i\cdot x+\sqrt{1-x^2}\right)$	–
$\mathrm{arctg}(x)$	$-\frac{i}{2}\cdot\ln\left(\frac{1+i\cdot x}{1-i\cdot x}\right)$	–
$\mathrm{arcctg}(x)$	$-\frac{i}{2}\cdot\ln\left(\frac{i\cdot x-1}{i\cdot x+1}\right)$	–
$\mathrm{th}(x)$	$\frac{\mathrm{sh}(x)}{\mathrm{ch}(x)}$	–
$\mathrm{cth}(x)$	$\frac{\mathrm{ch}(x)}{\mathrm{sh}(x)}$	–

Пример 10. Определить технологические размеры угловой впадины на поверхности заготовки с помощью мерного ролика в соответствии с расчетной схемой, представленной на рис. 4.4.

РЕШЕНИЕ. Опираясь на расчетную схему (см. рис. 4.4), искомые значения глубины угловой впадины Б и расстояния от дна угловой впадины до выступающей части ролика В можно рассчитать по следующим зависимостям:

$$Б = A\cdot\mathrm{ctg}(\alpha);$$

$$B = 0{,}5\cdot D\cdot\left(1+\mathrm{ctg}\left(\frac{\alpha}{2}\right)\right).$$

Учитывая то, что вычислительный процесс предлагается автоматизировать и искомые Б и В определить с помощью программы на языке TURBO PASCAL, необходимо осуществить аппроксимацию функции $\mathrm{ctg}(\alpha)$ (см. табл. 4.2), так как эта функция не зарезервирована выше обозначенным языком программирования. В результате, расчетные зависимости по определению Б и В можно представить следующим образом:

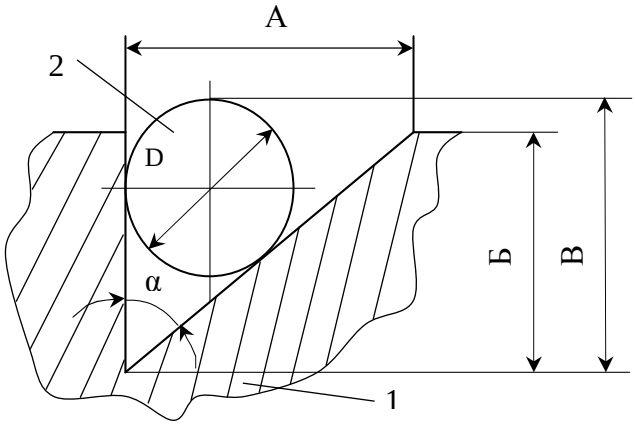


Рис. 4.4. Расчетная схема: 1 – заготовка; 2 – мерный ролик; А – ширина угловой впадины, мм (известная величина); D – диаметр мерного ролика, мм (известная величина); α – угол наклона боковой поверхности угловой впадины, град. (известная величина); Б – глубина угловой впадины, мм (искомая величина); В – расстояние от дна угловой впадины до выступающей части ролика, мм (искомая величина)

$$Б^* = A \cdot \left(\frac{1}{\alpha} - \frac{1}{3} \cdot \alpha - \frac{1}{45} \cdot \alpha^3 - \frac{2}{945} \cdot \alpha^5 \right);$$

$$B^* = 0{,}5 \cdot D \cdot \left(1 + \frac{1}{\alpha/2} - \frac{1}{3} \cdot \left(\alpha/2\right) - \frac{1}{45} \cdot \left(\alpha/2\right)^3 - \frac{2}{945} \cdot \left(\alpha/2\right)^5 \right),$$

где α в рад.

Пусть А = 20 мм, D = 15 мм, а $\alpha = 45^\circ$, тогда при расчете Б и В по зависимостям без использования аппроксимации функции $\mathrm{ctg}(\alpha)$ получим:

$$Б = 20 \cdot \mathrm{ctg}\left(45^\circ\right) = 20 \text{ мм};$$

$$B = 0{,}5 \cdot 15 \cdot \left(1 + \mathrm{ctg}\left(45^\circ/2 \right) \right) = 25{,}606 \text{ мм}.$$

При расчете Б и В по зависимостям с использованием аппроксимации функции $\mathrm{ctg}(\alpha)$ получим:

$$Б^* = 20 \cdot \left(\frac{1}{0{,}785} - \frac{1}{3} \cdot 0{,}785 - \frac{1}{45} \cdot 0{,}785^3 - \frac{2}{945} \cdot 0{,}785^5 \right) =$$

$$= 20 \cdot \left(1{,}274 - 0{,}262 - 0{,}011 - 0{,}0006\right) = 20{,}008 \text{ мм};$$

$$B^* = 0{,}5 \cdot 15 \cdot \left(1 + \frac{1}{0{,}393} - \frac{1}{3} \cdot 0{,}393 - \frac{1}{45} \cdot 0{,}393^3 - \frac{2}{945} \cdot 0{,}393^5 \right) =$$

$$= 7{,}5 \cdot \left(1 + 2{,}545 - 0{,}131 - 0{,}0013 - 0{,}000019\right) = 25{,}595 \text{ мм}.$$

Таким образом, относительная погрешность аппроксимации при нахождении Б составит равной:

$$\xi_{Б} = \frac{Б - Б^{*}}{Б} \cdot 100\%;$$

$$\xi_{Б} = \frac{20 - 20{,}008}{20} \cdot 100\% = -0{,}04\%,$$

а при нахождении В:

$$\xi_{В} = \frac{В - В^{*}}{В} \cdot 100\%;$$

$$\xi_{В} = \frac{25{,}606 - 25{,}595}{25{,}606} \cdot 100\% = 0{,}043\%.$$

В соответствии с постановкой задачи (см. пример 10) разработаем алгоритм (рис. 4.5) и программу TEX_RAZMER ее решения на языке TURBO PASCAL.

Рис. 4.5. Блок-схема алгоритма нахождения технологических размеров угловой впадины на поверхности заготовки с помощью мерного ролика (постановка задачи см. пример 10, расчетная схема см. рис. 4.4)

Программа TEX_RAZMER
(постановка задачи см. пример 10, алгоритм см. рис. 4.5)

```
Program TEX_RAZMER;
Uses crt;
Var A, D, AL, G, R, AL1: Real;
Begin
Clrscr;
Write('Введите значение ширины угловой впадины А в мм:');
Readln(A);
Write ('Введите значение диаметра мерного ролика');
Writeln (' D в мм (D < A):');
Readln(D);
Write('Введите значение угла наклона боковой');
Write (' поверхности угловой впадины Альфа в град.:');
Writeln (' (0 < Альфа < 90):');
Readln(AL1);
AL:=(AL1*pi)/180; {Перевод градусов в радианы}
G:=A*((1/AL)-((1/3*AL)-((1/45)*AL*Sqr(AL))-((2/945)*Sqr(AL)*Sqr(AL)*AL));
{Расчет параметра Б*}
R:=0.5*D*(1+(1/(AL/2))-((1/3)*(AL/2))-((1/45)*(AL/2)*Sqr(AL/2))-
-((2/945)*Sqr(AL/2)*Sqr(AL/2)*(AL/2))); {Расчет параметра В*}
Writeln('При А = ',A:5:3,' мм, D = ',D:5:3,' мм и Альфа = ',AL1:5:1,' град – Б*=
',G:5:3,' мм и В* = ', R:5:3,' мм.');
Readkey;
End.
```

4.3. Использование интерполирования для нахождения выходных параметров технологических систем

Интерполирование осуществляется в тех случаях, когда, например, необходимо найти приближенное значение выходного параметра y^* технологической системы по известной зависимости $y = f(x)$, являющейся точечной и заданной в виде таблицы со множеством значений $\{x_i, y_i\}$, полученных, например, экспериментально. Причем, речь идет о нахождении y^* при таком значении x^*, которое не совпадает ни с одним из значений x_i таблицы.

Из ранее обозначенных видов интерполирования (см. п. 4.1) широко применяется интерполирование линейными многочленами, параболическими многочленами и многочленами Ньютона.

При линейном интерполировании искомое значение выходного параметра y^* технологической системы определяется из уравнения прямой (рис. 4.6):

$$y^* = a_i \cdot x^* + b_i, \qquad (10)$$

где a_i – угловой коэффициент прямой, проходящей через две соседние точки точечной зависимости $y = f(x)$ с координатами (x_i, y_i) и (x_{i-1}, y_{i-1}) с учетом выполнения условия $x_{i-1} < x^* < x_i$:

$$a_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}};$$

b_i – коэффициент смещения прямой:

$$b_i = y_{i-1} - a_i \cdot x_{i-1}.$$

Таким образом, при реализации линейной интерполяции сначала необходимо определить интервал, в который попадает заданное значение аргумента x^*, а затем подставив его в формулу (10) уравнения прямой, при известных a_i и b_i найти приближенное значение выходного параметра y^*.

При параболическом интерполировании искомое значение выходного параметра y^* определяется из уравнения параболы (рис. 4.7):

$$y^* = a_i \cdot x^{*2} + b_i \cdot x^* + c_i, \qquad (11)$$

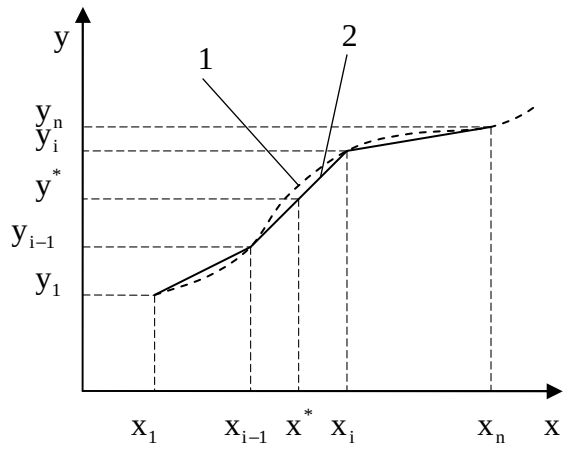


Рис. 4.6. Расчетная схема к нахождению приближенного значения выходного параметра y^* при линейном локальном интерполировании: 1 – исходная зависимость $y = f(x)$, представленная в виде точечной; 2 – аппроксимационная зависимость $y = \varphi(x)$, описываемая уравнением прямой (линейный многочлен) на заданном отрезке от x_{i-1} до x_i, содержащем x^*

где x^* должно находится в интервале $x_{i-1} \le x^* \le x_{i+1}$, а коэффициенты a_i, b_i, c_i определяются из условия прохождения параболы через точки (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}) точечной зависимости $y = f(x)$:

$$\left.\begin{aligned} a_i \cdot x_{i-1}^2 + b_i \cdot x_{i-1} + c_i &= y_{i-1} \\ a_i \cdot x_i^2 + b_i \cdot x_i + c_i &= y_i \\ a_i \cdot x_{i+1}^2 + b_i \cdot x_{i+1} + c_i &= y_{i+1} \end{aligned}\right\}.$$

Решив систему относительно a_i, b_i, c_i и подставив эти коэффициенты в уравнение (11), в итоге получим:

$$y^* = A \cdot x^{*2} + B \cdot x^* + C, \qquad (12)$$

где

$$A = \frac{(y_{i+1} - y_i)\cdot(x_{i-1} - x_i) - (y_{i-1} - y_i)\cdot(x_{i+1} - x_i)}{(x_{i+1} - x_i)\cdot(x_{i-1} - x_i)\cdot(x_{i+1} - x_{i-1})};$$

$$B = \frac{(y_{i-1} - y_i)\cdot(x_{i+1}^2 - x_i^2) - (y_{i+1} - y_i)\cdot(x_{i-1}^2 - x_i^2)}{(x_{i-1} - x_i)\cdot(x_{i+1} - x_i)\cdot(x_{i+1} - x_{i-1})};$$

$$C = y_i - A \cdot x_i^2 - B \cdot x_i.$$

Таким образом, при реализации параболической интерполяции сначала необходимо определить интервал, в который попадает заданное значение аргумента (входного параметра) x^*, а затем, подставив его в формулу (12) уравнения параболы, при известных A, B и C найти приближенное значение y^*.

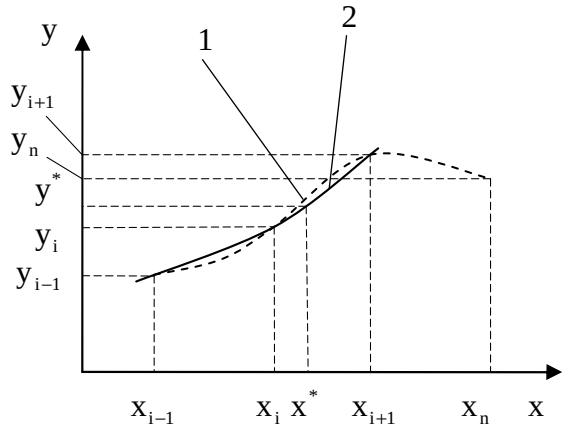


Рис. 4.7. Расчетная схема к нахождению приближенного значения выходного параметра y^* при параболическом локальном интерполировании: 1 – исходная зависимость $y = f(x)$, представленная в виде точечной; 2 – аппроксимационная зависимость $y = \varphi(x)$, описываемая уравнением параболы (параболический многочлен) на заданном отрезке от x_{i-1} до x_{i+1}, содержащем x^*

При интерполировании многочленом Ньютона значение выходного параметра y^* определяется из уравнения многочлена Ньютона (рис. 4.8):

$$y^* = y_i + t \cdot \Delta^{(1)} y_i + \frac{t \cdot (t-1)}{2!} \cdot \Delta^{(2)} y_i + \frac{t \cdot (t-1) \cdot (t-2) \cdot (t-3) \cdot \ldots \cdot (t-n+1)}{n!} \cdot \Delta^{(n)} y_i, \quad (13)$$

где $t = (x^* - x_i)/h$, при $x_{i-1} \le x^* \le x_i$ и шаге $h = x_i - x_{i-1}$ – величина неизменная $(x_2 - x_1 = x_3 - x_2 = \ldots = x_i - x_{i-1})$;

$\Delta^{(1)} y_i, \Delta^{(2)} y_i, \ldots, \Delta^{(n)} y_i$ – разности соответственно первого, второго и n-го порядков: $\Delta^{(1)} y_i = y_i - y_{i-1}$, $\Delta^{(2)} y_i = \Delta^{(1)} y_{i+1} - \Delta^{(1)} y_i, \ldots$, $\Delta^{(n)} y_i = \Delta^{(n-1)} y_{i+1} - \Delta^{(n-1)} y_i$; n – число точек точечной зависимости $y = f(x)$.

Таким образом, при реализации интерполяции многочленом Ньютона сначала необходимо определить интервал, в которой попадает заданное значение аргумента x^*, а затем, подставив его в формулу (13) уравнения многочлена Ньютона, при известных разностях соответствующих порядков найти приближенное значение y^*.

Осуществим практическую реализацию точечной аппроксимации при локальном линейном интерполировании по постановке задачи (пример 11).

Пример 11. Известна закономерность изменения шероховатости обрабатываемой поверхности заготовки при сверлении R_a от скорости резания V, полученная в результате проведенных экспериментальных исследований и представленная в виде таблицы:

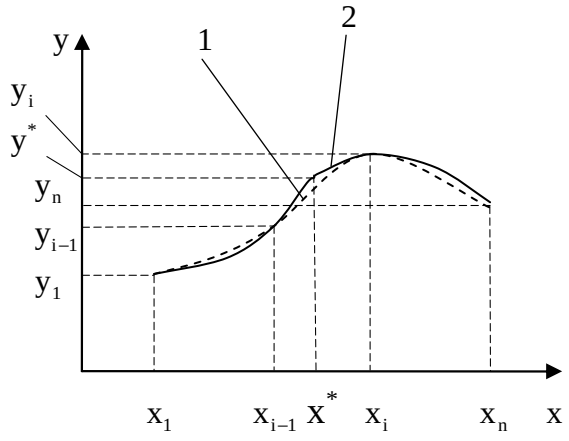


Рис. 4.8. Расчетная схема к нахождению приближенного значения выходного параметра y^* при локальном интерполировании многочленом Ньютона: 1 – исходная зависимость $y = f(x)$, представленная в виде точечной; 2 – аппроксимационная зависимость $y = \varphi(x)$, описываемая многочленом Ньютона (кривая) на заданном отрезке от x_{i-1} до x_i, содержащем x^*

№ опыта	1	2	3	4	5
$V, \frac{м}{мин}$	12	14	16	18	20
R_a, мкм	6,23	6,31	6,45	6,52	6,60

где R_a – среднее арифметическое отклонение микронеровностей профиля, мкм; V – скорость резания при сверлении, м/мин; диаметр сверла d: d = 12 мм; подача сверла S: $S = 0{,}10$ мм/об. Также известна исходная зависимость $R_a = f(V)$:

$$R_a = 6{,}36 \cdot d^{0{,}25} \cdot V^{0{,}12} \cdot S^{0{,}41},$$

полученная по результатам теоретико-экспериментальных исследований. Адекватность этой зависимости доказана.

Необходимо определить приближенное значение R_a^* при $V^* = 15$ м/мин, реализовав локальную линейную интерполяцию точечной зависимости, представленной в виде таблицы, и установить величину погрешности аппроксимации в нахождении искомого значения R_a^*.

РЕШЕНИЕ. Из табличных значений зависимости $R_a = f(V)$ видно, что $V^* = 15$ м/мин попадает в интервал скорости резания при сверлении от $V_2 = 14$ м/мин до $V_3 = 16$ м/мин, следовательно, воспользовавшись уравнением (10), запишем:

$$R_a^* = a_3 \cdot V^* + b_3,$$

где a_3 – угловой коэффициент прямой, проходящей через две соседние точки точечной зависимости $R_a = f(V)$ с координатами (V_3, R_{a_3}) и (V_2, R_{a_2}) (рис. 4.9):

$$a_3 = \frac{R_{a_3} - R_{a_2}}{V_3 - V_2};$$

$$a_3 = \frac{6{,}45 - 6{,}31}{16 - 14} = 0{,}07;$$

b_3 – коэффициент смещения прямой, проходящей через те же точки (V_3, R_{a_3}) и (V_2, R_{a_2}):

$$b_3 = R_{a_2} - a_3 \cdot V_2;$$

$$b_3 = 6{,}31 - 0{,}07 \cdot 14 = 5{,}33.$$

В результате:

$$R_a^* = 0{,}07 \cdot V^* + 5{,}33$$

и при $V^* = 15$ м/мин:

$$R_a^* = 0{,}07 \cdot 15 + 5{,}33 = 6{,}380 \text{ мкм}.$$

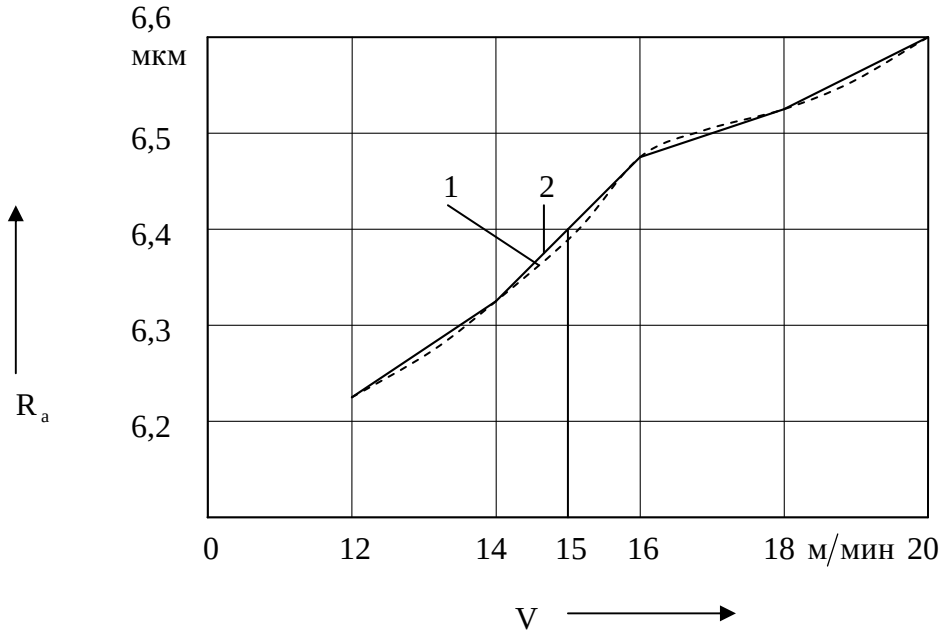


Рис. 4.9. Зависимость изменения шероховатости R_a от скорости резания V при сверлении: 1 – исходная зависимость $R_a = f(V)$, представленная в виде точечной; 2 – аппроксимационная зависимость $R_a = \varphi(V)$, описываемая уравнением прямой на отрезке от 14 до 16 м/мин; d = 12 мм; S = 0,10 мм/об

Определим наиболее близкое к точному значение R_a при $V^* = 15$ м/мин:

$$R_a = 6{,}36 \cdot 12^{0{,}25} \cdot 15^{0{,}12} \cdot 0{,}10^{0{,}41} = 6{,}372 \text{ мкм}.$$

Найдем абсолютную и относительную погрешность точечной аппроксимации при локальном линейном интерполировании:

$$R_{a_\Delta} = R_a - R_a^*;$$

$$R_{a_\Delta} = 6{,}372 - 6{,}380 = -0{,}008 \text{ мкм};$$

$$\xi_{R_a} = \frac{R_a - R_a^*}{R_a} \cdot 100;$$

$$\xi_{R_a} = \frac{6{,}372 - 6{,}380}{6{,}372} \cdot 100 = -0{,}13\,\%.$$

Независимо от вида интерполирования уменьшить погрешность аппроксимации можно за счет уменьшения величины шага изменения входного параметра. В нашем случае скорости резания при сверлении V, т. е. $h_V \to h_{V_{min}}$, где $h_{V_{min}} \approx 0$.

Для наглядности определим R_a^* при параболической интерполяции и интерполяции многочленом Ньютона.

При параболической интерполяции, опираясь на зависимость (11):

$$R_a^* = A \cdot V^{*2} + B \cdot V^* + C,$$

где

$$A = \frac{(R_{a_3} - R_{a_2}) \cdot (V_1 - V_2) - (R_{a_1} - R_{a_2}) \cdot (V_3 - V_2)}{(V_3 - V_2) \cdot (V_1 - V_2) \cdot (V_3 - V_1)};$$

$$A = \frac{(6{,}45 - 6{,}31) \cdot (12 - 14) - (6{,}23 - 6{,}31) \cdot (16 - 14)}{(16 - 14) \cdot (12 - 14) \cdot (16 - 12)} = -0{,}0275;$$

$$B = \frac{(R_{a_1} - R_{a_2}) \cdot (V_3^2 - V_2^2) - (R_{a_3} - R_{a_2}) \cdot (V_1^2 - V_2^2)}{(V_1 - V_2) \cdot (V_3 - V_2) \cdot (V_3 - V_1)};$$

$$B = \frac{(6{,}23 - 6{,}31) \cdot (16^2 - 14^2) - (6{,}45 - 6{,}31) \cdot (12^2 - 14^2)}{(12 - 14) \cdot (16 - 14) \cdot (16 - 12)} = -0{,}155;$$

$$C = R_{a_2} - A \cdot V_2^2 - B \cdot V_2;$$

$$C = 6{,}31 - (-0{,}0275) \cdot 14^2 - (-0{,}155) \cdot 14 = 13{,}87.$$

В результате:

$$R_a^* = -0{,}0275 \cdot V^{*2} - 0{,}155 \cdot V^* + 13{,}87$$

и при $V^* = 15$ м/мин:

$$R_a^* = -0{,}0275 \cdot 15^2 - 0{,}155 \cdot 15 + 13{,}87 = 6{,}375 \text{ мкм}.$$

Определим абсолютную и относительную погрешность точечной аппроксимации при локальном параболическом интерполировании:

$$R_{a_\Delta} = 6{,}372 - 6{,}375 = -0{,}003 \text{ мкм};$$

$$\xi_{R_a} = \frac{6{,}372 - 6{,}375}{6{,}372} \cdot 100 = -0{,}05\%.$$

Погрешность аппроксимации уменьшилась примерно в два раза.

При интерполяции многочленом Ньютона, опираясь на зависимость (13):

$$R_a^* = R_{a_2} + t \cdot \Delta^{(1)} R_{a_2} + \frac{t \cdot (t-1)}{2!} \cdot \Delta^{(2)} R_{a_2} + \frac{t \cdot (t-1) \cdot (t-2)}{3!} \cdot \Delta^{(3)} R_{a_2},$$

где $t = \frac{V^* - V_2}{h}$, $h = V_3 - V_2$, $\Delta^{(1)} R_{a_2} = R_{a_3} - R_{a_2}$, $\Delta^{(2)} R_{a_2} = \Delta^{(1)} R_{a_3} - \Delta^{(1)} R_{a_2}$

$\Delta^{(1)} R_{a_3} = R_{a_4} - R_{a_3}$, $\Delta^{(3)} R_{a_2} = \Delta^{(2)} R_{a_3} - \Delta^{(2)} R_{a_2}$, $\Delta^{(2)} R_{a_3} = \Delta^{(1)} R_{a_4} - \Delta^{(1)} R_{a_3}$,

$\Delta^{(1)} R_{a_4} = R_{a_5} - R_{a_4}$.

Таким образом: $t = 0{,}5$, $\Delta^{(1)} R_{a_2} = 0{,}14$, $\Delta^{(2)} R_{a_2} = -0{,}07$, $\Delta^{(3)} R_{a_2} = 0{,}08$. В результате: $R_a^* = 6{,}373$ мкм.

Определим абсолютную и относительную погрешность точечной аппроксимации при локальном интерполировании многочленом Ньютона:

$$R_{a_\Delta} = 6{,}372 - 6{,}373 = -0{,}001 \text{ мкм};$$

$$\xi_{R_a} = \frac{6{,}372 - 6{,}373}{6{,}372} \cdot 100 = -0{,}02\%.$$

Таким образом, точечная аппроксимация при локальном интерполировании многочленом Ньютона обеспечивает при прочих равных условиях наименьшую погрешность.

В соответствии с постановкой задачи (см. пример 11) разработаем алгоритм (рис. 4.10) и программу LLI её решения на языке TURBO PASCAL.

ПРОГРАММА LLI

(постановка задачи см. пример 11, алгоритм см. рис. 4.10)

```
Program LLI; {LLI – локальная линейная интерполяция}
Uses Crt;
Var Vz, S, d, ai, bi, Radel, PRa, Raz, Rat: Real;
N, i: Byte; V, Ra: Array[1..50] Of Real;
Begin
Clrscr;
Write('Введите заданную скорость резания в м/мин:');
Readln(Vz); {Vz = 15 м/мин}
Write('Введите подачу в мм/об:');
Readln(S); {S = 0,1 мм/об}
Write('Введите диаметр сверла в мм:');
Readln(d); {d = 12 мм}
Write('Введите число опытов:');
Readln(N); {N = 5}
Writeln('Ввод табличных значений скорости резания.');
For i:=1 To N Do Begin
Write('V[',i,']=');
Readln(V[i]); {V1 = 12; V2 = 14; V3 = 16; V4 = 18; V5 = 20 м/мин}
End;
Writeln('Ввод табличных значений шероховатости.');
For i:=1 To N Do Begin
Write('Ra[',i,']=');
Readln(Ra[i]); {Ra1 = 6,23; Ra2 = 6,31; Ra3 = 6,45; Ra4 = 6,52; Ra5 = 6,6 мкм}
End;
i:=0;
Repeat
i:=i+1
Until (Vz>V[i]);
ai:=(Ra[i]-Ra[i-1])/(V[i]-V[i-1]);
```

Рис. 4.10. Блок-схема алгоритма нахождения приближенного значения выходного параметра R_a^* при реализации локальной линейной интерполяции (постановка задачи см. пример 10)

```
bi:=Ra[i-1]-ai*V[i-1];
Raz:=ai*Vz+bi; {Raz–приближенное значение шероховатости}
Rat:=6.36*Exp(0.25*Ln(d))*Exp(0.12*Ln(Vz))*Exp(0.41*Ln(S));
{Rat – наиболее близкое к точному значение шероховатости}
Radel:=Rat-Raz; {Абсолютная погрешность аппроксимации}
PRa:=((Rat-Raz)/Rat)*100; {Относительная погрешность аппроксимации}
Writeln('Приближенное значение шероховатости Raz = ', Raz:6:3,' мкм.');
Writeln('Наиболее близкое к точному значение шероховатости Rat = ', Rat:6:3,' мкм.');
Writeln('Абсолютная погрешность аппроксимации: ', Radel:6:3,' мкм.');
Writeln('Относительная погрешность аппроксимации: ', PRa:6:3,' %.');
Readkey;
End.
```

5. ПРИМЕНЕНИЕ МЕТОДОВ ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ ДЛЯ НАХОЖДЕНИЯ ВЫХОДНЫХ ПАРАМЕТРОВ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ

Численное интегрирование применяют при решении задач машиностроения, связанных с определением площади поверхности заготовок или элементов конструкции машин сложной формы.

Рассмотрим сущность численного интегрирования (рис. 5.1).

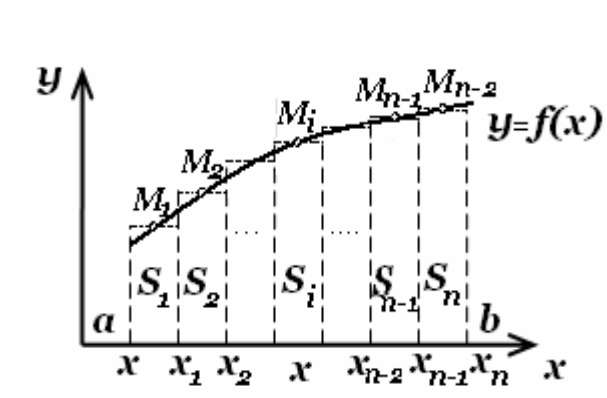


Рис. 5.1. Геометрическая интерпретация численного интегрирования

Пусть на отрезке [a, b] задана функция y = f(x). С помощью точек x_0, x_1, ..., x_n разобьем отрезок на n – элементарных отрезков $[x_{i-1}, x_i]$ (i = 1, 2, ..., n), причем $x_0 = a$, $x_n = b$. На каждом из этих отрезков выберем произвольную точку ξ_i ($x_{i-1} \le \xi_i \le x_i$) и найдем произведение S_i значения функции в этой точке $f(\xi_i)$ на длину элементарного отрезка $\Delta x_i = x_i - x_{i-1}$:

$$S_i = f(\xi_i)\Delta x_i. \qquad (14)$$

Составим сумму всех таких произведений:

$$S_n = S_1 + S_2 + ... + S_n = \sum_{i=1}^{n} f(\xi_i)\Delta x_i. \qquad (15)$$

Сумма S_n называется **интегральной суммой**.

Определенным интегралом от функции f(x) на отрезке [a, b] называется предел интегральной суммы при неограниченном увеличении числа точек разбиения, при этом длина наибольшего из элементарных отрезков стремится к нулю:

$$\int_a^b f(x)\,dx = \lim \sum_{i=1}^{n} f(\xi_i)\Delta x_i \qquad (16)$$

при $\max \Delta x_i \to 0$.

Теорема существования определенного интеграла: Если функция f(x) непрерывна на [a, b], то предел интегральной суммы существует и не зависит ни от способа разбиения отрезка [a, b] на элементарные отрезки, ни от выбора точек ξ_i.

Геометрический смысл интеграла при f(x) > 0 заключается в следующем.

Абсциссами точек M_i являются значения ξ_i, а ординатами – значения $f(\xi_i)$. Выражения (14) при i = 1, 2, ..., n описывают площади элементарных прямоугольников, а интегральная сумма (15) – площадь ступенчатой фигуры, образуемой этими прямоугольниками.

При неограниченном увеличении числа точек деления и стремлении к нулю всех элементов Δx_i, верхняя граница фигуры (ломаная) переходит в линию $y = f(x)$.

Площадь полученной фигуры, которую называют криволинейной трапецией, равна определенному интегралу (17).

Во многих случаях, когда подынтегральная функция задана в аналитическом виде, определенный интеграл удается вычислить непосредственно с помощью неопределенного интеграла (первообразной) по **формуле Ньютона-Лейбница**. Она представляет определенный интеграл в виде приращения первообразной F(x) на отрезке интегрирования:

$$\int_a^b f(x)\,dx = F(x)\Big|_a^b = F(b) - F(a). \qquad (16)$$

Однако на практике этой формулой часто нельзя воспользоваться по двум основным причинам:

1) вид функции f(x) не допускает непосредственного интегрирования, т. е. первообразную нельзя выразить в элементарных функциях;

2) значения функции f(x) заданы только на фиксированном конечном множестве точек x_i, т. е. функция задана в виде таблицы.

В этих случаях используются **методы численного интегрирования**, основанные на аппроксимации подынтегральной функции некоторыми простыми выражениями, например многочленами.

В зависимости от способа вычисления подынтегральной функции различают методы прямоугольников, трапеции, парабол и пр.

Простейшим методом численного интегрирования является **метод прямоугольников**.

При его реализации непосредственно используют замену определенного интеграла интегральной суммой (15). При этом в качестве точек ξ_i могут выбираться левые ($\xi_i = x_{i-1}$) или правые ($\xi_i = x_i$) границы элементарных отрезков.

Обозначая $f(x_i) = y_i$, $\Delta x_i = h_i$, получим следующие формулы метода прямоугольников соответственно для этих двух случаев:

$$\int_a^b f(x)\,dx = h_1y_0 + h_2y_1 + \ldots + h_ny_{n-1}, \qquad (18)$$

$$\int_a^b f(x)\,dx = h_1y_1 + h_2y_2 + \ldots + h_ny_n. \qquad (19)$$

Пример 12. Составить программу для определения площади заготовки, контур которой ограничен кривой $y = x^2$ на отрезке $x \in [0; 10]$, методом левых прямоугольников.

Алгоритм программы представлен на рис. 5.2.

```
Program Integral;
```

```
Uses crt;
Var i, n: integer;
      a, b, h, int, x: Real;
begin
clrscr;
write (‘Введите левую границу отрезка:’);
readln (a);
write (‘Введите правую границу отрезка:’);
readln (b);
write (‘Введите количество разбиений отрезка:’);
readln (n);
h:= (b-a)/n;
x:= a;
int:= 0;
for i:=1 to n do
   begin
int:= int + h*sqr(x);
  x:= x+h
     end;
writeln (‘Площадь заготовки равна’, int:4:2,’мм2’);
readkey
end.
```

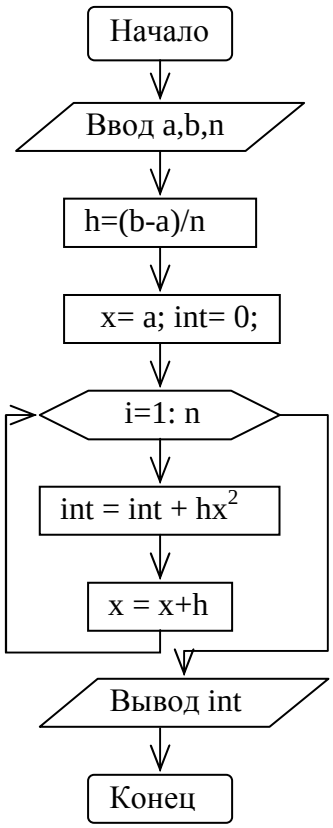


Рис. 5.2. Алгоритм программы для расчета площади заготовки методом левых прямоугольников

Более точной является формула прямоугольников, использующая значения функции в средних точках элементарных отрезков (в полуцелых узлах):

$$\int_a^b f\ (x)\ dx = \sum_{i=1}^{n} h_i\ f\ (x_{i-\ 1/2}), \qquad (20)$$

$$x_{i-\ 1/2} = (x_{i-1} + x_i) / 2 = x_{i-1} + h_i / 2,\ i = 1, 2, ..., n.$$

Метод, использующий для расчетов зависимость (20), называется **методом средних прямоугольников**.

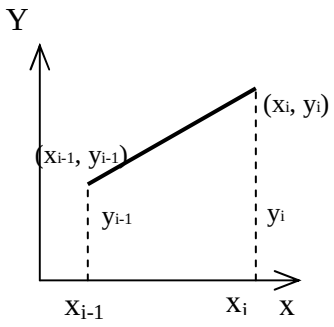


Рис. 5.3. Вид элементарной трапеции

Метод трапеций использует линейную интерполяцию, т. е. линейный график функции, соединяющей точки (x_i, y_i).

В этом случае площадь всей фигуры (криволинейной трапеции) складывается из площадей элементарных прямолинейных трапеций (рис. 5.3).

Площадь каждой такой трапеции равна произведению полусуммы оснований на высоту:

$$S_i = \frac{y_{i-1} + y_i}{2} h_i, , i = 1, 2, ..., n.$$

Сложив все эти равенства, получим **формулу трапеций**:

$$\int_a^b f\ (x)\ dx = 1/2\ \sum_{i=1}^{n} h_i\ (y_{i-1} + y_i). \qquad (21)$$

Пример 13. Модернизировать программу из примера 12, используя для расчета формулу трапеций.

```
Program Integral_Trap;
Uses crt;
Var i, n: integer;
      a, b, h, int, x: Real;
begin
clrscr;
write (‘Введите левую границу отрезка:’);
readln (a);
write (‘Введите правую границу отрезка:’);
readln (b);
write (‘Введите количество разбиений отрезка:’);
readln (n);
h:= (b-a)/n;
x:= a;
int:= 0;
for i:=1 to n do
   begin
```

```
int:= int + h*(sqr(x+h)+sqr(x))/2;
  x:= x+h
    end;
writeln ('Площадь заготовки равна', int:4:2,'мм2');
readkey
end.
```

Алгоритм программы представлен на рис. 5.4.

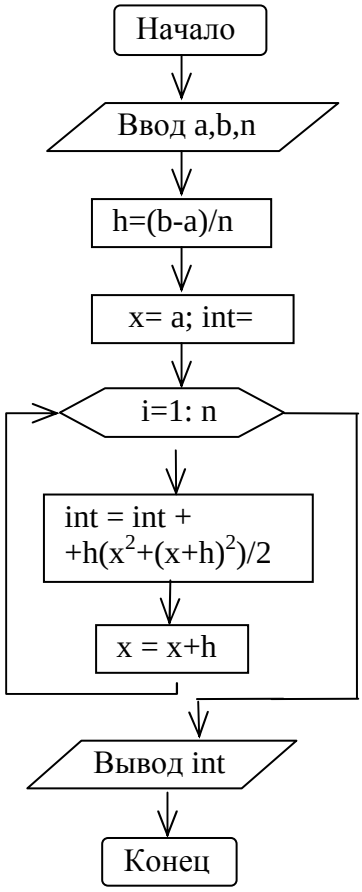


Рис. 5.4. Алгоритм программы для расчета площади заготовки методом трапеции

При численном интегрировании с постоянным шагом: $h_i = h = \text{const}$ (i = 1, 2, ..., n) формулы средних прямоугольников и трапеций принимают вид

$$\int_a^b f(x)\,dx = h\sum_{i=1}^{n} f(x_{i-1/2}), \tag{22}$$

$$\int_a^b f(x)\,dx = h\left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right). \tag{23}$$

На основании формул прямоугольников и трапеции можно получить **уточненные значения интегралов**, если учесть характер погрешностей этих формул.

Главный член погрешности формулы средних прямоугольников (22) на каждом отрезке $[x_{i-1}, x_i]$ равен $\left(\frac{1}{24}\right)h_i^3 f''(x_{i-1/2})$.

Для формулы трапеций он равен $\left(-\frac{1}{12}\right)h_i^3 f''(x_i)$, т. е. примерно вдвое больше и имеет другой знак.

На основании этого можно записать **уточненную формулу для вычисления определенного интеграла** с использованием значений I1 и I2, вычисленных по методам прямоугольников и трапеций:

$$I \approx (2*I_{\text{прямоугольников}} + I_{\text{трапеций}}) / 3 . \qquad (24)$$

Так как погрешность численного интегрирования определяется шагом разбиения, то уменьшая его, можно добиться большей точности вычисления интеграла.

Однако увеличение числа точек разбиения интервала не всегда возможно. Особенно если функция задана в табличном виде.

Поэтому в такой ситуации повысить точность численного интегрирования можно за счет повышения степени используемых интерполяционных многочленов, как в методе Симпсона.

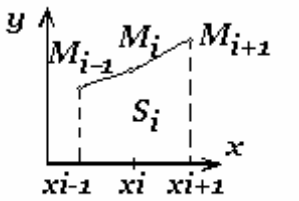


Рис. 5.5. Вид криволинейной трапеции

В **методе Симпсона** отрезок интегрирования [a, b] разбивают на четное число n равных частей с шагом h. На каждом отрезке $[x_0, x_2]$, $[x_2, x_4]$, …, $[x_{i-1}, x_{i+1}]$, …, $[x_{i-2}, x_n]$ подынтегральную функцию f(x) заменяют интерполяционным многочленом второй степени (рис. 5.5):

$$f(x) \approx \varphi_i(x) = a_i x^2 + b_i x + c_i,$$
$$x_{i-1} \le x \le x_{i+1}$$

Коэффициенты этих квадратных трехчленов могут быть найдены из условий равенства многочлена в точках x_i соответствующим табличным значениям функции y_i.

В качестве $\varphi_i(x)$ можно принять интерполяционный многочлен Лагранжа второй степени, проходящий через точки $M_{i-1}(x_{i-1}, y_{i-1})$, $M_i(x_i, y_i)$, $M_{i+1}(x_{i+1}, y_{i+1})$:

$$\varphi_i(x) = \frac{(x - x_i)(x - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} \times y_{i-1} +$$
$$+ \frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} \times y_i + \frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} \times y_{i+1}.$$

Элементарная площадь S_i (рис. 5.5) может быть вычислена с помощью определенного интеграла.

Учитывая равенства $x_{i-1} - x_i = x_i - x_{i-1} = h$, получим:

$$S_i = \int_{x_{i-1}}^{x_{i+1}} \varphi_i(x)\,dx = 1 / (2h^2) \int_{x_{i-1}}^{x_{i+1}} [(x - x_i)(x - x_{i+1})\,y_{i-1} - 2(x - x_{i-1})(x - x_{i+1})\,y_i +$$

$$+(x - x_{i-1}) \times (x - x_i)\,y_{i+1}]\,dx = h / 3(y_{i-1} + 4y_i + y_{i+1}).$$

Проведя также вычисления для каждого элементарного отрезка $[x_{i-1}, x_{i+1}]$, просуммируем полученные выражения:

$$S = h/3(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + ... + 2y_{n-2} + 4y_{n-1} + y_n).$$

Данное выражение для S принимается в качестве значения определенного интеграла **(формула Симпсона)**:

$$\int_a^b f(x)\,dx \approx h/3[y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + yn]. \qquad (25)$$

Главный член погрешности метода Симпсона имеет вид $R_n = -(h^4/180)f^{IV}(x)$.

Пример 14. Усовершенствовать программу для расчета площади заготовки (см. пример 12) с использованием метода Симпсона.

Алгоритм программы представлен на рис. 5.6.

```
Program Integral_Simpson;
Uses crt;
Var i, n: integer;
      a, b, h, int, x: Real;
begin
clrscr;
write (‘Введите левую границу отрезка:’);
readln (a);
write (‘Введите правую границу отрезка:’);
readln (b);
write (‘Введите количество разбиений отрезка:’);
readln (n);
h:= (b-a)/n;
x:= a+h/2;
int:= 0;
for i:=1 to n do
   begin
int:= int + (h/6)*(sqr(x-h/2)+4*sqr(x)+sqr(x+h/2));
  x:= x+h
     end;
writeln (‘Площадь заготовки равна’, int:4:2,’мм2’);
readkey
end.
```

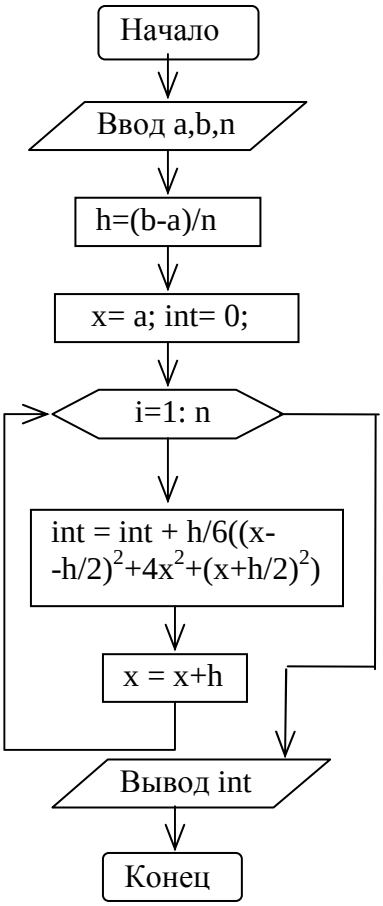


Рис. 5.6. Алгоритм программы для расчета площади заготовки методом Симпсона

Погрешность методов прямоугольников и трапеций имеет порядок $P(h^3)$, а уточненная формула (24) построена так, что коэффициент при h^3 в выражении для погрешности обращается в нуль.

Таким образом, погрешности метода Симпсона и формулы (24), использующей методы прямоугольников и трапеций, имеют один порядок.

Однако, несмотря на одинаковую точность с методом Симпсона, формула (24) требует двукратного вычисления интеграла разными методами.

Кроме того, при реализации метода Симпсона нужно почти вдвое меньше табличных значений функции, в то время как для метода прямоугольников нужны дополнительные данные в полуцелых точках.

6. ПРИМЕНЕНИЕ МЕТОДОВ ЧИСЛЕННОГО ДИФФЕРЕНЦИРОВАНИЯ ДЛЯ НАХОЖДЕНИЯ ВЫХОДНЫХ ПАРАМЕТРОВ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ

6.1. Основные понятия численного дифференцирования

Численное дифференцирование применяют при решении задач машиностроения, связанных с нахождением скоростей и ускорений перемещения машин, расчетом скоростей деформаций деталей машин и механизмов, а также с процессами распространения температур при обработке заготовок.

Напомним, что **производной функцией** $y = f(x)$ называется предел отношения приращения функции Δy к приращению аргумента Δx при стремлении Δx к нулю:

$$y' = f'(x) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}; \qquad \Delta y = f(x + \Delta x) - f(x). \tag{26}$$

Обычно для вычисления производных используют готовые формулы (таблицу производных) и не прибегают к выражению (26).

Однако в численных расчетах на ЭВМ использование этих формул не всегда удобно и возможно. Например, если функция $y = f(x)$ задана в виде таблицы. В таких случаях производные находят при помощи зависимости (26).

Значение шага Δx при этом полагают равным некоторому конечному числу и для вычисления значения производной получают приближенное равенство:

$$y' \approx \frac{\Delta y}{\Delta x}. \tag{27}$$

Это соотношение называется **аппроксимацией** (приближением) производной с помощью отношения конечных разностей, так как значения Δy и Δx в выражении (27) конечные в отличие от их бесконечно малых значений в формуле (26).

Рассмотрим аппроксимацию производной для функции $y = f(x)$, заданной в табличном виде: y_0, y_1, … при $x = x_0$, x_1, … .

x	x_0	x_1	x_2	…….	x_n
y	y_0	y_1	y_2	…….	y_n

Пусть **шаг** – разность между соседними значениями аргумента – постоянный и равен h (рис. 6.1). Запишем выражения для производной y_1 при $x = x_1$.

В зависимости от способа вычисления конечных разностей существуют различные формулы для вычисления производной в одной и той же точке:

$$\Delta y_1 = y_1 - y_0; \quad \Delta x = h; \quad y_1' \approx \frac{y_1 - y_0}{h} \text{ – } \textbf{с помощью левых разностей}. \quad (28)$$

$$\Delta y_1 = y_2 - y_0; \; \Delta x = 2h; \; y_1' \approx \frac{y_2 - y_0}{2h} \text{ – } \textbf{с помощью центральных разностей}. \quad (29)$$

$$\Delta y_1 = y_2 - y_1; \; \Delta x = h; \; y_1' \approx \frac{y_2 - y_1}{h} \text{ – } \textbf{с помощью правых разностей}. \quad (30)$$

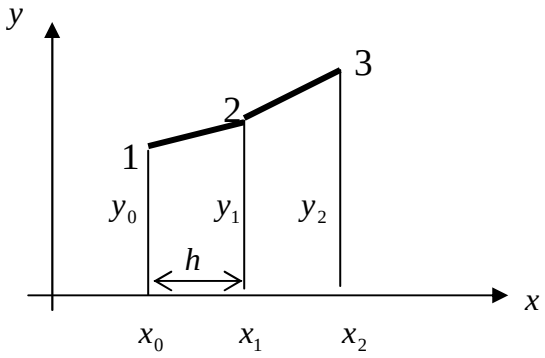


Рис. 6.1. Иллюстрация метода конечных разностей

Для второй производной будем иметь:

$$y_1'' = (y_1')' \approx \frac{y_2' - y_1'}{h} \approx \frac{(y_2 - y_1)/h - (y_1 - y_0)/h}{h} = \frac{y_2 - 2y_1 + y_0}{h^2}. \quad (31)$$

Аналогично для производных 3 и 4 порядков:

$$y_1''' \approx \frac{y_3 - 3y_2 + 3y_1 - y_0}{h^3}; \quad (32)$$

$$y_1^{IV} \approx \frac{y_4 - 4y_3 + 6y_2 - 4y_1 + y_0}{h^4}. \quad (33)$$

Таким образом, по формуле (27) можно найти приближенные значения производных любого порядка.

Пример 15. Составить программу для аппроксимации производных 1 – 4 порядков таблично заданной функции y.

Алгоритм программы представлен на рис. 6.2.

```
Program Differential;
Uses crt;      label  nn, rr;
Var  h, y0, y1, y2, y3, y4, y: real;
                        r, n: integer;
begin
clrscr;
write (‘введите шаг дифференцирования h:’);
readln (h);
```

```
nn: write (‘Выберите порядок производной:’);
writeln;
writeln (‘1 – производная 1-ого порядка’);
writeln (‘2 – производная 2-ого порядка’);
writeln (‘3 – производная 3-ого порядка’);
writeln (‘4 – производная 4-ого порядка’);
readln(n);
 case n of
1:      begin
     rr: write (‘Выберите вид аппроксимации:’);
         writeln;
         writeln (‘1 – с помощью левых разностей’);
         writeln (‘2 – с помощью центральных разностей’);
         writeln (‘3 – с помощью правых разностей’);
         readln(r);
            case r of
         1:      begin
             write (‘Введите y0:’);
             readln(y0);
             write (‘Введите y1:’);
             readln(y1);
             y:=(y1-y0)/h;
                  end;
         2:      begin
             write (‘Введите y0:’);
             readln(y0);
             write (‘Введите y2:’);
             readln(y2);
             y:=(y2-y0)/(2*h);
                  end;
         3:      begin
             write (‘Введите y1:’);
             readln(y1);
             write (‘Введите y2:’);
             readln(y2);
             y:=(y2-y1)/h;
                  end;
         else  goto rr;
           end;
           end;
```

```
2:        begin
          write (‘Введите y0:’);
          readln(y0);
          write (‘Введите y1:’);
          readln(y1);
          write (‘Введите y2:’);
          readln(y2);
          y:=(y2-2*y1+y0)/sqr(h);
             end;
3:        begin
          write (‘Введите y0:’);
          readln(y0);
          write (‘Введите y1:’);
          readln(y1);
          write (‘Введите y2:’);
          readln(y2);
          write (‘Введите y3:’);
          readln(y3);
          y:=(y3-3*y2+3*y1-y0)/(h*h*h);
          end;
3:        begin
          write (‘Введите y0:’);
          readln(y0);
          write (‘Введите y1:’);
          readln(y1);
          write (‘Введите y2:’);
          readln(y2);
          write (‘Введите y3:’);
          readln(y3);
          write (‘Введите y4:’);
          readln(y4);
          y:=(y4-4*y3+6*y2-4*y1+y0)/(sqr(h)*sqr(h));
          end;
else  goto nn;
end;
write (‘Аппроксимация производной’, n , ‘-ого порядка’);
writeln (‘Равна’, y:4:2);
readkey;
end.
```

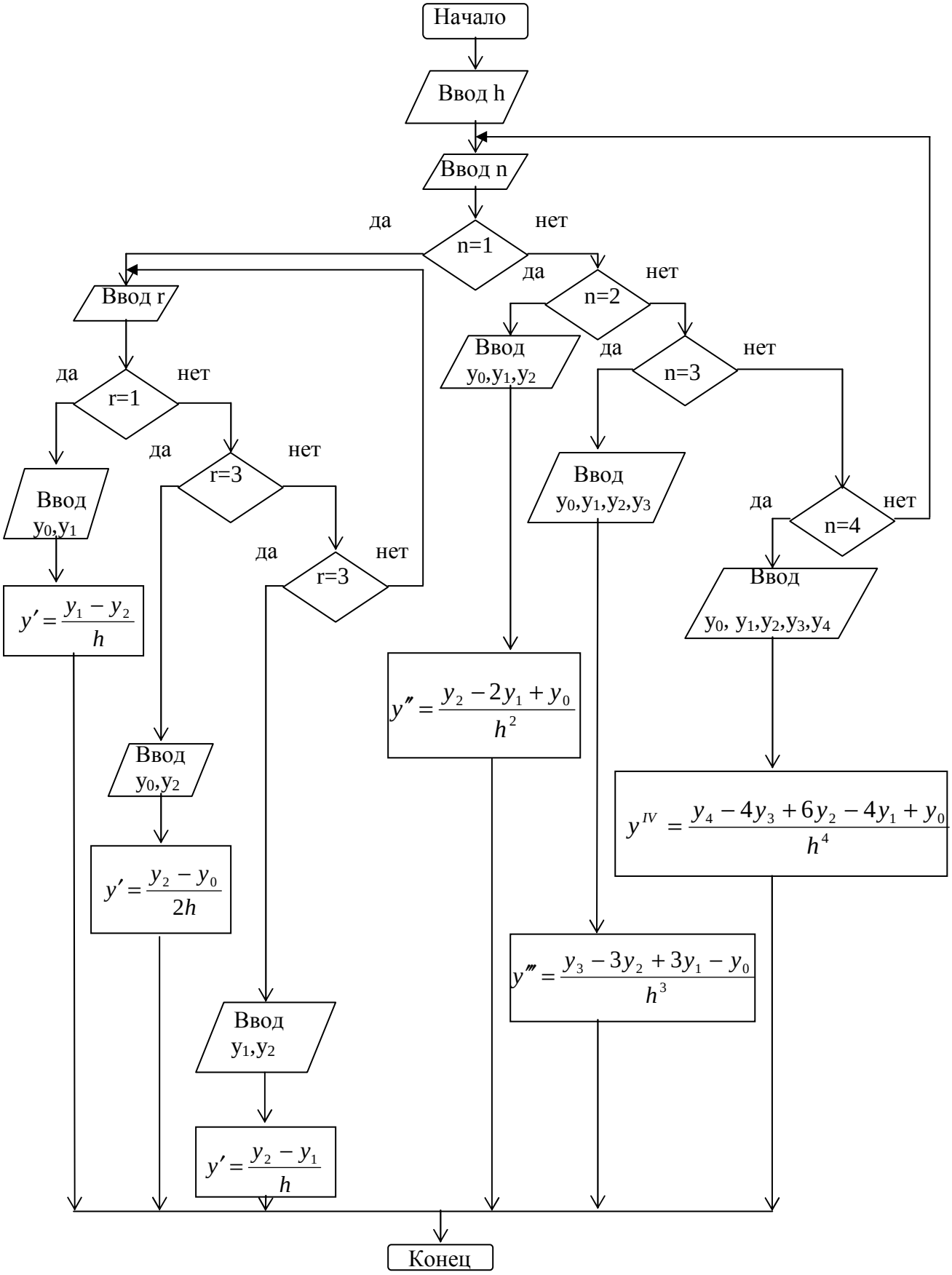


Рис. 6.2. Алгоритм программы для аппроксимации производных 1 – 4 порядков таблично заданной функции y

6.2. Определение погрешности численного дифференцирования

Определим погрешность численного дифференцирования при помощи аппроксимации производных по формуле (27).

Аппроксимируем функцию $f(x)$ некоторой функцией $\varphi(x)$, т. е. представим ее в виде:

$$f(x) = \varphi(x) + R(x). \qquad (34)$$

Дифференцируя выражение (34) необходимое число раз, можно найти значения производных $f'(x)$, $f''(x)$, …:

$$f'(x) = \varphi'(x) + R'(x), \quad f''(x) = \varphi''(x) + R''(x), \ldots.$$

В качестве приближенного значения производной порядка k функции $f(x)$ можно принять соответствующее значение производной функции $\varphi(x)$, т. е.

$$f^{(k)}(x) \approx \varphi^{(k)}(x).$$

Величина $R^{(k)}(x) = f^{(k)}(x) - \varphi^{(k)}(x)$, характеризующая отклонение приближенного значения производной от ее истинного значения, называется **погрешностью аппроксимации производной**.

При численном дифференцировании функции, заданной в виде таблицы с шагом h, эта погрешность зависит от h и ее записывают в виде: $0(h^k)$: $R^{(k)}(x) \Leftrightarrow 0(h^k)$.

Показатель степени k называется **порядком погрешности аппроксимации производной (или порядком аппроксимации)**.

При этом обычно принимают, что значение шага по модулю меньше 1: $|h| < 1$.

Оценку погрешности можно найти с помощью ряда Тэйлора:

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)}{2!}\Delta x^2 + \frac{f'''(x)}{3!}\Delta x^3 + \ldots..$$

Пусть функция $f(x)$ задана в виде таблицы $f(x_i) = y_i$ $(i = 1, 2, \ldots, n)$.

Запишем ряд Тэйлора при $x = x_1$, $\Delta x = -h$ с точностью до членов порядка n:

$$y_0 = y_1 - y_1' h + 0(h^2).$$

Отсюда найдем значение производной в точке $x = x_1$:

$$y_1' = \frac{y_1 - y_0}{h} + 0(h).$$

Это выражение совпадает с формулой (28), которая является аппроксимацией первого порядка $(k = 1)$.

Записывая ряд Тэйлора при $\Delta x = h$, можно получить аппроксимацию зависимости (30), имеющую тоже первый порядок.

Используем ряд Тэйлора для оценки погрешностей аппроксимации формул (29) и (31) . Полагая $\Delta x = h$ и $\Delta x = -h$, получим:

$$y_2 = y_1 + y_1' h + \frac{y_2''}{2!} h^2 + \frac{y_3'''}{3!} h^3 + 0(h^4);$$

$$y_0 = y_1 - y_1' h + \frac{y_2''}{2!} h^2 - \frac{y_3'''}{3!} h^3 + 0(h^4). \qquad (35)$$

Вычитая эти равенства одно из другого и преобразуя разность, получим:

$$y_1' = \frac{y_2 - y_0}{2h} + 0(h^2).$$

Следовательно, аппроксимация производной по формуле (29) с помощью центральных разностей, имеет второй порядок.

Складывая равенства (35), найдем оценку погрешности аппроксимации производной 2-ого порядка вида (31):

$$y'' = \frac{y_0 - 2y_1 + y_2}{h^2} + 0(h^2).$$

Эта аппроксимация также имеет 2 порядок.

Аналогично можно получить аппроксимацию производных более высоких порядков, оценив их погрешности.

Погрешность $R(x^k) = 0(h^k)$ зависит от нескольких факторов:

1. Погрешности уточнения, определяемой величиной остаточного члена. При уменьшении шага h она уменьшается.

2. Неточных значений функции y_i в узлах.

3. Погрешности округлений при проведении расчетов на ЭВМ. В отличие от погрешности аппроксимации погрешность округления возрастает с уменьшением шага h.

Поэтому **суммарная погрешность численного дифференцирования** $R^{(k)}(x)$ может убывать при уменьшении шага лишь до некоторого предельного значения, после чего дальнейшее уменьшение шага не повысит точность результата.

Оптимальная точность может быть достигнута за счет **регуляризации численного дифференцирования**. Регуляризация – это отклонение допустимых решений задачи с целью обеспечить их устойчивость при малых изменениях исходных параметров.

Простейшим способом регуляризации является такой выбор шага h, при котором справедливо неравенство $|f(x+h) - f(x)| > \varepsilon$, где $\varepsilon > 0$ – некоторое малое число. При вычислении производной это исключает вычитание близких по величине чисел, которое приводит к увеличению погрешности. Что усугубляется последующим делением приращения функции на малое число h.

Другой способ регуляризации – сглаживание табличных значений подбором некоторой гладкой аппроксимационной функции, т. е. с использованием интерполяционных формул.

Предположим, что функция $f(x)$, заданная в виде таблицы с постоянным шагом $h = x_i - x_{i-1}$ (i = 1, 2, …, n), может быть аппроксимирована **интерполяционным многочленом Ньютона:**

$$y \approx N(x_0 + th) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + ... + \frac{t(t-n)...(t-n+1)}{n!}\Delta^n y_0 \ , \quad (36)$$

где $t = \frac{x - x_0}{h}$; Δy_0 – конечные разности: $\Delta y_0 = y_1 - y_0$, $\Delta^2 y_0 = \Delta y_1 - \Delta y_0$; $\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$.

Дифференцируя этот многочлен по x с учетом правила дифференцирования сложной функции:

$$\frac{dN}{dx} = \frac{dN}{dt} \cdot \frac{dt}{dx} = \frac{1}{h} \cdot \frac{dN}{dt}, \quad (37)$$

получим формулы для вычисления производных любого порядка:

$$y' \approx \frac{1}{h}(\Delta y_0 + \frac{2t-1}{2!}\Delta^2 y_0 + \frac{3t^2 - 6t + 2}{3!}\Delta^3 y_0 + \frac{4t^3 - 18t^2 + 22t - 6}{4!}\Delta^4 y_0 + \frac{5t^4 - 40t^3 + 105t^2 - 100t + 24}{5!}\Delta^5 y_0 + ...) \ ; \quad (38)$$

$$y'' \approx \frac{1}{h^2}(\Delta^2 y_0 + \frac{6t-6}{3!}\Delta^3 y_0 + \frac{12t^2 - 36t + 22}{4!}\Delta^4 y_0 + \frac{20t^3 - 120t^2 + 210t - 100}{5!}\Delta^5 y_0 + ...) .$$

Остаточный член многочлена Ньютона: $R_N(x) = \frac{t(t-1)...(t-n)}{(n+1)!} f^{(n+1)}(x_*) h^{(n+1)}$

при $\Delta y_0 = \text{const}$ и $f \cdot h = \Delta^{n+1} y_0$.

Интерполяционные многочлены Ньютона (Стирлинга и Бесселя) дают выражения для производных через разности $\Delta^k y$ (k = 1, 2, …).

Однако на практике выгоднее выражать значения производных не через разности, а непосредственно через значения функции в узлах.

Для получения таких формул удобно пользоваться **интерполяционным многочленом Лагранжа с равномерным расположением узлов:**

$$L(x)=\sum_{i=0}^{n} y_i \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}. \qquad (39)$$

Тогда для трех узлов интерполяции ($n = 2;\ i = 0, 1, 2,$) имеем:

$$y = L(x) = y_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + y_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + y_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} =$$

$$= \frac{y_0}{2h^2}(x-x_1)(x-x_2) - \frac{y_1}{h^2}(x-x_0)(x-x_2) + \frac{y_2}{2h^2}(x-x_0)(x-x_1) =$$

$$= \frac{1}{2h^2}\left[y_0(x-x_1)(x-x_2) - 2y_1(x-x_0)(x-x_2) + y_2(x-x_0)(x-x_1)\right].$$

Остаточный член: $R_L(x) = \frac{y'''_*}{3!}(x-x_0)(x-x_1)(x-x_2)$.

Производная $L'(x)$ равна:

$$L'(x) = \frac{1}{2h^2}\left[y_0(2x-x_1-x_2) - 2y_1(2x-x_0-x_2) + y_2(2x-x_0-x_1)\right]; \qquad (40)$$

$$R'_L(x) = \frac{y'''_*}{3!}\left[(x-x_1)(x-x_2) + (x-x_0)(x-x_2) + (x-x_0)(x-x_1)\right].$$

Здесь y'''_* – значение производной 3 порядка в некоторой внутренней точке $x_* \in [x_0, x_n]$.

Тогда для производной y'_0 при $x = x_0$

$$y'_0 = L'(x_0) + R'_L(x) = \frac{1}{2h}(-3y_0 + 4y_1 - y_2) + \frac{h^2}{3}y'''_*. \qquad (41)$$

Аналогичные соотношения можно получить и для значений y'_1 и y'_2 при $x = x_2$ и $x = x_1$:

$$y'_1 = \frac{1}{2h}(y_2 - y_0) - \frac{h^2}{6}y'''_*; \qquad y'_2 = \frac{1}{2h}(y_0 - 4y_1 + 3y_2) + \frac{h^3}{3}y'''_*.$$

Таким образом, используя значение функции в $(n+1)$-узлах, можно получить аппроксимацию производных n-ого порядка точности.

Эти формулы можно использовать не только для узлов $x = x_0, x_1, ...$, но и для любых узлов $x = x_i, x_{i+1}, ...$, изменяя значение индексов.

Причем для четных n наиболее простые выражения и наименьшие коэффициенты в остаточных членах получаются для производных в средних (центральных) узлах (y'_1 при $h = 2$, y'_2 при $h = 4$ и т. д.).

Выпишем аппроксимации производных для узла с произвольным номером i, считая его центральным:

$$y'_i = \frac{1}{2h}(y_{i+1} - y_{i-1}) - \frac{h^2}{6} y'''_*;\ n = 2, \qquad (42)$$

$$y'_i = \frac{1}{12h}(y_{i-2} - 8y_{i-1} + 8y_{i+1} - y_{i+2}) + \frac{h^4}{30} y_*^{V};\ n = 4$$

Выражения (42) называются **аппроксимациями производных с помощью центральных разностей** и широко применяются на практике.

Остаточный член многочлена Лагранжа:

$$R_L(x) = \frac{(x - x_0)(x - x_1)...(x - x_n)}{(n+1)!} \cdot f^{(n+1)}(x_*),$$

где $f^{(n+1)}(x_*)$ – производная $(n+1)$ порядка функции $f(x)$ в некоторой точке $x = x_*,\ x_* \in [x_0, x_n]$.

Если использовать многочлен Лагранжа для случая произвольно расположенных узлов, то придется вычислять громоздкие выражения. Поэтому здесь удобнее применять **метод неопределенных коэффициентов**.

Сущность его заключается в следующем.

Искомое выражение для производной k-го порядка в некоторой точке $x = x_i$ представляют в виде линейной комбинации заданных значений функции в узлах $x_0,\ x_1, \ldots,\ x_n$:

$$y_i^{(k)} = c_0 y_0 + c_1 y_1 + ... + c_n y_n. \qquad (43)$$

При этом предполагают, что формула (43) имеет место для многочленов $y = 1,\ y = x - x_i, \ldots,\ y = (x - x_i)^n$.

Подставляя последовательно эти выражения в (40), получают систему $(n+1)$ линейных алгебраических уравнений для определения неизвестных коэффициентов $c_0,\ c_1, \ldots, c_n$.

Пример 16. Найти выражение для производной y'_1 в случае четырех равноотстоящих узлов $(n = 3)$.

Равенство (43) запишем в виде:

$$y'_1 = c_0 y_0 + c_1 y_1 + c_2 y_2 + c_3 y_3. \qquad (44)$$

Используем следующие многочлены:

$$y = 1;\ \ y = x - x_0;\ \ y = (x - x_0)^2;\ \ y = (x - x_0)^3. \qquad (45)$$

Вычислим их производные:

$$y' = 0;\ \ y' = 1;\ \ y' = 2(x - x_0);\ \ y' = 3(x - x_0)^2. \qquad (46)$$

Подставим последовательно соотношения (45) и (46) соответственно в правую и левую части равенства (44) при $x = x_1$:

$$0 = c_0 \cdot 1 + c_1 \cdot 1 + c_2 \cdot 1 + c_3 \cdot 1;$$

$$1 = c_0(x_0 - x_0) + c_1(x_1 - x_0) + c_2(x_2 - x_0) + c_3(x_3 - x_0);$$

$$2(x_1 - x_0) = c_0(x_0 - x_0)^2 + c_1(x_1 - x_0)^2 + c_2(x_2 - x_0)^2 + c_3(x_3 - x_0)^2;$$

$$3(x_1 - x_0)^2 = c_0(x_0 - x_0)^3 + c_1(x_1 - x_0)^3 + c_2(x_2 - x_0)^3 + c_3(x_3 - x_0)^3.$$

Окончательно получим систему линейных уравнений в виде:

$$\begin{cases} c_0 + c_1 + c_2 + c_3 = 0; \\ hc_1 + 2hc_2 + 3hc_3 = 1; \\ hc_1 + 4hc_2 + 9hc_3 = 2; \\ hc_1 + 8hc_2 + 27hc_3 = 3. \end{cases}$$

Решая эту систему, получим

$$c_0 = -\frac{1}{3h}; \quad c_1 = -\frac{1}{2h}; \quad c_2 = \frac{1}{h}; \quad c_3 = -\frac{1}{6h}.$$

Подставляя эти значения в равенство (41), находим выражение для искомой производной:

$$y_1' = \frac{1}{6h}(-2y_0 - 3y_1 + 6y_2 - y_3).$$

Если для аппроксимации производных использовать конечно-разностные соотношения (интерполяционные формулы), то порядок их точности прямо пропорционален числу узлов, используемых при аппроксимации. В тоже время с увеличением числа узлов эти соотношения становятся более громоздкими, и растет объем вычислений.

Однако существует простой и эффективный способ уточнения решения при фиксированном числе узлов, используемых в аппроксимирующих конечно-разностных соотношениях. Это – **метод Рунге-Ромберга**.

Пусть $F^p(x)$ – исходная производная, которая подлежит аппроксимации; $f^p(x,h)$ – конечно-разностная аппроксимация этой производной на равномерной сетке с шагом h; R – погрешность (остаточный член) аппроксимации, главный член которой можно записать в виде $h^p \cdot \varphi(x)$, т. е.

$$R = h^p \cdot \varphi(x) + 0(h^{p+1}).$$

Тогда выражение для аппроксимации производной в общем случае можно представить в виде:

$$F^p(x) = f^p(x,h) + h^p \cdot \varphi(x) + 0(h^{p+1}). \qquad (47)$$

Запишем это соотношение в той же точке x при другом шаге $h_1 = kh$:

$$F^p(x) = f^p(x,kh) + (kh)^p \cdot \varphi(x) + 0((kh)^{p+1}). \qquad (48)$$

Приравнивая правые части (47) и (48), находим выражение для главного члена погрешности аппроксимации производной:

$$h^p \cdot \varphi(x) = \frac{f^p(x,h) - f^p(x,kh)}{k^p - 1} + 0(h^{p+1}).$$

Подставляя найденное выражение в (47), получаем **формулу Рунге**:

$$F^p(x) = f^p(x,h) + \frac{f^p(x,h) - f^p(x,kh)}{k^p - 1} + 0(h^{p+1}). \qquad (49)$$

Эта формула позволяет по результатам двух расчетов значений производной $f^p(x,h)$ и $f^p(x,kh)$ (с шагом h и kh) с порядком точности p найти ее уточненное значение с порядком точности $(p+1)$.

Пример 17. Вычислить производную функции $y = x^3$ в точке $x = 1$ аналитическим и численным методами и составить программу для реализации этих расчетов.

$y' = 3x^2 \Rightarrow y'(1) = 3$ – точное значение производной.

Найдем теперь эту производную численно.

Составим таблицу значений функции:

x	0.8	0.9	1.0
y	0.512	0.729	1.0

Воспользуемся аппроксимацией производной с помощью левых разностей, имеющих первый порядок $(p = 1)$.

Примем шаг равным 0.1 и 0.2, т. е. k =2. Получим:

$$f'(x,h) = y'(1,0.1) = \frac{y(1) - y(0.9)}{0.1} = \frac{1 - 0.729}{0.1} = 2.71;$$

$$f'(x,kh) = y'(1,0.2) = \frac{y(1) - y(0.8)}{0.2} = \frac{1 - 0.512}{0.2} = 2.44.$$

По формуле Рунге найдем уточненное значение производной:

$$F'(x) = y'(1) \approx 2.71 + \frac{2.71 - 2.44}{2^1 - 1} = 2.98.$$

Таким образом, формула Рунге дает более точное значение производной. В общем случае порядок точности аппроксимации увеличивается на единицу.

Составим программу для вычисления первой производной функции $y = x^3$ в точке x = 1 аналитическим методом, с помощью конечных разностей и формулы Рунге.

Алгоритм программы представлен на рис. 6.3.

```
Program Runge;
Uses crt;
```

```
Var  x, y, y1, y2, y3, h, k: real;
Begin
Clrscr;
Write (‘Введите значение x:’);
Readln(x);
Write (‘Введите значение шага h:’);
Readln(h);
Write (‘Введите значение коэффициента k:’);
Readln(k);
y:=3*sqr(x);
y1:=(x*x*x – exp(3*ln(x – h)))/h;
y2:=(x*x*x – exp(3*ln(x –k*h)))/(k*h);
y3:=y1+(y1 – y2)/(k – 1);
Writeln(‘Значение первой производной, вычисленное:’);
Writeln(‘Аналитически – ‘, y:4:2);
Writeln(‘С помощью конечных разностей (h=’, h:4:1, ‘ ) – ‘ , y1:4:2);
 Writeln(‘С помощью конечных разностей (h=’,k* h:4:1, ‘ ) – ‘ , y2:4:2);
Writeln(‘С помощью формулы Рунге - ’,  y3:4:2);
Readkey;
End.
```

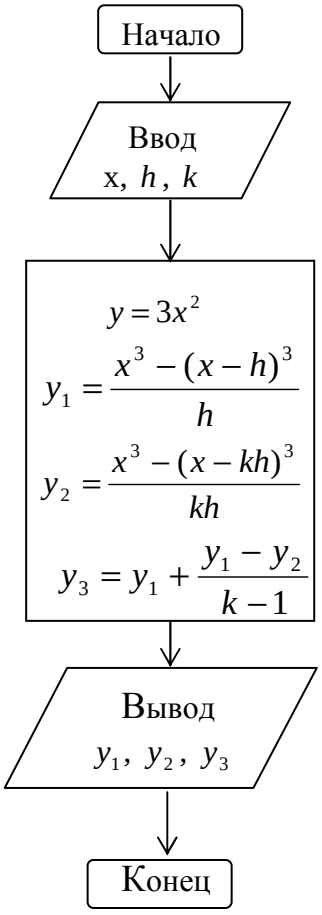


Рис. 6.3. Алгоритм программы для вычисления 1-ой производной функции аналитическим методом, с помощью конечных разностей и формулы Рунге

Предположим, что расчеты необходимо провести с шагами h_1, h_2, …, h_q. Тогда уточненное решение для производной $F^p(x)$ можно получить по **формуле Ромберга:**

$$F^p(x) = \begin{vmatrix} f^p(x,h_1) & h_1^p & h_1^{p+1} & \dots & h_1^{p+q-2} \\ f^p(x,h_2) & h_2^p & h_2^{p+1} & \dots & h_2^{p+q-2} \\ \dots\dots\dots & \dots & \dots\dots & \dots & \dots\dots \\ f^p(x,hq) & h_q^p & h_q^{p+1} & \dots & h_q^{p+q-2} \end{vmatrix} \times$$

$$\times \begin{vmatrix} 1 & h_1^p & h_1^{p+1} & \dots & h_1^{p+q-2} \\ 1 & h_2^p & h_2^{p+1} & \dots & h_2^{p+q-2} \\ \dots\dots\dots & \dots & \dots\dots & \dots & \dots\dots \\ 1 & h_q^p & h_q^{p+1} & \dots & h_q^{p+q-2} \end{vmatrix} + 0(h^{p+q-1}). \qquad (50)$$

Таким образом, порядок точности возрастает на $(q-1)$. При этом исходная функция должна иметь непрерывные производные достаточно высокого порядка.

6.3. Основные понятия, связанные с дифференциальными уравнениями

В зависимости от числа независимых переменных дифференциальные уравнения делятся на две группы:

1. обыкновенные дифференциальные уравнения, содержащие одну независимую переменную;

2. уравнения с частными производными, содержащие несколько независимых переменных.

Обыкновенными дифференциальными уравнениями называются такие уравнения, которые содержат одну или несколько производных от искомой функции y = f (x):

$$F(x, y, y', \dots, y^{(n)}) = 0, \qquad (51)$$

где x – независимая переменная.

Наивысший порядок n входящий в уравнение (51) производной называется **порядком дифференциального уравнения:**

$F(x, y, y') = 0$ – дифференциальное уравнение 1-го порядка;

$F(x, y, y', y'') = 0$ – дифференциальное уравнение 2-го порядка.

В ряде случаев из формулы (51) можно выделить старшую производную в явном виде:

$$y' = f(x, y); \quad y'' = f(x, y, y'). \qquad (52)$$

Такая форма записи называется **уравнением, разрешенным относительно старшей производной.**

Линейное дифференциальное уравнение – это уравнение, линейное относительно искомой функции и ее производных:

$y' - x^2 y = \sin^3 x$ – линейное уравнение 1-го порядка.

Решением дифференциального уравнения называется всякая функция $y = \varphi(x)$, которая после ее подстановки в уравнение превращает его в тождество.

Общее решение дифференциального уравнения n-го порядка содержит n произвольных постоянных $C_1, C_2, \ldots, C_n$, т. е. имеет вид

$$y = \varphi(x, C_1, C_2, \ldots, C_n). \qquad (53)$$

Частное решение дифференциальных уравнений получается из общего, если произвольным постоянным придать определенное значение. **Для уравнения 1-го порядка** общее решение зависит от одной произвольной постоянной:

$$y = f(x, C). \qquad (54)$$

Если постоянная принимает определенное значение $C = C_0$, то частное решение примет вид

$$y = (x, C_0).$$

Геометрическая интерпретация дифференциального уравнения 1-го порядка (49) заключается в следующем.

Так как производная y' характеризует наклон касательной к интегральной кривой в данной точке, то при $y' = k = \text{const}$ из формулы (52) получим $f(x, y) = k$ – уравнение линии постоянного наклона – изоклины. Меняя k, получаем семейство изоклин.

Геометрической интерпретацией общего решения уравнения 1-го порядка (51) является бесконечное семейство интегральных кривых с параметром C, а **частному решению** соответствует одна кривая из этого семейства.

В качестве **дополнительных условий для обыкновенных дифференциальных уравнений** могут задаваться значения искомой функции и ее производных при некоторых значениях независимой переменной, т. е. в некоторых точках.

В зависимости от способа задания дополнительных условий для получения частного решения дифференциального уравнения существуют два различных типа задач: задача Коши и краевая задача.

Если эти условия задаются в одной точке, то такая задача называется **задачей Коши.** При этом дополнительные условия называются **начальными условиями**, а точка $x = x_0$, в которой они задаются – **начальной точкой**.

Если же дополнительные условия задаются в более чем одной точке, т. е. при разных значениях независимой переменной, то такая задача называется **краевой**. Здесь сами дополнительные условия называются **граничными (или краевыми)**.

Обычно граничные условия задаются в двух точках x = a и x = в, являющихся границами области решения дифференциального уравнения.

Пример задачи Коши:

$\frac{dx}{dt} = x^2 \cos t$; t > 0; x(0) = 1.

y'' = y'/x + x^2, x > 1; y(1) = 2; y'(1) = 0.

Пример краевой задачи:

y'' + 2y' – y = sin x; $0 \le x \le 1$; y(0) = 1; y(1) = 0.

y''' = x + yy'; $1 \le x \le 3$; y(1) = 0; y'(1) = 1; y'(3) = 2.

6.4. Методы решения обыкновенных дифференциальных уравнений

Обыкновенные дифференциальные уравнения можно решить графическими, аналитическими, приближенными и численными методами

Графические методы используют геометрические построения. Например, **метод изоклин** определяет интегральные кривые по заранее построенному полю направлений, определенному изоклинами.

Аналитическими методами удается получить решения в виде формул путем аналитических преобразований.

В приближенных методах сами дифференциальные уравнения упрощаются путем обоснованного отбрасывания некоторых его членов, а также специальным выборам классов искомых функций.

Наиболее эффективными являются **численные методы** решения дифференциальных уравнений, самым распространенным и универсальным из которых является **метод конечных разностей**.

Его сущность состоит в следующем.

Область непрерывного изменения аргумента (например, отрезок) заменяется дискретным множеством точек, называемых **узлами**. Эти узлы составляют **разностную сетку**.

Искомая функция непрерывного аргумента приближенно заменяется функцией дискретного аргумента на заданной сетке. Эта функция называется **сеточной**.

Исходное дифференциальное уравнение заменяется разностным уравнением относительно сеточной функции. При этом для входящих в уравнение про-

изводных используются соответствующие конечно-разностные соотношения (см. раздел «численное дифференцирование»).

Такая замена дифференциального уравнения разностным называется его **аппроксимацией на сетке (или разностной аппроксимацией)**.

Таким образом, решение дифференциального уравнения сводится к отысканию значений сеточной функции в узлах сетки.

6.4.1. Решение задачи Коши

Решим задачу Коши при помощи разностной схемы.

Пусть требуется найти функцию $Y = Y(x)$, удовлетворяющую уравнению

$$dY/dx = f(x,Y) \qquad (55)$$

и принимающую при $x = x_0$ заданное значение Y_0:

$$Y(x_0) = Y_0. \qquad (56)$$

При этом решение нужно получить для значения $x > x_0$.

Введем последовательность точек $x_0, x_1, \ldots$ и шаги $h_i = x_{i+1} - x_i$ ($i = 0, 1, \ldots$).

В каждой точке x_i, называемой **узлом**, вместо значений функции $Y(x_i)$ вводятся числа y_i, аппроксимирующие точное решение Y на данном множестве точек.

Функцию y, заданную в виде таблицы $\{x_i, y_i\}$ ($i = 0, 1, \ldots$), называют **сеточной функцией**.

Заменяя значение производной в уравнении (55) отношением конечных разностей, осуществим переход от дифференциальной задачи (55) и (56) относительно функции Y к разностной задаче относительно сеточной функции y:

$$y_{i+1} = F(x_i, h_i, y_{i+1}, y_i, \ldots, y_{i-k+1}); \qquad (57)$$

$$y_0 = Y_0. \qquad (58)$$

Разностное уравнение (57) записано в общем виде, т. к. конкретное выражение его правой части зависит от способа аппроксимации производной и для каждого численного метода получается свой вид уравнения (57).

Если в правой части уравнения (57) отсутствует y_{i+1}, т. е. значение y явно вычисляется по k-предыдущим значениям $y_i, y_{i-1}, \ldots, y_{i-k+1}$, то разностная схема называется **явной**.

При этом получается **k-шаговый метод**: $k = 1$ – одношаговый, $k = 2$ – двухшаговый и т. д., т. е. в одношаговых методах для вычисления y_{i+1} используется лишь одно ранее найденное значение на предыдущем шаге y_i, в многошаговых – многие из них.

Если в правую часть уравнения (57) входит искомое значение y_{i+1}, то решение этого уравнения усложняется. В таких методах, называемых **неявными**,

приходится искать решение (57) относительно y_{i+1} с помощью итерациональных методов.

Простейшим одношаговым численным методом решения задачи Коши является **метод Эйлера**.

Он основан на разложении искомой функции Y(x) в ряд Тейлора в окрестностях узлов $x = x_i$ (i = 0, 1, …), в котором отрабатываются все члены, содержащие производные второго и более высоких порядков.

Запишем это разложение в виде

$$Y(x_i + \Delta x_i) = Y(x_i) + Y'(x_i)\Delta x_i + O(\Delta x_i^2). \qquad (59)$$

Заменим значение функции Y в узлах x_i значениями сеточной функции y_i. Кроме этого, согласно (55), имеем

$$Y'(x_i) = f(x_i, Y(x_i)) = f(x_i, y_i).$$

Будем считать для простоты узлы равноотстоящими, т. е. $\Delta x_i = x_{i+1} - x_i = h = \text{const}$ (i = 0, 1, …).

Учитывая введенные обозначения и пренебрегая членами порядка $O(h^2)$, из равенства (59) получим

$$y_{i+1} = y_i + hf(x_0, y_0), \; i = 0, 1, \ldots. \qquad (60)$$

Полагая i = 0, с помощью (60) найдем значение сеточной функции y_1 при $x = x_1$

$$y_1 = y_0 + hf(x_0, y_0).$$

При этом значение y_0 задано начальным условием (56), т. е. $y_0 = Y(x_0) = Y_0$.

Аналогично могут быть найдены значения сеточной функции в других узлах:

$$y_2 = y_1 + hf(x_1, y_1);$$
$$\text{----------} ;$$
$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1}).$$

Разностная схема этого метода представлена соотношениями (60). Они имеют вид рекуррентных формул, с помощью которых значение сеточной функции y_{i+1} в любом узле x_{i+1} вычисляется по ее значению y_i в предыдущем узле x_i. В связи с этим метод Эйлера и относится к одношаговым методам.

Рассмотрим алгоритм метода Эйлера (рис. 6.4). На первом этапе задают начальные значения x, y_0, а также величины шага h и количество расчетных точек n.

Решение получают в узлах x + h; x + 2h; …, x + nh.

Вывод результатов предусмотрен на каждом шаге.

Если найденное значение необходимо хранить в памяти машины, то вводят массив значений y_0, y_1, … y_n.

Поясним сущность метода Эйлера геометрически (рис. 6.5).

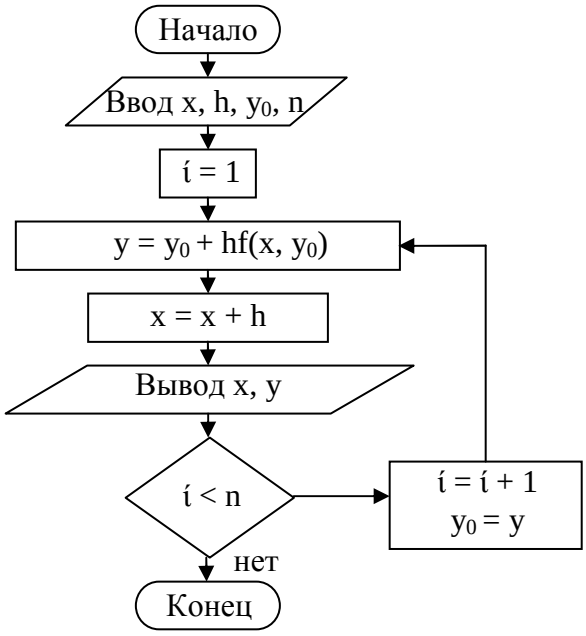


Рис. 6.4. Алгоритм метода Эйлера

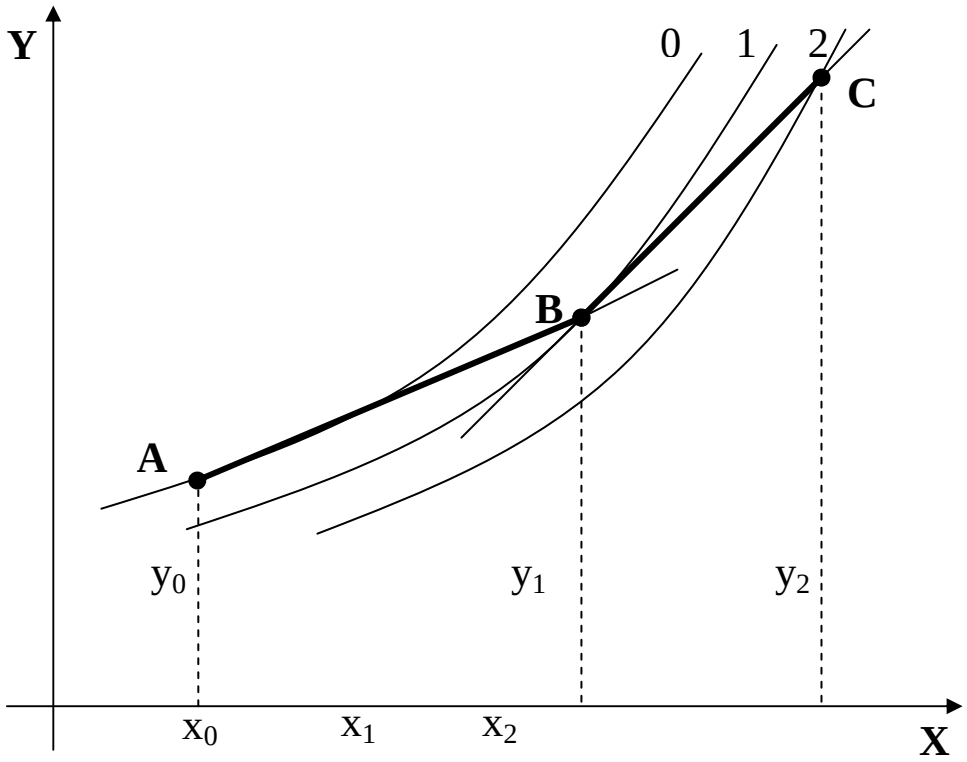


Рис. 6.5. Геометрическая интерпретация метода Эйлера

На рис. 6.50 изображены первые два шага, т. е. проиллюстрировано вычисление сеточной функции в точках x_1 и x_2. Интегральные кривые 0, 1, 2 описывают решения уравнения (55). При этом кривая 0 соответствует точному решению задачи Коши (55) и (56), т. к. она проходит через начальную точку A(x_0, y_0). Точки B и C получены в результате численного решения задачи Коши методом Эйлера. Их отклонение от кривой 0 характеризуют погрешность метода. При выполнении каждого шага мы фактически попадаем на другую интегральную кривую.

Отрезок AB – отрезок касательной к кривой 0 в точке A, ее наклон характеризуется значением производной $y'_0 = f(x_0, y_0)$.

Касательная BC уже проводится к другой интегральной кривой 1.

Таким образом, погрешность метода Эйлера приводит к тому, что на каждом шаге решение переходит на другую интегральную кривую.

Погрешность в каждой точке x_i обусловлена отброшенными членами при разложении в ряд Тейлора (59), которые имеют порядок 0 (h^2).

При нахождении решения в точке x_n, отстоящей на конечном расстоянии L от точки x_0, погрешность суммируется и становится равной $n \cdot 0(h^2)$.

С учетом того, что $h = L / n$, получим

$$n \cdot 0(h^2) = \frac{L}{h} 0(h^2) = 0(h). \qquad (61)$$

Следовательно, метод Эйлера имеет первый порядок точности.

Пример 18. Составить программу для решения дифференциального уравнения $\frac{dy}{dx} = x^2$ методом Эйлера.

Алгоритм программы представлен на рис. 6.6.

```
Program Metod_Eylera;
Uses crt;
Var  x,x0,xk,y,y0,n,h: real;
Begin
Clrscr;
Write('Введите начальное значение аргумента x0:');
Readln(x0);
Write('Введите конечное значение аргумента xk:');
Readln(xk);
Write('Введите количество узлов N:');
Readln(n);
Write('Введите начальное значение функции y0:');
Readln(y0);
Writeln;
Writeln('Результаты решения диференц. уравнения dY/dX=x^2 ');
Writeln('Методом Эйлера');
Writeln;
```

```
H:=(xk – x0)/n;
x:=x0;
y:=y0;
    while (x<=xk) do
      begin
        y:=y+h*sqr(x);
        writeln(‘при X= ’,x:4:2,’Y= ’,y:4:2);
        x:=x+h;
      end;
readkey;
End.
```

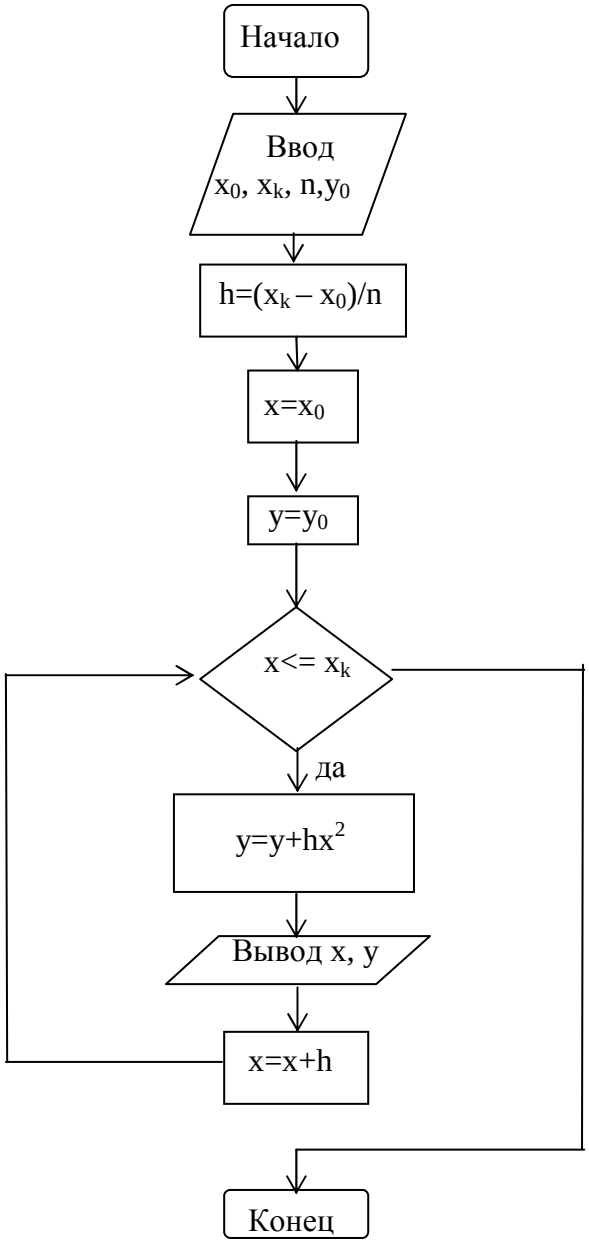


Рис. 6.6. Алгоритм программы для решения дифференциального уравнения методом Эйлера

Другим одношаговым методам является **метод Рунг-Кутта**.

На его основе могут быть построены разностные схемы любого порядка точности.

Например, для 4 порядка точности разностная схема по методу Рунге–Кутта будет такой:

$$y_{i+1} = y_i + \frac{h}{6}(k_0 + 2k_1 + 2k_2 + 2k_3),\ i = 0, 1, ...;$$
$$k_0 = f(x_i, y_i);\ k_1 = f(x_i + h/2;\ y_i + k_0/2); \qquad (62)$$
$$k_2 = f(x_i + h/2;\ y_i + k_1/2);\ k_3 = f(x_i + h,\ y_i + k_2).$$

Пример 19. Составить программу для решения дифференциального уравнения $\frac{dy}{dx} = x^2$ методом Рунге-Кутта.

Алгоритм программы представлен на рис. 6.7.

```
Program Metod_Runge_Kutta;
Uses crt;
Var  x,x0,x1,xk,y,y1,k0,k1,k2,k3,n,h: real;
Begin
Clrscr;
Write(‘Введите начальное значение аргумента x0:’);
Readln(x0);
Write(‘Введите конечное значение аргумента xk:’);
Readln(xk);
Write(‘Введите количество узлов N:’);
Readln(n);
Writeln;
Writeln(‘Результаты решения дифференц. уравнения dY/dX=x^2 ‘);
Writeln(‘Методом Рунге-Кутта’);
Writeln;
h:=(xk – x0)/n;
x:=x0;
y:=sqr(x);
Writeln(‘при X= ’,x:4:2, ‘Y= ’,y:4:2);
  While (x<=xk) do
     begin
       k0:=y;
       x1:=x+0.5*h;
       y1:=sqr(x1)+0.5k0;
       k1:=y1;
       x1:=x+0.5*h;
       y1:=sqr(x1)+k2;
       k3:=y1;
```

```
        y:=y+h*(k0+2k1+2*k2+k3)/6;
        writeln('при X= ',x:4:2,'Y= ',y:4:2);
        x:=x+h;
      end;
readkey;
End.
```

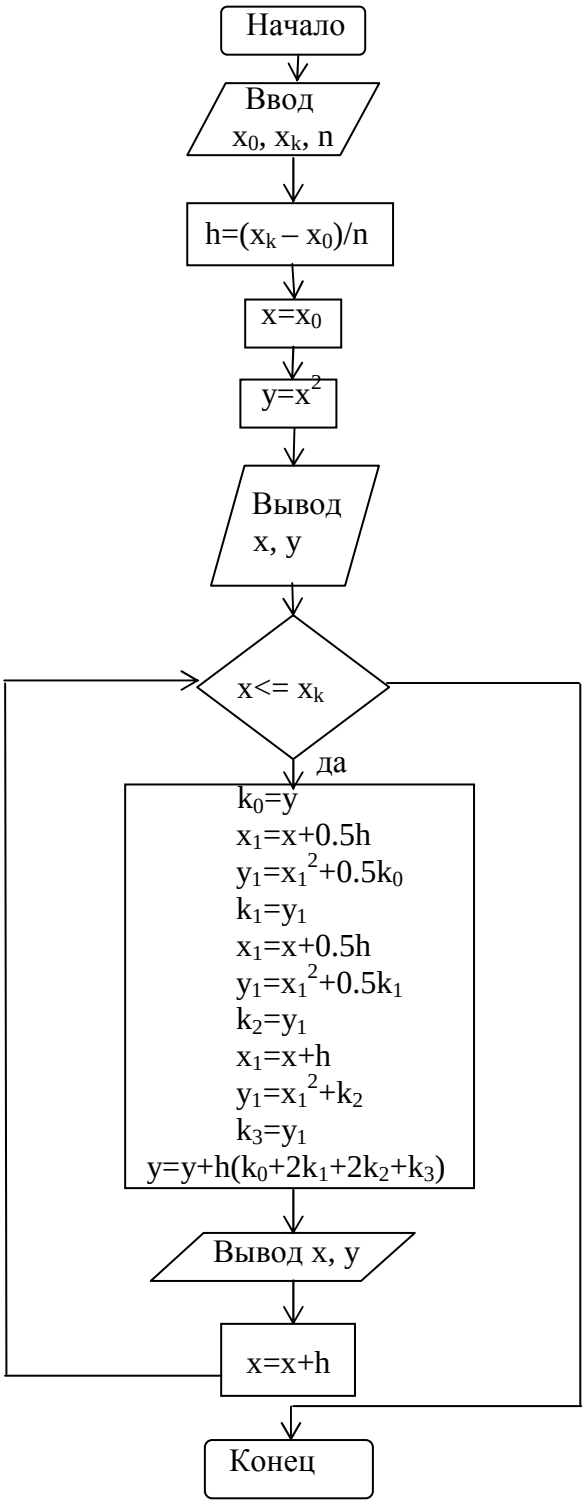


Рис. 6.7. Алгоритм программы для решения дифференциального уравнения методом Рунге-Кутта

Таким образом, метод Рунге-Кутта требует на каждом шаге четырехкратного вычисления правой части уравнения f(x, y).

При этом метод Эйлера – это метод Рунге-Кутта первого порядка.

Метод Рунге-Кутта (62) требует большего объема вычислений, однако это окупается повышенной точностью, что дает возможность проводить счет с большим шагом, т. е. для получения результатов с одинаковой точностью в методе Эйлера потребуется значительно меньший шаг, чем в методе Рунге-Кутта.

Кроме этого с уменьшением шага h локальная погрешность метода Эйлера снизится, однако при этом возрастет число узлов, что плохо повлияет на точность результатов. Поэтому метод Эйлера применяется сравнительно редко при небольшом числе расчетных точек. Наиболее употребительным одношаговым методом является метод Рунге-Кутта.

Итак, особенность одношаговых методов состоит в том, что для получения решения в каждом новом расчетном узле достаточно иметь значение сеточной функции лишь в предыдущем узле. Это позволяет начать счет при $i = 0$ по известным начальным значениям. Кроме того, указанная особенность допускает изменение шага в любой точке в процессе счета, что позволяет строить алгоритмы с автоматическим выбором шага.

В **многошаговых методах** разностная схема строится на том, что для вычисления значения y используются результаты не одного, а k-предыдущих шагов, т. е. значения y_{i-k+1}, y_{i-k+2}., …, y_i.

Запишем исходное уравнение (55) в виде

$$dY(x) = f(x, Y)dx. \qquad (63)$$

Проинтегрируем обе части этого уравнения по x на отрезке $[x_i, x_{i+1}]$.

Интеграл от левой части вычисляется легко:

$$\int_{x_i}^{x_{i+1}} dY(x) = Y(x_{i+1}) - Y(x_i) \approx y_{i+1} - y_i . \qquad (64)$$

Для вычисления интеграла от правой части уравнения (63) строится сначала интегральный многочлен $P_{k-1}(x)$ степени (k-1) для аппроксимации функции f(x, Y) на отрезке $[x_i, x_{i+1}]$ по значениям $f(x_{i-k+1}, y_{i-k+1})$, $f(x_{i-k+2}, y_{i-k+2})$, …, $f(x_i, y_i)$. После этого можно записать:

$$\int_{x_i}^{x_{i+1}} f(x, Y)dx \approx \int_{x_i}^{x_{i+1}} P_{k-1}(x)dx . \qquad (65)$$

Приравнивая (64) и (65) можно получить формулу для определения неизвестного значения функции y_{i+1} в узле x_{i+1}

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} P_{k-1}(x)dx \,. \qquad (66)$$

На основе этой формулы можно строить различные многошаговые методы любого порядка точности.

Порядок точности зависит от степени интерполяции многочлена $P_{k-1}(x)$, для построения которого используются значения сеточной функции y_i, y_{i-1}, …, y_{i-k+1}, вычисленные на k-предыдущих шагах.

Широко распространенным семейством многошаговых методов являются **методы Адамса.**

Простейший из них, получающийся при k = 1, совпадает с методом Эйлера первого порядка точности.

На практике чаще всего используют вариант метода Адамса, имеющий четвертый порядок точности и использующий на каждом шаге результаты предыдущих четырех шагов. Он, собственно, и называется **методом Адамса**.

Пусть найдены значения y_{i-3}, y_{i-2}, y_{i-1} и y_i, в четырех последовательных узлах (k = 4). При этом имеются также вычисленные ранее значения правой части f_{i-3}, f_{i-2}, f_{i-1}, f_i.

В качестве интерполяционного многочлена $P_3(x)$ можно взять многочлен Ньютона.

В случае постоянного шага h конечные разности для правой части в узле x_i имеют вид:

$$\Delta f_i = f_i - f_{i-1} \;;$$
$$\Delta^2 f_i = f_i - 2f_{i-1} + f_{i-2} \;;$$
$$\Delta^3 f_i = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3} \;.$$

Тогда **разностная схема 4 порядка метода Адамса** запишется в виде:

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f_i. \qquad (67)$$

По сравнению с методом Рунге-Кутта метод Адамса экономичнее, т. к. он требует вычисления лишь одного значения правой части на каждом шаге (метод Рунге-Кутта – четырех).

Но метод Адамса неудобен тем, что невозможно начать счет по одному лишь известному значению y_0. Расчет может быть начат только с узла x_3.

При этом значения y_1, y_2, y_3, необходимые для вычисления y_3, нужно получить каким-либо другим способом (например, методом Рунге-Кутта), что существенно усложняет алгоритм.

Кроме того, метод Адамса не позволяет (без усложнения формул) изменить шаг h в процессе счета. Этого недостатка лишены одношаговые методы.

6.4.2. Решение краевых задач

Ранее рассматривались задачи с начальными условиями, т. е. с условиями в одной (начальной) точке: при $x = x_0$, $t = 0$ и т. п.

На практике часто приходится решать такие задачи, в которых условия задаются при двух значениях независимой переменной (на концах рассматриваемого отрезка). Такие задачи, называемые **краевыми**, получаются при решении уравнений высших порядков или систем уравнений.

Рассмотрим линейное дифференциальное уравнение 2 порядка:

$$Y'' + \rho(x)Y' + g(x)Y = f(x). \qquad (68)$$

Краевая задача состоит в отыскании решения $Y=Y(x)$ уравнения (68) на отрезке [а, в], удовлетворяющего на концах отрезка условиям:

$$Y(a) = A,\ Y(в) = B. \qquad (69)$$

Граничные условия могут быть заданы не только в частном виде (69), но и в более общем виде:

$$\alpha_1 Y(a) + \beta_1 Y'(a) = A;$$
$$\alpha_i Y(b) + \beta_i Y'(b) = B. \qquad (70)$$

Краевые задачи можно решить аналитическими, приближенными и численными методами.

Аналитические методы изучаются в курсе дифференциальных уравнений и обычно применяются в исследовании различных физических процессов (теории колебаний, динамике твердого тела и т. п.).

Приближенные методы появились задолго до ЭВМ. К ним относится метод коллокаций, метод наименьших квадратов, метод Галёркина и т. д.

При приближенном решении (68) и (70) выбирается некоторая линейно независимая (базисная) система дважды дифференцируемых функций $\varphi_0(x)$, $\varphi_1(x)$, …, $\varphi_n(x)$.

При этом $\varphi_0(x)$ удовлетворяет граничным условиям (69), а $\varphi_1(x)$,…, $\varphi_n(x)$ – соответствующим однородным граничным условиям.

Искомое решение представляется в виде линейной комбинации базисных функций:

$$y(x) = \varphi_0(x) + \alpha_1\varphi_1(x) + \alpha_2\varphi_2(x) + \dots\ \alpha_n\varphi_n(x). \qquad (71)$$

Подставляя это выражение в уравнение (68), можно найти разность между его левой и правой частями, которая называется **невязкой**. Она является функцией, зависимой от переменной x и параметров $a_1, a_2, \dots, a_n$ и имеет вид:

$$\Psi(x, a_1, a_2, \dots, a_n) = Y'' + \rho x Y' + q(x)Y - f(x). \qquad (72)$$

Коэффициенты $a_1, a_2, \dots, a_n$ стараются подобрать так, чтобы невязка была минимальной. Способ определения этих коэффициентов и характеризует тот или иной приближенный метод.

В **методе коллокаций** выбирается n – точек $x = x_i (i = 1, 2, \ldots, n, x_i \in [a, b])$, называемых точками коллокации, невязка (72) в которых приравнивается нулю. Получается система n – линейных алгебраических уравнений относительно a_1, a_2, …, a_n, решая которую можно найти эти коэффициенты и подставить их в (71).

Метод наименьших квадратов основан на минимизации суммы квадратов невязок в заданной системе точек $x_1, x_2, \ldots, x_n$. Из этих условий также получается система линейных алгебраических уравнений относительно $a_1, a_2, \ldots, a_n$.

В основе **метода Галёркина** лежит требование ортогональности базисных функций $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$ к невязке $\psi(x, a_1, \ldots, a_n)$, которое выражается в виде:

$$\int_{\alpha}^{\beta} \psi(x, a_1, \ldots, a_n)\psi_i(x) = 0, \quad i = 1, 2, \ldots, n .$$

Из этих условий также получается система линейных алгебраических уравнений относительно коэффициентов линейного соотношения (71).

Численные методы позволяют свести решение краевой задачи к последовательности решений задач Коши и непосредственно применению конечно-разностных методов. К ним относятся метод стрельбы, метод конечных разностей и т. п.

Рассмотрим краевую задачу для уравнения 2 порядка, разрешенного относительно 2 производной:

$$Y'' = f(x, Y, Y'). \qquad (73)$$

Найдем решение $y = y(x)$ этого уравнения на отрезке [0, 1].

Граничные условия возьмем из (69):

$$Y(0) = y_0; Y(1) = y_1. \qquad (74)$$

Сущность **метода стрельбы** заключается в сведении решения краевой задачи (73), (74) к решению задач Коши для того же уравнения (73) с начальными условиями:

$$Y(0) = y_0; Y'(0) = k = tg\alpha. \qquad (75)$$

Поясним идею метода стрельбы геометрически (рис. 6.8).

Здесь y_0 – точка на оси ординат, в которой помещается начало искомой интегральной кривой; α – угол наклона касательной к интегральной кривой в этой точке.

Считая решение задачи Коши $Y = Y(x, \alpha)$ зависящим от параметра α, будем искать такую интегральную кривую $Y = Y(x, \alpha_*)$, которая выходит из точки $(0, y_0)$ и попадает в точку $(1, y_1)$.

Таким образом, если $\alpha = \alpha_*$, то решение $Y(x, \alpha)$ задачи Коши совпадает с решением $Y(x)$ краевой задачи.

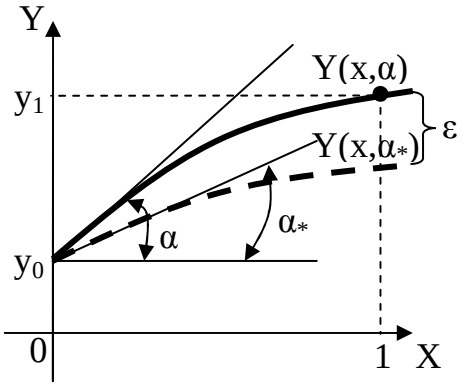


Рис. 6.8. Геометрическая интерпретация метода стрельбы

При x = 1, учитывая второе граничное условие (74), получим $Y(1, \alpha) = y_1$ или

$$Y(1, \alpha) - y_1 = 0. \qquad (76)$$

Следовательно, получим уравнение вида $F(\alpha) = 0$, где $F(\alpha) = Y(1, \alpha) - y_1$.

Это уравнение отличается от привычной записи тем, что функцию $F(\alpha)$ нельзя представить в виде некоторого аналитического выражения, т. к. она является решением задачи Коши (73), (75).

Для решения этого уравнения может быть использован любой из методов решения нелинейных уравнений, например, метод деления отрезка пополам.

Найдем начальный отрезок $[\alpha_0, \alpha_1]$, содержащий значение α_*, на концах которого функция $F(\alpha)$ принимает значения разных знаков. Для этого решение задачи Коши $Y(1, \alpha_0)$ должно при x = 1 находится ниже точки y_1, а $Y(1, \alpha_1)$ – выше.

Далее, полагая $\alpha_2 = (\alpha_0 + \alpha_1) / 2$, снова решаем задачу Коши при $\alpha = \alpha_2$ и в соответствии с методом деления отрезка пополам отбрасываем один из отрезков: $[\alpha_0, \alpha_2]$ или $[\alpha_2, \alpha_1]$, на котором функция $F(\alpha)$ не меняет знак и т. д.

Процесс поиска решения прекращается, если разность двух последовательно найденных значений α меньше некоторого заданного малого числа, т. е. $\alpha_{i+1} - \alpha_i \le \varepsilon$.

В этом случае последнее решение задачи Коши и будет искомым решением краевой задачи.

Данный алгоритм называется «методом стрельбы» потому, что в нем как бы проводится «пристрелка» по углу наклона интегральной кривой в начальной точке.

Одним из самых надежных алгоритмов метода стрельбы является **метод Ньютона**. Он состоит в следующем.

Пусть α_0 – начальное приближение α, а $\alpha_* = \alpha_0 + \Delta\alpha$ – искомое значение α. Решая задачу Коши при $\alpha = \alpha_0$ находим $Y(x, \alpha_0)$. Тогда запишем разложение в ряд с сохранением только линейных по $\Delta\alpha$ членов:

$$Y(1, \alpha_0 + \Delta\alpha) \approx Y(1, \alpha_0) + \frac{\partial Y}{\partial \alpha}\Delta\alpha.$$

Полагая $Y(1, \alpha_0 + \Delta\alpha) = Y(1, \alpha_* = y_1)$, находим

$$\Delta\alpha = \frac{y_1 - Y(1, \alpha_0)}{\partial Y(1, \alpha_0)/\partial\alpha}. \qquad (77)$$

Производную в знаменателе этого выражения можно найти численно:

$$\frac{\partial Y(1, \alpha_0)}{\partial\alpha} \approx \frac{Y(1, \alpha_0 + \delta\alpha) - Y(1, \alpha_0)}{\delta\alpha}. \qquad (78)$$

Здесь δα – произвольное малое возмущение α.

Для вычисления правой части (78) нужно решить задачу Коши при $\alpha = \alpha_0 + \delta\alpha$, в результате чего найдем значение $Y(1, \alpha_0 + \delta\alpha)$. Вычисляя затем по формуле (77) поправку Δα, находим следующее приближение параметра α: $\alpha_1 = \alpha_0 + \Delta\alpha$ и т. д.

Этот итерационный процесс продолжается до тех пор, пока очередное значение поправки Δα по абсолютной величине не станет меньше заданного малого числа ε.

В блок-схему на рис. 6.9 решение задачи Коши входит отдельным модулем с входным параметры α. На выходе из модуля получается решение $Y(x, \alpha)$ в виде значений

y_i(í = 0, 1, …, n) в точках x = 0, h, …, 1, где n = 1/h.

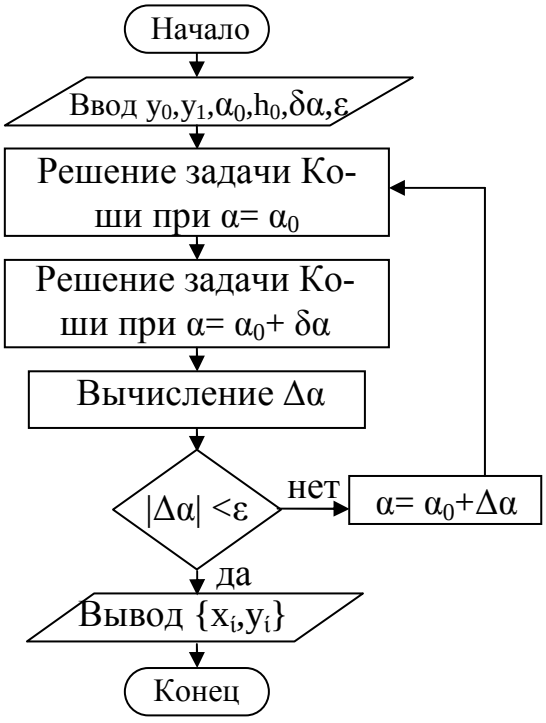


Рис. 6.9. Алгоритм метода стрельбы

Метод стрельбы может быть использован как при решении краевых задач для уравнений высших порядков, так и для систем уравнений.

Использование методов конечных разностей позволяет свести решение краевой задачи для дифференциального уравнения к решению системы алгебраических уравнений относительно значений искомой функции на заданном множестве точек.

Это достигается путем замены производных, входящих в дифференциальное уравнение, их конечно-разностными аппроксимациями.

Решим этим методом дифференциальное уравнение 2 порядка (73) при заданных граничных условиях (74).

Разобьем отрезок [0, 1] на n – равных частей точками $x_i = ih$ ($i = 0, 1, \ldots, n$).

Решение краевой задачи (73), (74) сведем к вычислению значений сеточной функции y_i в узловых точках x_i.

Для этого запишем уравнение (73) для внутренних узлов:

$$Y''(x_i) = f(x_i, Y(x_i), Y'(x_i)), \quad i = 1, 2, \ldots, n - 1. \tag{79}$$

Заменим производные, входящие в эти соотношения, их конечно-разностными аппроксимациями:

$$Y''(x_i) = \frac{y_{i+1} - y_{i-1}}{2h}; \; Y''(x_i) = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}. \tag{80}$$

Подставляя эти выражения в (79), получим систему разностных уравнений:

$$F(x_i, y_{i-1}, y_i, y_{i+1}) = 0; \quad i = 1, 2, \ldots, n - 1, \tag{81}$$

являющуюся системой (n – 1) алгебраических уравнений относительно значений сеточной функцией $y_1, y_2, \ldots, y_{n-1}$.

Входящие в данную систему y_0 (при $i = 1$) и y_n (при $i = n - 1$) берут из граничных условий, если они заданы непосредственно.

На практике часто граничные условия задаются в более общем виде:

$$\begin{aligned} a_1 Y(0) + в_1 Y'(0) &= C_1; \\ a_2 Y(1) + в_2 Y'(1) &= C_2. \end{aligned} \tag{82}$$

В этом случае граничные условия также должны представляться в разностном виде путем аппроксимации производных Y'(0) и Y'(1) с помощью конечно-разностных соотношений.

Для односторонних разностей:

$$\begin{aligned} a_1 y_0 + b_1 \frac{y_1 - y_0}{h} &= C_1; \\ a_2 y_n + b_2 \frac{y_n - y_{n-1}}{h} &= C_2. \end{aligned} \tag{83}$$

Отсюда легко найти y_0 и y_n.

Однако, предпочтительнее аппроксимировать производные, входящие в (82), со вторым порядком точности с помощью центральных разностей:

$$Y'(0) = \frac{y_1 - y_{-1}}{2h};\ Y'(1) = \frac{y_{n+1} - y_{n-1}}{2h}. \qquad (84)$$

В эти выражения входят значения сеточной функции y_{-1} и y_{n+1} в так называемых **фиктивных узлах** $x_{-1} = -h$ и $x_{n+1} = 1+h$, лежащих вне рассматриваемого отрезка. В этих узлах искомая функция должна быть определена. Количество неизвестных значений сеточной функции при этом увеличивается на два. Для замыкания системы привлекают еще два разностных уравнения (81) – при $i = 0, n$.

Таким образом, решение краевой задачи для дифференциального уравнения сведено к решению системы алгебраических уравнений вида (81). Эта система может быть линейной или нелинейной в зависимости от того, линейно или нелинейно дифференциальное уравнение (81).

Пример 20. Решить краевую задачу для линейного дифференциального уравнения 2 порядка

$$Y''(x) - \rho(x)Y(x) = f(x); \qquad (85)$$
$$\rho(x) > 0,\ 0 \le x \le 1.$$

Граничные условия

$$Y(0) = A;\ Y(1) = B. \qquad (86)$$

Разобьем отрезок [0, 1] на части постоянным шагом h с помощью узлов $x_i = ih (i = 0, 1, \ldots, n)$.

Аппроксимируем вторую производную Y" конечно-разностным соотношением (80). При этом значения искомой функции в узлах $Y(x_i)$ заменяем соответственно значениями сеточной функции y_i. Записывая уравнение (85) в каждом узле с использованием указанных аппроксимаций, получим:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \rho(x_i)y_i = f(x).$$

Обозначим через ρ_i, f_i соответственно значения $\rho(x_i)$, $f(x_i)$. После несложных преобразований приведем последнее равенство к виду:

$$y_{i-1} - (2 + h^2\rho_i)y_i + y_{i+1} = h^2 f_i,\ i = 1, 2, .., n - 1. \qquad (87)$$

Получилась система (n – 1) линейных уравнений, число которых совпадает с числом неизвестных значений сеточной функции $y_1, y_2, \ldots, y_{n-1}$ в узлах.

Ее значения на концах отрезка определены граничными условиями (86):

$$y_0 = A;\quad y_n = B. \qquad (88)$$

Решая систему уравнений (87) с учетом условий (88), находим значения сеточной функции, которые и будут приближенно равны значениям искомой функции.

7. МЕТОДЫ ОПТИМИЗАЦИИ В ЗАДАЧАХ МАШИНОСТРОЕНИЯ

7.1. Основные понятия

Оптимизация – это процесс выбора наилучшего варианта из всех возможных.

В процессе решения задачи оптимизации обычно необходимо найти оптимальные значения некоторых параметров, определяющих данную задачу. При решении инженерных задач их принято называть **проектными параметрами**, а в экономических задачах их называют **параметрами плана**. В качестве проектных параметров могут быть линейные размеры, скорость, масса, температура детали или изделия и т. п.

Число n проектных параметров x_1, x_2, …, x_n характеризует **размерность** (и степень сложности) задачи оптимизации.

Выбор оптимального решения или сравнение двух и более альтернативных решений проводится с помощью некоторой зависимой величины (функции), определяемой проектными параметрами. Эта величина называется **целевой функцией** (или критерием качества).

В процессе решения задачи оптимизации должны быть найдены такие значения проектных параметров, при которых целевая функция имеет минимум (или максимум).

В общем виде целевую функцию можно записать так:

$$u = f(x_1, x_2, \ldots, x_n). \qquad (89)$$

Примеры целевой функции – максимальная прочность или минимальная масса конструкции, максимальная мощность двигателя, минимальная себестоимость продукции, максимальная прибыль и т. п.

При n = 1 функция (89) является целевой функцией одной переменной, а ее график – некоторая кривая на плоскости.

При n = 2 зависимость (89) будет функцией 2-х переменных, и ее графиком является поверхность.

Целевая функция не всегда может быть представлена в виде формулы. Иногда она может принимать только некоторые дискретные значения, задаваться в виде таблицы.

Целевых функций может быть несколько (например, у изделия должны быть минимальный вес, максимальная прочность и минимальная себестоимость). При этом некоторые из этих функций могут оказаться несовместимыми, тогда вводят приоритет той или иной целевой функции.

Задачи оптимизации бывают **безусловные** и **условные**.

Безусловная задача оптимизации состоит в отыскании максимума или минимума действительной функции (89) от n действительных переменных и определении соответствующих значений аргументов на некотором множестве σ n-мерного пространства.

Условные задачи оптимизации (задачи с ограничениями) – это такие задачи, при формулировке которых задаются некоторые условия (ограничения) на множестве σ в виде совокупности некоторых функций, удовлетворяющих уравнениям или неравенствам.

В результате ограничений область проектирования σ, определяемая всеми n проектными параметрами, может быть существенно уменьшена в соответствии с физической сущностью задачи.

Ограничения-равенства выражают зависимость между проектными параметрами, которая должна учитываться при нахождении решения. Эти ограничения отражают законы природы, наличие ресурсов и т. д.

Число m ограничений-равенств может быть произвольным:

$$\begin{aligned} &g_1(x_1, x_2, \ldots, x_n) = 0;\\ &g_2(x_1, x_2, \ldots, x_n) = 0;\\ &\ldots\ldots\ldots\ldots\ldots\ldots;\\ &g_m(x_1, x_2, \ldots, x_n) = 0. \end{aligned} \qquad (90)$$

В ряде случаев из этих соотношений можно выразить одни проектные параметры через другие. Это позволяет исключить некоторые параметры из процесса оптимизации и уменьшить размерность задачи, то есть облегчить ее решение.

Аналогично вводятся ограничения-неравенства:

$$\begin{aligned} &\alpha_1 \le \gamma_1(x_1, x_2, \ldots, x_n) \le b_1;\\ &\alpha_2 \le \gamma_2(x_1, x_2, \ldots, x_n) \le b_2;\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots;\\ &\alpha_k \le \gamma_k(x_1, x_2, \ldots, x_n) \le b_k. \end{aligned} \qquad (91)$$

Если есть ограничения, то оптимальное решение может соответствовать либо **локальному экстремуму** (max или min) внутри области проектирования, либо значению целевой функции на границе области.

Если ограничения отсутствуют, то ищется оптимальное решение на всей области проектирования, то есть **глобальный экстремум**.

Пример 21. Спроектировать прямоугольный контейнер объемом V = 1 м^3, при этом на его изготовление должно уйти как можно меньше материала.

При постоянной толщине стенок последнее условие означает, что площадь полной поверхности контейнера S должна быть минимальной. Если обозначить через x_1, x_2 и x_3 длины ребер контейнера, то задача сведется к минимизации функции

$$S = 2(x_1x_2 + x_2x_3 + x_1x_3). \quad (92)$$

Функция (92) является целевой, а условие $V = 1$ м^3 – ограничением-равенством, позволяющим исключить один проектный параметр:

$$V = x_1x_2\,x_3 = 1;$$

$$x_3 = \frac{1}{x_1x_2}\ ; \quad (93)$$

$$S = 2\left(x_1x_2 + \frac{1}{x_1} + \frac{1}{x_2}\right).$$

Таким образом, задача свелась к минимизации двух переменных. В результате решения вначале находят x_1 и x_2, а затем x_3.

Данная задача – это задача безусловной оптимизации, так как ограничение-равенство было использовано для исключения параметра x_3. Если же задачу усложнить и потребовать, чтобы контейнер имел длину не менее 2 м, то добавляется ограничение-равенство

$$x_1 \geq 2, \quad (94)$$

и задача оптимизации становится условной: минимизируя функцию (93) и учитывая (94), найти оптимальные значения проектных параметров x_1, x_2 ($x_1 \geq 0$, $x_2 \geq 0$).

7.2. Одномерная оптимизация

Одномерная задача оптимизации в общем случае формулируется так:

Найти наименьшее (или наибольшее) значение целевой функции $y = f(x)$, заданной на множестве σ и определить значение проектного параметра $x \in \sigma$, при котором целевая функция принимает экстремальное значение.

Существование решения этой задачи вытекает из следующей теоремы.

Теорема Вейерштрасса: Всякая функция f(x), непрерывная на отрезке [a, b], принимает на этом отрезке наименьшие и наибольшие значения, то есть на отрезке [a, b] существуют такие точки x_1 и x_2, что для любого $x \in$ [a, b] имеет место неравенство

$$f(x_1) \leq f(x) \leq f(x_2).$$

Эта теорема не доказывает единственности решения. Не исключена возможность, когда равные экстремальные значения достигаются сразу в нескольких точках данного отрезка.

Для различных классов целевых функций существуют разные методы оптимизации – методы экстремумов и поиска.

7.2.1. Метод экстремумов

Наиболее просто найти оптимум, если целевая функция задана в виде y = f(x), при этом она дифференцируема на отрезке [a, b], то есть может быть найдено явное выражение для ее производной f′ (x). Нахождение экстремумов таких функций рассматривается в курсе высшей математики.

Напомним суть нахождения экстремумов.

Функция f(x) может достигать своего наибольшего и наименьшего значений либо в граничных точках отрезка [a, b], либо в точках минимума и максимума. Последние точки обязательно должны быть критическими, то есть производная f′(x) в этих точках обращается в нуль, – это **необходимое условие экстремума**.

Следовательно, для определения наименьшего или наибольшего значений f(x) на отрезке [a, b] нужно вычислить ее значения во всех критических точках данного отрезка и в его граничных точках и сравнить полученные значения: наибольшее или наименьшее из них и будет искомым значением.

Пример 22. Найти наименьшее и наибольшее значения целевой функции $f(x)=\frac{x^3}{3}-x^2$ на отрезке [1, 3].

$f'(x) = x^2 - 2x$. Приравняем нулю $f'(x)$ и найдем критические точки:

$$x^2 - 2x = 0;\ \ x_1 = 0;\ \ x_2 = 2.$$

Так как $x_1 = 0$ лежит вне отрезка [1, 3], то для анализа оставляем 3 точки: $a = 1$, $x_2 = 2$ и $b = 3$.

Вычислим значения функции в этих точках:

$$f(1) = -2/3; \qquad f(2) = -4/3; \qquad f(3) = 0.$$

Наименьшего значения f(x) достигает в точке $x_2 = 2$, а наибольшего – в точке $b = 3$, то есть:

$$f_{min} = f(2) = -4/3; \qquad f_{max} = f(3) = 0.$$

Если решение уравнения $f'(x) = 0$ затруднено, то для этого используют численные методы решения нелинейных уравнений.

7.2.2. Методы поиска

В том случае, когда целевая функция задана в табличном виде или может быть вычислена при некоторых дискретных значениях аргумента, используют различные **методы поиска**. Они основаны на вычислении целевой функции в отдельных точках и выборе среди них наибольших или наименьших значений.

Рассмотрим нахождение минимума функции f(x) на отрезке [a, b]. Пусть эта целевая функция – **унимодальна**, то есть на данном отрезке имеет только один минимум.

Процесс решения задачи методом поиска состоит в последовательном сужении (уменьшении) интервала изменения проектного параметра, называемого **интервалом неопределенности**. В начале его длина равна [b – a], а к концу она должна стать меньше заданного допустимого значения ε, то есть оптимальное значение проектного параметра, при котором целевая функция минимальна, должно находиться в интервале неопределенности – отрезке $[x_n, x_{n+1}]$, причем $x_{n+1} - x_n < \varepsilon$.

Наиболее распространенными методами поиска экстремумов целевой функции являются:

1. метод перебора;
2. метод перебора с уточнением;
3. метод золотого сечения.

Метод перебора

Этот метод позволяет уменьшить интервал неопределенности путем деления его на некоторое число равных частей с последующим вычислением значений целевой функции в точках разбиения (рис. 7.1).

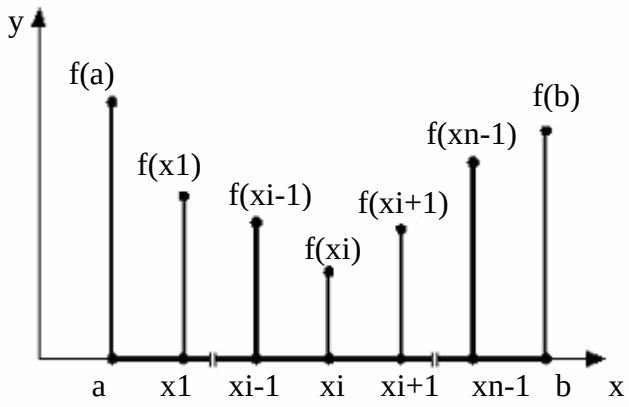


Рис. 7.1. Геометрическая интерпретация метода перебора

Пусть n – число элементарных отрезков, $h = \frac{(b-a)}{n}$ – шаг разбиения. Вычислим целевую функцию $y_k = f_k(x)$ в узлах $x_k = a + k_n$ (k = 0, 1, …, n). Сравнивая полученные значения $f(x_k)$, найдем среди них наименьшее $y_i = f(x_i)$.

Число $m_n = y_i$ можно приближенно принять за наименьшее значение целевой функции f(x) на отрезке [a, b]:

$$\lim_{n\to\infty} m_n = m.$$

С увеличением числа точек погрешность в определении минимума стремится к нулю. Недостатком этого метода являются трудность в выборе рационального количества n и оценке погрешности, а также большой объем вычислений.

Пример 23. Составить программу для нахождения экстремумов целевой функции $y = x^2$ на отрезке [a; b] методом перебора.

Алгоритм программы представлен на рис. 7.2.

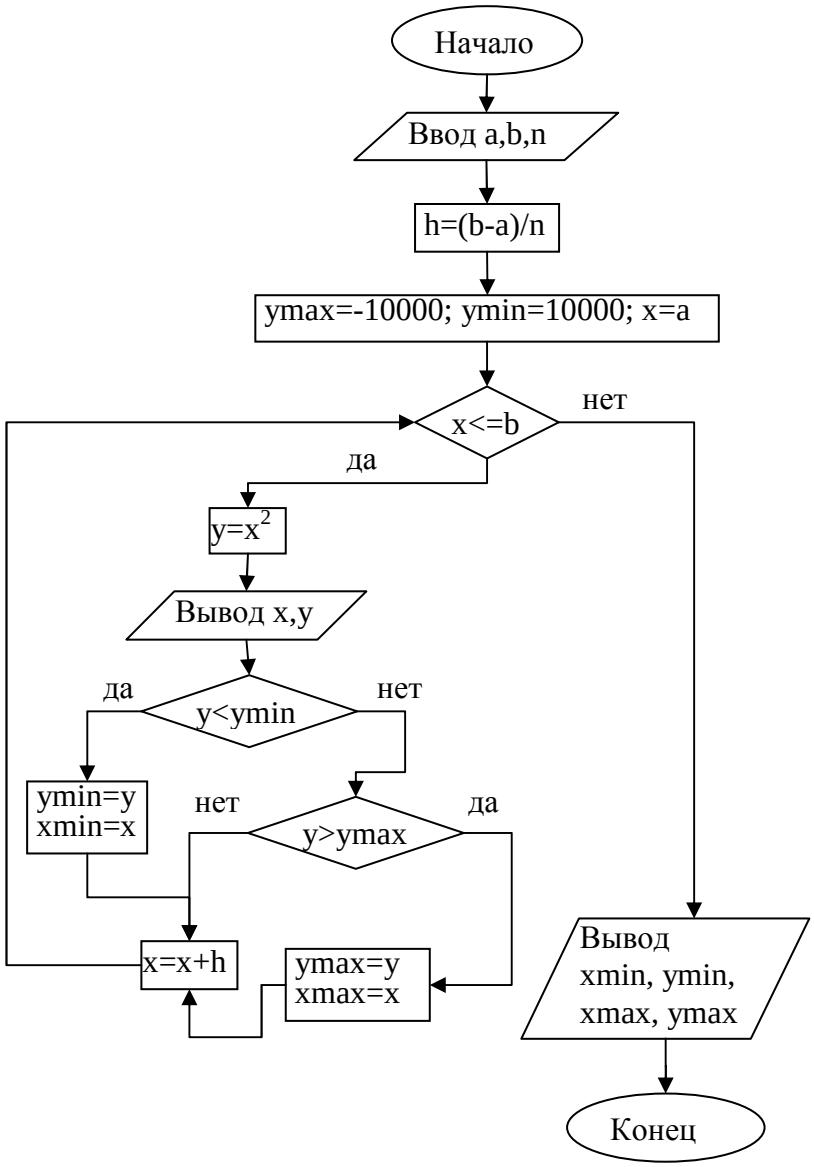


Рис. 7.2. Алгоритм программы поиска экстремумов целевой функции методом перебора

```
Program Perebor;
uses crt;
var a,b,x,xmin,xmax,y,ymin,ymax,h: real;
            i,n: integer;
begin
clrscr;
write('Введите A:');
readln(a);
write('Введите B:');
readln(b);
write('Введите N:');
```

```
readln(n);
h:=(b-a)/n;
ymax:=-100000;
ymin:=100000;
x:=a;
while (x<=b) do
  begin
y:=sqr(x);
writeln(‘ X=’, x:4:2,’ Y=’, y:4:2);
if (y<ymin) then begin
                    ymin:=y;
                    xmin:=x;
                      end;
if (y>ymax) then begin
                    ymax:=y;
                    xmax:=x;
                      end;
x:=x+h
  end;
writeln;
writeln (‘Pri X=’,xmin:4:2,’ Ymin=’,ymin:4:2);
writeln (‘Pri X=’,xmax:4:2,’ Ymax=’,ymax:4:2);
readkey
end.
```

Метод перебора с уточнением

Метод перебора с уточнением позволяет более экономично уточнять оптимальный параметр, используя свойства унимодальности целевой функции, позволяющей сузить интервал неопределенности (рис. 7.3).

Пусть среди всех значений унимодальной функции $y = f(x)$, вычисленных в узлах x_k ($k = 0, 1, \ldots, n$), наименьшим оказалось y_i. Это значит, что оптимальное значение проектного параметра находится на отрезке $[x_{i-1}, x_{i+1}]$, то есть интервал неопределенности сузился до длины двух шагов. Если размер интервала недостаточен для удовлетворения заданной погрешности, то есть $[x_{i-1}, x_{i+1}] > \varepsilon$, то его снова можно уменьшить путем нового разбиения, и так далее до достижения заданного размера интервала неопределенности.

Например, пусть начальная длина интервала неопределенности $b - a = 1$. Нужно добиться его уменьшения в 100 раз. Это можно сделать, разбив интервал на 200 частей, вычислив значения целевой функции $f(x_k)$ ($k = 0, 1, \ldots, 200$) и выбрав минимум $f(x_i)$ в искомом интервале $[x_{i-1}, x_{i+1}] = 0{,}01$.

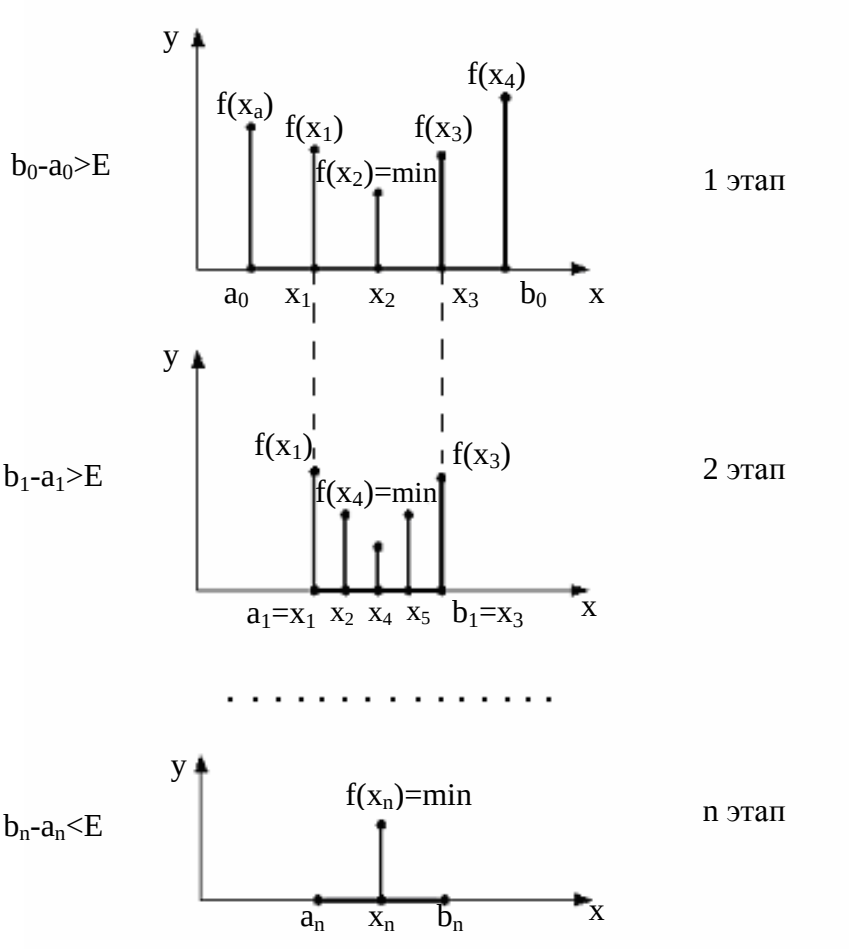


Рис. 7.3. Геометрическая интерпретация метода перебора с уточнением

Однако можно поступить иначе. Сначала разобьем отрезок [a, b] на 20 частей и найдем интервал неопределенности длиной 0,1, при этом вычислим значения целевой функции в точках $x_k = a + 0{,}05\ k$ (k = 0, 1, …, 20). Теперь отрезок $[x_{i-1}, x_{i+1}]$ снова разобьем на 20 частей; получим искомый интервал длиной 0,01, причем значения целевой функции вычисляем в точках $x_k = x_{i-1} + 0{,}005\ k$ (k = 1, 2, …, 19) (в точках x_{i-1} и x_{i+1} значения функции f(x) уже найдены). Таким образом, во втором случае в процессе оптимизации произведено 40 вычислений, а в первом случае – 201, то есть способ разбиения позволяет получить существенную экономию вычислений.

Пример 24. Составить программу для расчета минимума целевой функции $y=x^2$ на отрезке [a; b] методом перебора с уточнением.

Алгоритм программы представлен на рис. 7.4.

```
Program Perebor_s_utochneniem;
uses crt;
var a0,b0,a,b,x,xmin,y0,y,y0min,ymin,h0,h,e:real;
        i,n0,n,m: integer;
        hm:array [1..50] of real;
begin
```

```
clrscr;

write('Введите A0:');
readln(a0);
write('Введите B0:');
readln(b0);
write('Введите N0:');
readln(n0);
h0:=(b0-a0)/n0;
write('Введите E:');
readln(e);
y0min:=100000;
x:=a0;
      while (x<=b0) do
        begin
           y:=sqr(x);
           writeln('Pri X=', x:4:2,' Y=', y:4:2);
               if (y<y0min) then begin
                       y0min:=y;
                       xmin:=x;
                         end;
         x:=x+h0
        end;
writeln;
writeln ('Pri X0=',xmin:4:2,' Y0min=',y0min:4:2);
writeln;
ymin:=y0min;
hm[1]:=h0;
i:=2;
  repeat
   a:=xmin-hm[i-1];
   b:=xmin+hm[i-1];
   write ('Введите N:');
   readln(n);
   hm[i]:=(b-a)/n;
   x:=a;
      while (x<=b) do
        begin
           y:=sqr(x);
           writeln('Pri X=', x:4:2,' Y=', y:4:6);
               if (y<ymin) then begin
                       ymin:=y;
                       xmin:=x;
                         end;
```

```
            x:=x+hm[i]
            end;
            writeln;
      writeln ('Pri X=', xmin:4:2,' Ymin=',ymin:4:6);
    i:=i+1;
   until (b-a<e);
readkey
end.
```

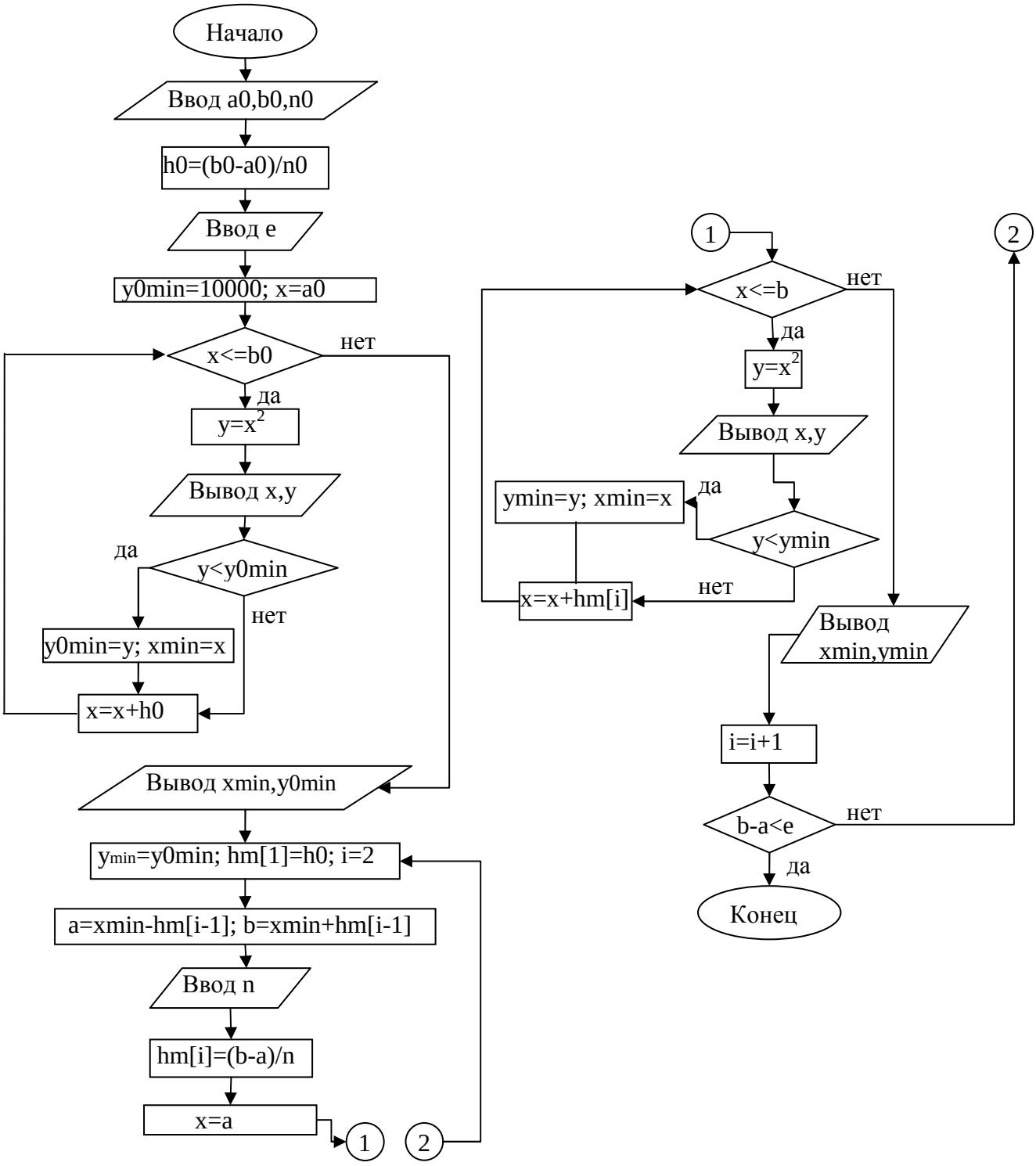


Рис 7.4. Алгоритм программы для расчета минимума целевой функции методом перебора с уточнением

Метод золотого сечения

Одним из наиболее эффективных методов поиска, в которых при ограниченном количестве вычислений целевой функции f(x) достигается наилучшая точность, является метод золотого сечения.

Сущность его заключается в построении последовательности отрезков $[a_0, b_0]$, $[a_1, b_1]$, …, стягивающихся к точке минимума функции f(x). На каждом шаге, за исключением первого, вычисление значения функции f(x) проводятся лишь один раз, в определенной точке отрезка. Эта точка, называемая **золотым сечением**, выбирается специальным образом (координаты точки были найдены в средние века Леонардо да Винчи).

Поясним сначала идею метода геометрически, а затем выведем необходимые соотношения (рис. 7.5).

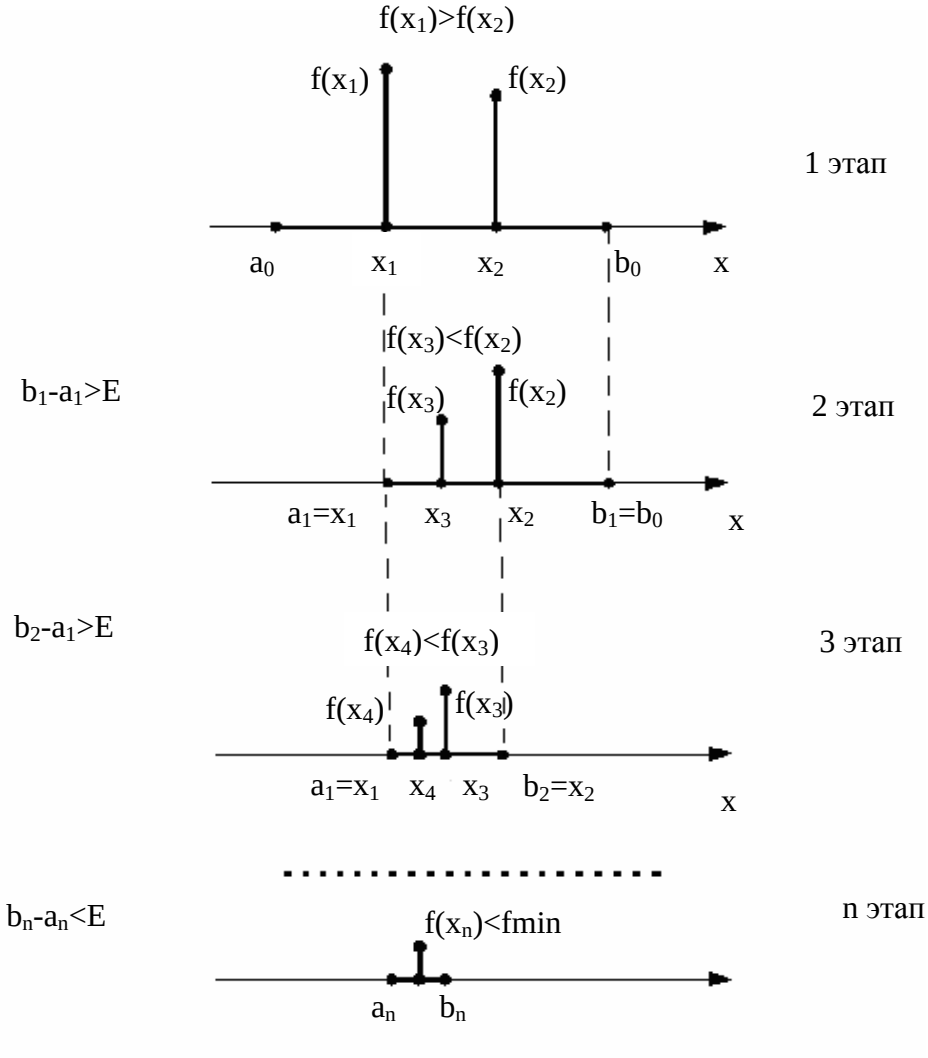


Рис. 7.5. Геометрическая интерпретация метода золотого сечения

На первом шаге оптимизации внутри отрезка $[a_0, b_0]$ выбираем две внутренние точки x_1 и x_2 и вычисляем значения целевой функции $f(x_1)$ и $f(x_2)$. Так как в данном случае $f(x_2) < f(x_1)$, то, очевидно, что минимум расположен на одном из прилегающих к x_2 отрезков $[b_0, x_2]$ или $[x_1, x_2]$. Поэтому отрезок $[a_0, x_1]$ можно отбросить, сузив тем самым первоначальный интервал неопределенности.

Второй шаг проводим на отрезке $[a_1, b_1]$, где $a_1 = x_1$, $b_1 = b_0$. Нужно снова выбрать две внутренние точки, но одна из них (x_2) осталась из предыдущего шага, поэтому достаточно выбрать лишь одну точку x_3, вычислить $f(x_3)$ и провести сравнение. Поскольку здесь $f(x_3) < f(x_2)$, ясно, что минимум находится на отрезке $[a_1, x_2]$. Обозначим этот отрезок как $[a_1, b_2]$, снова выберем одну внутреннюю точку и повторим процедуру сужения интервала неопределенности. Процесс оптимизации повторяется до тех пор, пока длина очередного отрезка $[a_n, b_n]$ не станет меньше заданной величины ε.

Теперь рассмотрим способ размещения внутренних точек на каждом отрезке $[a_k, b_k]$. Пусть длина интервала неопределенности равна ℓ, а точка деления делит его на части ℓ_1 и ℓ_2: $\ell_1 > \ell_2$, $\ell = \ell_1 + \ell_2$.

Золотое сечение интервала неопределенности выбирается так, чтобы отношение длины большего отрезка к длине всего интервала равнялось отношению длины меньшего отрезка к длине большего отрезка:

$$\frac{\ell_1}{\ell} = \frac{\ell_2}{\ell_1}. \qquad (95)$$

Из этого соотношения можно найти точку деления, определив отношение ℓ_2 / ℓ_1.

Преобразуем выражение (95) и найдем это значение:

$$\ell_1^2 = \ell_2 \ell; \qquad \ell_1^2 = \ell_2(\ell_1 + \ell_2);$$

$$\ell_2^2 + \ell_1\ell_2 - \ell_1^2 = 0; \qquad \left(\frac{\ell_2}{\ell_1}\right)^2 + \frac{\ell_2}{\ell_1} - 1 = 0; \qquad \frac{\ell_2}{\ell_1} = \frac{-1 \pm \sqrt{5}}{2}.$$

Поскольку нас интересует только положительное решение, то

$$\frac{\ell_2}{\ell_1} = \frac{\ell_1}{\ell} = \frac{-1+\sqrt{5}}{2} \approx 0{,}618.$$

Отсюда

$$\ell_1 \approx 0{,}618\ell; \qquad \ell_2 \approx 0{,}382\ell.$$

Так как заранее неизвестно, в какой последовательности (ℓ_1 и ℓ_2 или ℓ_2 и ℓ_1) делить интервал неопределенности, то рассматривают внутренние точки, соответствующие двум этим способам деления. На рисунке 7.5 точки деления выбираются с учетом полученных значений для частей отрезка. В данном случае имеем

$$x_1 - a_0 = b_0 - x_2 = 0{,}382d_0;$$
$$b_0 - x_1 = x_2 - a_0 = 0{,}618d_0;$$
$$d_0 = b_0 - a_0.$$

После первого шага оптимизации получается новый интервал неопределенности – отрезок $[a_1, b_1]$ (см. рис. 7.5). Можно показать, что точка x_3 делит этот отрезок в требуемом отношении, при этом

$$x_3 - a_1 = 0{,}382d_1; \qquad d_1 = b_0 - a_1.$$

Для этого проведем преобразования:

$$b_1 - x_2 = x_3 - x_1 = (b_0 - a_0) - (x_1 - a_0) - (b_0 - x_2) = d_0 - 0{,}382d_0 - 0{,}382d_0 = 0{,}236d_0;$$
$$d_1 = b_1 - a_1 = 0{,}618d_0;$$
$$x_3 - a_1 = 0{,}236(d_1 / 0{,}618) = 0{,}382d_1.$$

Вторая точка деления x_4 выбирается на таком же расстоянии от левой границы отрезка, то есть $x_4 - a_1 = 0{,}382d_1$. И снова интервал неопределенности уменьшается до размера

$$d_2 = b_2 - a_1 = x_2 - x_1 = 0{,}618d_1 = 0{,}618^2 d_0.$$

Используя полученные соотношения, можно записать координаты точек деления y и z отрезка $[a_k, b_k]$ на (k + 1)-м шаге оптимизации (y < z):

$$\begin{aligned} y_{k+1} &= 0{,}618\alpha_k + 0{,}382b_k; \\ z_{k+1} &= 0{,}382\alpha_k + 0{,}618b_k. \end{aligned} \qquad (96)$$

При этом длина интервала неопределенности равна

$$d_k = b_k - a_k = 0{,}618^k d_0. \qquad (97)$$

Процесс оптимизации заканчивается при $d_k < \varepsilon$. При этом проектный параметр оптимизации составляет $a_k < x < b_k$. Можно в качестве оптимального значения принять $x = a_k$ (или $x = b_k$, или $x = (a_k + b_k) / 2$ и т. п.).

На рисунке 7.6 представлен алгоритм процесса одномерной оптимизации целевой функции методом золотого сечения. Здесь y, z – точки деления отрезка [a, b], причем y < z. В результате выполнения алгоритма выдается оптимальное значение проектного параметра x, в качестве которого принимается середина последнего интервала неопределенности.

Пример 25. Составить программу для определения минимума целевой функции $y=x^2$ на отрезке [a; b] методом золотого сечения

```
Program Gold;
uses crt;
label 1,2;
var a,a0,a1,b,b0,b1,e,y,z,x:real;
begin
clrscr;
writeln('Программа поиска минимума функции Y=x^2');
writeln('Методом золотого сечения');
writeln;
```

```
write('Введите A:');
readln(a);
a0:=a;
write('Введите B:');
readln(b);
b0:=b;
write('Введите E:');
readln(e);
y:=0.618*a+0.382*b;
z:=0.382*a+0.618*b;
a1:=sqr(y);
b1:=sqr(z);
2:if (a1<b1) then begin
          b:=z;
          if(b-a)<e then goto 1
                    else begin
                      z:=y;
                      b1:=a1;
                      y:=0.618*a+0.382*b;
                      a1:=sqr(y);
                      goto 2
                      end
          end
      else begin
           a:=y;
           if(b-a)<e then goto 1
                    else begin
                      y:=z;
                      a1:=b1;
                      z:=0.382*a+0.618*b;
                      b1:=sqr(x);
                      goto 2
                      end
           end;
1:x:=(a+b)/2;
write('Pri X=',x:2:2,' Ymin=',sqr(x):2:2);
writeln(' на отрезке [',a0:2:0,';',b0:2:0,']');
readkey
end.
```

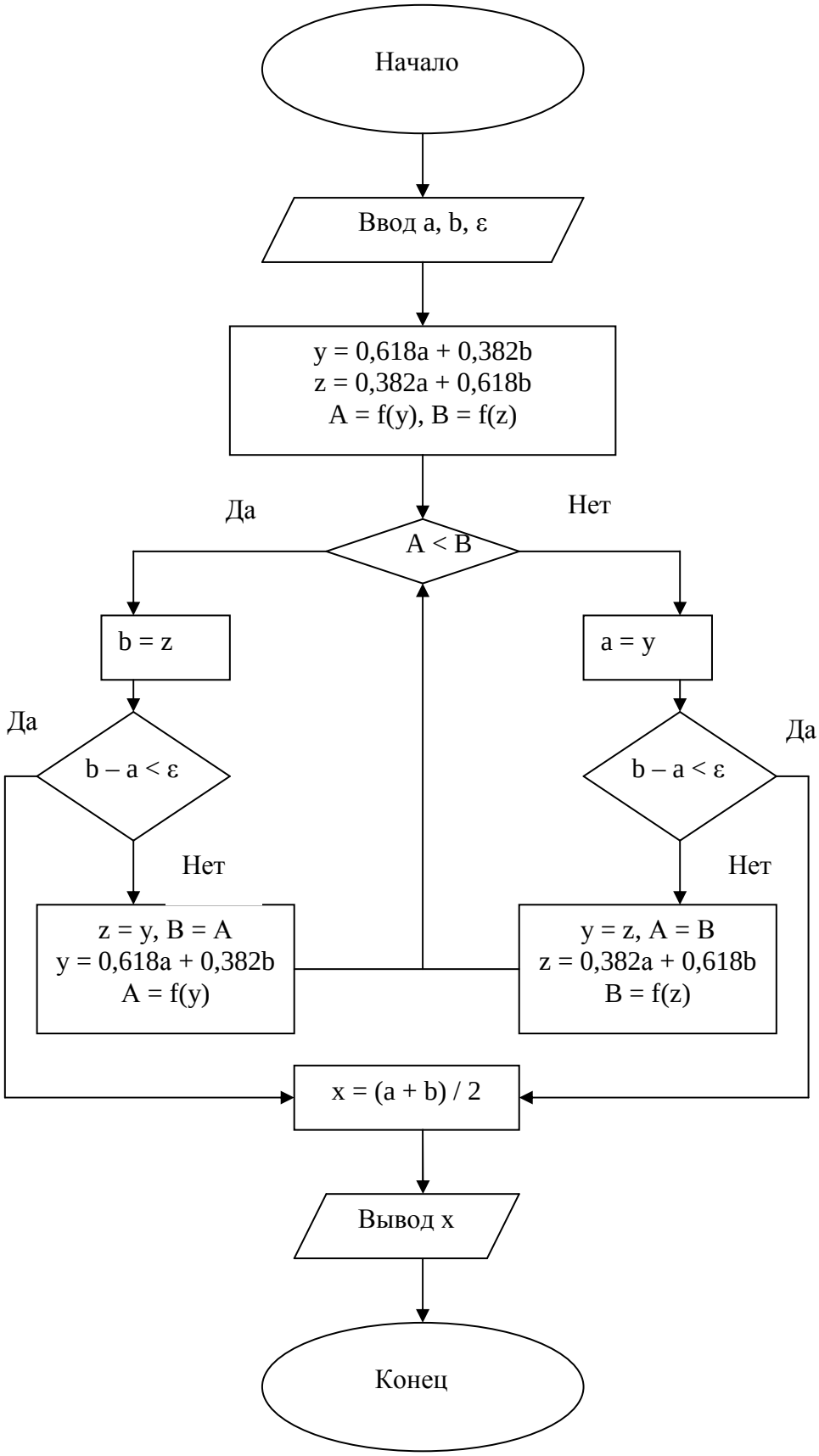


Рис. 7.6. Алгоритм метода золотого сечения

7.3. Многомерные задачи оптимизации

7.3.1. Задачи безусловной оптимизации

В рассмотренных ранее одномерных задачах оптимизации целевая функция зависела лишь от одного аргумента. Однако в большинстве реальных задач оптимизации, представляющих практический интерес, целевая функция зависит от многих проектных параметров. Например, на расход топлива автомобилем влияют объем двигателя, масса автомобиля, его лобовое сопротивление и т. п.

Метод экстремумов

Минимум дифференцируемой функции многих переменных $u = f(x_1, x_2, \ldots, x_n)$ можно найти, исследуя ее значения в критических точках, которые определяются из решения системы дифференциальных уравнений

$$\frac{\partial f}{\partial x_1} = 0,\ \frac{\partial f}{\partial x_2} = 0,\ \ldots,\ \frac{\partial f}{\partial x_n} = 0. \qquad (98)$$

Например, в разделе 7.1 рассматривалась задача об определении минимальных размеров контейнера объемом 1 м^3. Задача свелась к минимизации его полной поверхности, являющейся целевой функцией

$$S = 2\left(x_1 x_2 + \frac{1}{x_1} + \frac{1}{x_2} \right). \qquad (99)$$

В соответствии с (98) получаем систему

$$\begin{cases} \dfrac{\partial s}{\partial x_1} = 2\left(x_2 - \dfrac{1}{x_1^2} \right) = 0, \\ \dfrac{\partial s}{\partial x_2} = 2\left(x_1 - \dfrac{1}{x_2^2} \right) = 0. \end{cases}$$

Отсюда $x_1 = x_2 = 1$ м, $x_3 = 1/x_1x_2 = 1$ м. То есть оптимальная форма контейнера – куб с ребрами длиной 1 м.

Данный метод можно использовать только для дифференцируемой целевой функции. Но и в этом случае могут возникнуть проблемы при решении системы нелинейных уравнений (98).

Метод поиска

Во многих случаях целевая функция не описана никакой функцией. Имеется лишь возможность определения ее значений в произвольных точках рассматриваемой области с помощью вычислительного алгоритма или путем из-

мерений. Задача состоит в приближенном определении наименьшего значения функции во всей области при известных ее значениях в определенных точках.

Для решения подобной задачи в области проектирования G, в которых имеется минимум целевой функции $u = f(x_1, x_2, \ldots, x_n)$, можно ввести дискретное множество точек (узлов) путем разбиения интервалов изменения параметров $x_1, x_2, \ldots, x_n$ на части с шагами $h_1, h_2, \ldots, h_n$. В полученных узлах можно вычислить значения целевой функции и среди этих значений найти наименьшее.

Такой **метод общего поиска со сплошным перебором** для решения многомерных задач оптимизации не годится из-за большого объема вычисления. Здесь необходимы специальные численные методы, основанные на целенаправленном поиске.

Одним из таких методов является **метод покоординатного спуска**.

Пусть требуется найти наименьшее значение целевой функции $u = f(x_1, x_2, \ldots, x_n)$. В качестве начального приближения выберем в n-мерном пространстве некоторую точку M_0 с координатами $x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}$. Зафиксируем все координаты функции u, кроме первой. Тогда $u = f\left(x_1, x_2^{(0)}, ..., x_n^{(0)}\right)$ – функция одной переменной x_1. Решая одномерную задачу оптимизации для этой функции, мы от точки M_0 переходим к точке $M_1\left(x_1^{(1)}, x_2^{(0)}, ..., x_n^{(0)}\right)$, в которой функция u принимает наименьшее значение по координате x_1 при фиксированных остальных координатах. В этом состоит первый шаг процесса оптимизации, заключающийся в спуске по координате x_1.

Зафиксируем теперь все координаты, кроме x_2, и рассмотрим функцию этой переменной $u = f\left(x_1^{(1)}, x_2, x_3^{(0)}, ..., x_n^{(0)}\right)$. Снова решая одномерную задачу оптимизации, находим ее наименьшее значение при $x_2 = x_2^{(1)}$, то есть в точке $M_2\left(x_1^{(1)}, x_2^{(1)}, x_3^{(0)}, ..., x_n^{(0)}\right)$.

Аналогично проводится спуск по координатам $x_3, x_4, \ldots, x_n$, а затем процедура снова повторяется от x_1 до x_n и т. д.

В результате этого процесса получается последовательность точек M_0, M_1, …, в которых значения целевой функции составляют монотонно убывающую последовательность $f(M_0) \geq f(M_1) \geq \ldots$. На любом шаге k этот процесс можно прервать и принять значение $f(M_k)$ в качестве наименьшего значения целевой функции в рассматриваемой области.

Таким образом, метод покоординатного спуска сводит задачу о нахождении наименьшего значения функции многих переменных к многократному решению одномерных задач оптимизации по каждому параметру.

Для функции двух переменных z = f(x, y), описывающей некоторую поверхность в трехмерном пространстве, геометрическая интерпретация этого метода такова (рис. 7.7). Нанесем на плоскости (y0x) уровни этой поверхности.

Процесс оптимизации в этом случае проходит так. Точка $M_0(x_0, y_0)$ описывает начальное приближение. Проводя спуск по x, попадем в точку $M_1(x_1, y_0)$. Далее, двигаясь параллельно оси ординат, придем в точку $M_2(x_1, y_1)$ и т. д.

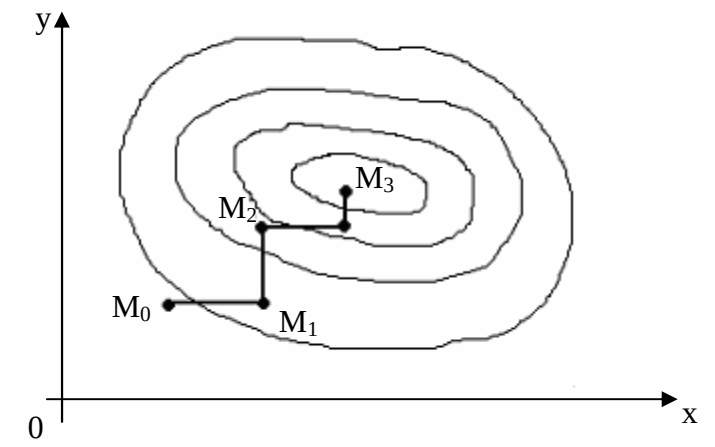


Рис. 7.7. Геометрическая интепретация метода покоординатного спуска для целевой функции z = f(x, y)

При этом важно, чтобы процесс оптимизации сходился, то есть сходилась последовательность значений целевой функции $f(M_0)$, $f(M_1)$, … к наименьшему ее значению в данной области.

Для функции двух переменных очевидно, что метод не примерим в случае наличия изломов в линиях уровня. Это соответствует «оврагу» на поверхности. Здесь возможен случай, когда спуск по одной координате приводит на «дно» оврага. Тогда любое движение вдоль другой координаты ведет к возрастанию функции, соответствующему подъему на «берег» оврага.

Для гладких функций при удачно выбранном начальном приближении (в некоторой окрестности минимума) процесс сходится к минимуму.

К достоинствам метода покоординатного спуска относится также возможность использования простых алгоритмов одномерной оптимизации.

Пример 26. Составить программу для расчета минимума целевой функции $z = x^2 + y^2$ методом покоординатного спуска.

Алгоритм программы представлен на рис. 7.8.

```
Program Koordinatny_spusk;
uses crt;
var a,z,x,x0,x1,xk,y,y0,y1,yk,zmin,h:real;
                i,n:integer;
begin
clrscr;
write('Ввод X0:');
readln(x0);
write('Ввод Xk:');
```

```
readln(xk);
write('Ввод Y0:');
readln(y0);
write('Ввод Yk:');
readln(yk);
write('Ввод H:');
readln(h);
zmin:=100000;
x:=x0; y:=y0;
while (x<=xk) do
   begin
z:=sqr(x)+sqr(y);
if(z<zmin) then begin
                 x1:=x;
               zmin:=z;
               end;
x:=x+h
  end;
while (y<=yk) do
  begin
z:=sqr(x1)+sqr(y);
if(z<zmin) then begin
                  y1:=y;
               zmin:=z;
               end;
y:=y+h
  end;
writeln;
writeln('Pri X=',x1:4:2,' Y=',y1:4:2,'Zmin=',zmin:4:2);
readkey
end.
```

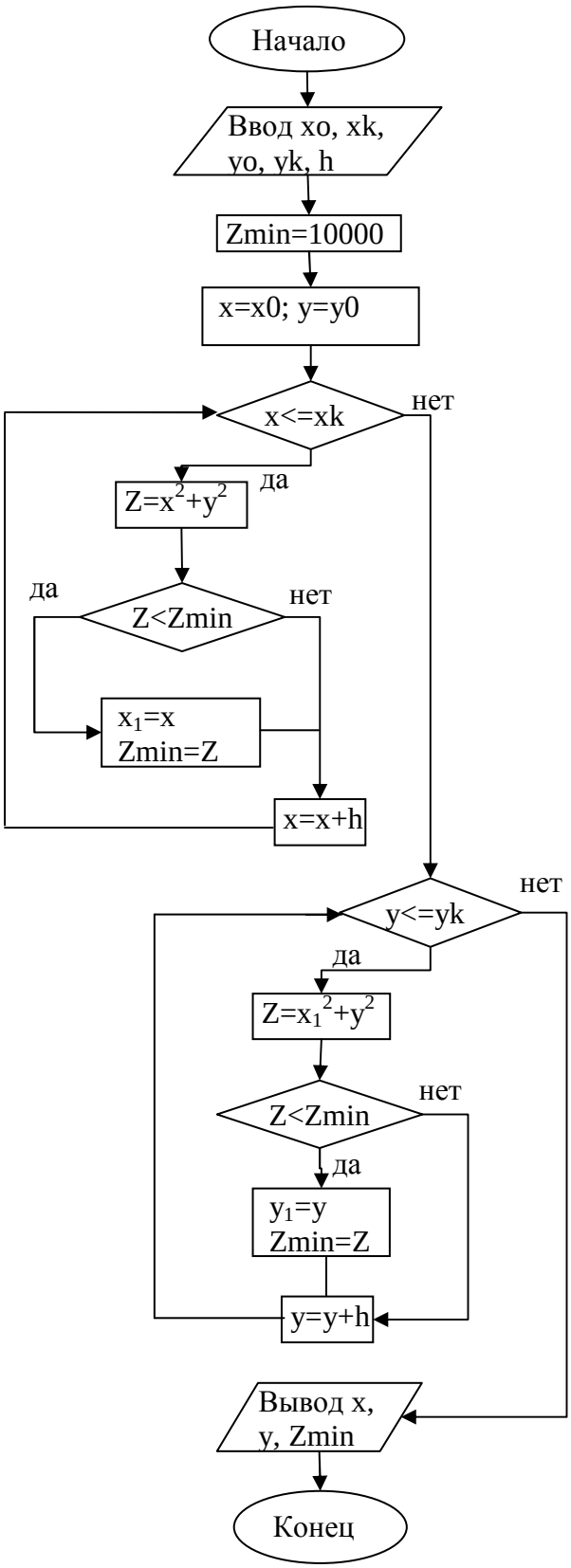


Рис. 7.8. Алгоритм программы для расчета минимума целевой функции $z = x^2 + y^2$ методом покоординатного спуска

Оптимизировать целевую функцию можно также при помощи **метода градиентного спуска**. Поясним сущность данного метода физически.

Некоторые природные явления сходны с решением задач оптимизации. Например, стекание воды с берега котлована на дно. Если принять, что его берега «унимодальны», то есть они гладкие и не содержат локальных углублений или выступов, то тогда вода устремится вниз в направлении наибольшей крутизны берега в каждой точке. Выражаясь математически, направление наискорейшего спуска соответствует направлению наибольшего убывания функции.

Из курса высшей математики известно, что направление наибольшего возрастания функции двух переменных u = f(x, y) характеризуется ее **градиентом**

$$\operatorname{grad}\vec{u} = \frac{\partial u}{\partial x}\vec{e}_1 + \frac{\partial u}{\partial y}\vec{e}_2,$$

где $\vec{e}_1$, $\vec{e}_2$ – единичные векторы (орты) в направлении координатных осей.

Следовательно, направление, противоположное градиентному, укажет путь, ведущий вниз вдоль наиболее крутой линии спуска.

Идея метода градиентного спуска состоит в следующем.

Выбираем некоторую начальную точку и вычисляем в ней градиент рассматриваемой функции. Делаем шаг в направлении, обратном градиентному. В результате приходим в точку, значение функции в которой обычно меньше первоначального. Если это условие не выполнено, то есть значение функции не изменилось или даже возросло, то нужно уменьшить шаг.

В новой точке процедуру повторяем: вычисляем градиент и снова делаем шаг в обратном к нему направлении. Процесс продолжается до получения наименьшего значения целевой функции. Момент окончания поиска наступает тогда, когда движение из полученной точки с любым шагом приводит к возрастанию значения целевой функции. Если в какой-либо точке внутри области grad = 0, то минимум целевой функции в ней достигнут.

В данном методе требуется вычислять на каждом шаге оптимизации градиент целевой функции $f(x_1, x_2, \ldots, x_n)$:

$$\operatorname{grad}\vec{f} = \left\{\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right\}.$$

При этом формулы для частных производных можно получить в явном виде лишь тогда, когда целевая функция задана аналитически. В противном случае эти производные могут быть вычислены с помощью численного дифференцирования:

$$\frac{\partial f}{\partial x_i} \approx \frac{1}{\Delta x_i}\left[f\left(x_1, \ldots, x_i + \Delta x_i, \ldots, x_n\right) - f\left(x_1, \ldots, x_i, \ldots, x_n\right)\right], \quad i = 1, 2, \ldots, n.$$

В данном методе градиент целевой функции вычисляется в каждой точке траектории спуска, увеличивая объем вычислений $\left(x^{n+1} = x^n - \delta n \operatorname{grad} \vec{f}\left(x^n\right)\right)$.

Метод наискорейшего спуска, являющийся модификацией градиентного спуска, уменьшает количество расчетов. Согласно этому методу, после определения в начальной точке направления, противоположного градиенту целевой функции, в этом направлении делают не один шаг, а двигаются до тех пор пока целевая функция убывает, достигая, таким образом, минимума в некоторой точке. В этой точке снова определяют направление спуска (с помощью градиента) и ищут новую точку минимума целевой функции и т. д. Поэтому спуск здесь происходит более крупными шагами, и градиент функции вычисляется в меньшем числе точек.

7.3.2. Задачи с ограничениями

Решение условных задач оптимизации (задач с ограничениями) более трудоемко по сравнению с задачами безусловной оптимизации, так как ограничения типа равенств или неравенств требуют их учета на каждом шаге оптимизации. Поэтому их решение стремятся свести к решению последовательности задач безусловной оптимизации.

На этом основан и метод **штрафных функций**.

Сущность метода состоит в следующем.

Пусть $f(x_1, x_2, \ldots, x_n)$ – целевая функция, для которой нужно найти минимум m в ограниченной области D ($x_1, x_2, \ldots, x_n \in D$). Заменим данную задачу задачей о безусловной оптимизации (минимизации) однопараметрического семейства функций

$$F(x,\beta) = f(x) + \frac{1}{\beta}\varphi(x), \qquad x = \{x_1, x_2, ..., x_n\}. \tag{100}$$

При этом **дополнительную (штрафную) функцию φ(x)** выберем таким образом, чтобы при $\beta \to 0$ решение вспомогательной задачи стремилось к решению исходной или, по крайней мере, чтобы их минимумы совпадали: $\min F(x, \beta) \to m$ при $\beta \to 0$.

Штрафная функция φ(x) должна учитывать ограничения, которые задаются при постановке задачи оптимизации. В частности, если имеются ограничения-неравенства вида $g_j(x_1, x_2, \ldots, x_n) \geq 0$ ($j = 1, 2, \ldots, J$), то в качестве штрафной можно взять функцию, которая: 1) равна нулю во всех точках пространства проектирования, удовлетворяющих заданным ограничениям-неравенствам; 2) стремится к бесконечности в тех точках, в которых эти неравенства не выполняются. Таким образом, при выполнении ограничений-неравенств функции f(x) и F(x, β) имеют один и тот же минимум.

Если хотя бы одно неравенство не выполнится, то вспомогательная целевая функция F(x, β) получает бесконечно большие добавки, и ее значения далеки от минимума функции f(x). Другими словами при несоблюдении ограничений-неравенств на целевую функцию налагается «штраф». Отсюда и термин «метод штрафных функций».

Рассмотрим случай, когда в задаче оптимизации заданы ограничения двух типов – равенства и неравенства:

$$g_i(x) = 0, \quad i = 1, 2, \ldots, I;$$
$$h_j(x) > 0, \quad j = 1, 2, \ldots, J; \qquad x = \{x_1, x_2, \ldots, x_n\}. \qquad (101)$$

В этом случае в качестве вспомогательной целевой функции, для которой формируется задача безусловной оптимизации во всем n-мерном пространстве, принимают функцию

$$F(x,\beta) = f(x) + \frac{1}{\beta}\left\{\sum_{i=1}^{I} g_i^2(x) + \sum_{j=1}^{J} h_j^2(x)\left[1 - \operatorname{sign} h_j(x)\right]\right\}, \ \beta > 0, \qquad (102)$$

где sign(x) – «сигнум», т. е. функция от действительных переменных, равная + 1 для положительных x, принимающая значение нуль при x = 0 и равная – 1 для отрицательных x.

Здесь взята такая штрафная функция, что при выполнении условий (101) она обращается в нуль. Если же эти условия нарушены (т. е. $g_i(x) \neq 0$, $h_j(x) < 0$ и $\operatorname{sign} h_j(x) = -1$), то штрафная функция положительна. Она увеличивает целевую функцию f(x) тем больше, чем больше нарушаются условия (101).

При малых значениях β вне области D функции F(x, β) сильно возрастает. Поэтому ее минимум может быть либо внутри D, либо снаружи вблизи границ этой области. В первом случае минимумы функций F(x, β) и f(x) совпадают, так как дополнительные члены в (102) равны нулю. Если минимум функции F(x, β) находится вне D, то минимум целевой функции f(x) лежит на границе D. При этом можно построить такую последовательность $\beta_k \to 0$, что соответствующая последовательность минимумов функции F(x, β) будет стремиться к минимуму функции f(x).

Таким образом, задача оптимизации для целевой функции f(x) с ограничениями (101) свелась к последовательности задач безусловной оптимизации для вспомогательной функции (102), решение которых может быть проведено с помощью методов спуска. При этом строится итерационный процесс при $\beta \to 0$.

Рассмотрим алгоритм, реализующий метод штрафных функций (рис. 7.9).

В качестве исходных данных вводятся начальное приближение искомого вектора $x^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$, начальные значения параметра β и ε, характеризующие точность расчета.

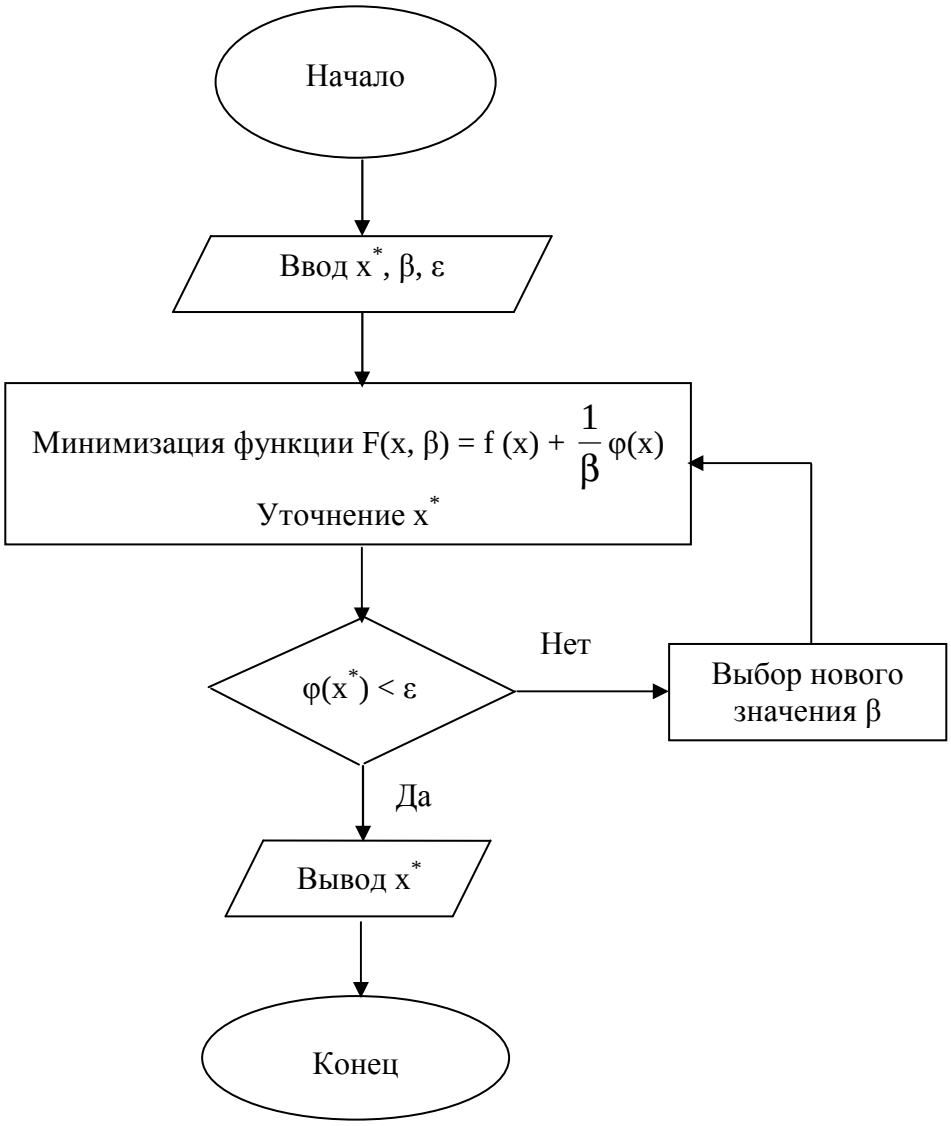


Рис. 7.9. Алгоритм метода штрафных функций

На каждом шаге итерационного процесса определяется оптимальное значение x^* вектора x, при этом в качестве начального приближения принимается результат предыдущей итерации.

Значение параметра β каждый раз уменьшается до тех пор, пока значение штрафной функции не станет заданной малой величиной. В этом случае точка x^* достаточна близка к границе области D и с необходимой точностью описывает оптимальные значения проектных параметров. Если точка минимума находится внутри области D, то искомый результат будет получен сразу после первого шага, так как в данном случае $\varphi(x^*) = 0$.

Линейное программирование изучает задачи оптимизации, в которых целевая функция является линейной функцией проектных параметров, а ограничения задаются в виде линейных уравнений и неравенств.

Стандартная постановка задачи линейного программирования формулируется следующим образом:

Найти значения переменных $x_1, x_2, \dots x_n$, которые:

1) удовлетворяют системе линейного уравнения

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2, \\ - - - - - - - - - - - - - - - - - - - \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n; \end{cases} \quad (103)$$

2) являются неотрицательными, т. е

$$x_1 \geq 0, \quad x_2 \geq 0, \quad \dots, \quad x_n \geq 0; \quad (104)$$

3) обеспечивают наименьшее значение линейной целевой функции

$$f(x_1, x_2, \dots, x_n) = c_0 + c_1x_1 + c_2x_2 + \dots + c_nx_n. \quad (105)$$

Всякое решение системы уравнений (103), удовлетворяющее системе неравенств (104), называется **допустимым решением**. Допустимое решение, которое минимизирует целевую функцию (105), называется **оптимальным решением**.

Задачи линейного программирования можно решать геометрическим методом.

Введем некоторые понятия.

Областью решения линейного неравенства с двумя переменными

$$a_0 + a_1x_1 + a_2x_2 \geq 0 \quad (106)$$

является полуплоскость.

Для того, чтобы определить, какая из двух полуплоскостей соответствует этому неравенству, нужно привести его к виду $x_2 \geq kx_1 + b$ или $x_2 \leq kx_1 + b$. Тогда искомая полуплоскость в первом случае расположена выше прямой $a_0 + a_1x_1 + a_2x_2 = 0$, во втором – ниже нее.

Если $a_2 = 0$, то неравенство (106) имеет вид $a_0 + a_1x_1 \geq 0$; в этом случае получим либо $x_1 \geq h$ – правую полуплоскость, либо $x_1 \leq h$ – левую полуплоскость.

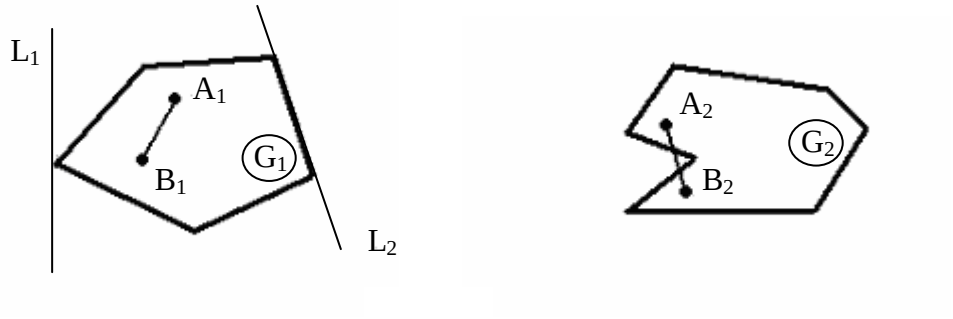
Областью решений системы неравенств двух переменных является пересечение конечного числа полуплоскостей, каждая из которых описывается отдельным неравенством.

Это пересечение представляет собой многоугольную область G. Она может быть как ограниченной, так и неограниченной и даже пустой (если система неравенств противоречива).

Область решений G обладает важным свойством выпуклости.

Область G_1 называется **выпуклой** (рис. 7.10, а), если две ее произвольные точки можно соединить отрезком A_1B_1, целиком принадлежащим данной области.

В **невыпуклой** области G_2 можно выбрать такие две ее точки, что не все точки отрезка A_2B_2 принадлежат области G_2 (рис. 7.10, б).



а) б)

Рис. 7.10. Выпуклая (а) и невыпуклая (б) области решений

Опорной прямой называется прямая, которая имеет с областью решений, по крайней мере, одну общую точку, при этом вся область расположена по одну сторону от этой прямой (прямые ℓ_1 и ℓ_2 на рис. 7.10, а).

При **геометрической интерпретации системы неравенств с тремя переменными** каждое неравенство описывает полупространство, а вся система – пересечение полупространств, т. е. многогранник, который также обладает свойством выпуклости. Здесь опорная плоскость проходит через вершину, ребро или грань многогранной области.

Основываясь на введенных понятиях, рассмотрим **геометрический метод** решения задачи линейного программирования.

Пусть задана линейная целевая функция $f = c_0 + c_1x_1 + c_2x_2$ двух независимых переменных, а также некоторая совместная система линейных неравенств, описывающих область решений G.

Требуется среди допустимых решений (x_1, x_2) є G найти такое, при котором линейная целевая функция f принимает наименьшее значение.

Положим функцию f равной некоторому постоянному значению c: $f = c_0 + c_1x_1 + c_2x_2 = c$. Это значение достигается в точках прямой, удовлетворяющих уравнению

$$c_0 + c_1x_1 + c_2x_2 = c. \qquad (107)$$

При параллельном переносе этой прямой в положительном направлении вектора нормали $\vec{n}(c_1, c_2)$ линейная функция f будет возрастать, а при переносе прямой в противоположном направлении – убывать.

Предположим, что прямая, записанная в виде (107), при параллельном переносе в положительном направлении вектора $\vec{n}$ первый раз встретится с областью допустимых решений G в некоторой ее вершине, при этом значение целевой функции равно c_1, и прямая становится опорной. Тогда значение c_1 будет минимальным, поскольку дальнейшее движение прямой в том же направлении приведет к увеличению функции f.

Если нас интересует максимум целевой функции, то параллельный перенос прямой (107) осуществляется в направлении, противоположном $\vec{n}$, пока она не станет опорной. Тогда вершина многоугольника G, через которую проходит прямая, будет соответствовать максимуму функции f. При дальнейшем переносе прямой целевая функция будет убывать.

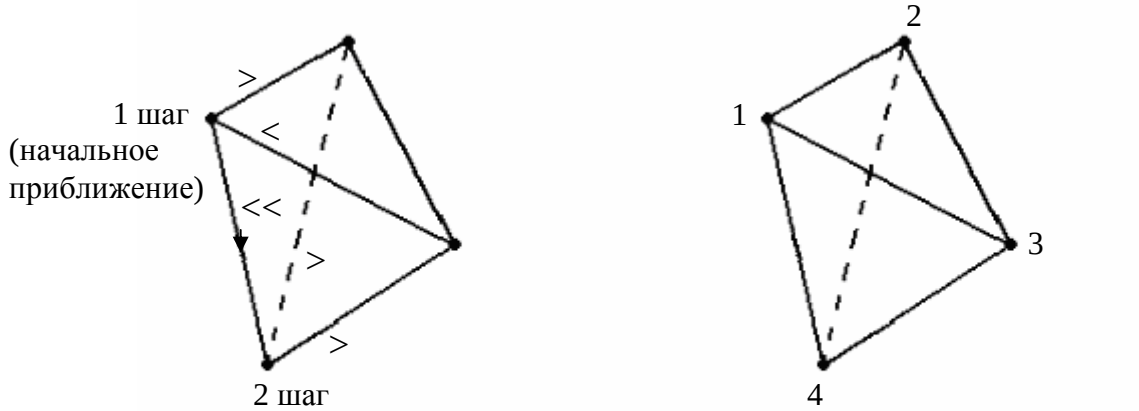
Таким образом, оптимизация линейной целевой функции на многоугольнике допустимых решений происходит в точках пересечения этого многоугольника с опорными прямыми, соответствующими данной целевой функции. При этом пересечение может быть в одной точке (в вершине многоугольника) или в бесконечном множестве точек (на ребре многоугольника).

Рассмотренный геометрический метод решения задач линейного программирования достаточно прост и нагляден для случая двух или трех переменных. Для большего числа переменных его применение становится невозможным из-за большого объема вычислений, связанных с перебором вершин и вычислениями в них значений целевой функции.

Поэтому для эффективного метода решений сложных задач линейного программирования с гораздо меньшим числом операций используют **симплекс-метод**.

Симплексом (от лат. «простой») называется выпуклая оболочка m, состоящая из n + 1 точек n-мерного метрического пространства: нольмерный симплекс – точка, одномерный – отрезок, двумерный – треугольник, трехмерный – тетраэдр.

Идея симплекс-метода заключается в следующем (рис. 7.11, а).



а) б)

Рис. 7.11. Геометрическая интерпретация решения задач линейного программирования симплекс-методом (а) и методом перебора (б)

Примем в качестве начального приближения координаты некоторой вершины многогранника допустимых решений и найдем все ребра, выходящие из этой вершины. Двигаемся вдоль того ребра, по которому линейная целевая функция убывает. Приходим в новую вершину, находим все выходящие из нее ребра, двигаемся по одному из них и т. д. В конце концов мы придем в такую вершину, движение из которой вдоль любого ребра приводит к возрастанию целевой функции. Следовательно, минимум достигнут, и координаты этой последней вершины принимаются в качестве оптимальных значений рассматриваемых проектных параметров.

Поскольку f – линейная функция, а многогранник выпуклый, то данный вычислительный процесс сходится к решению задачи, причем за конечное число шагов k. В данном случае их число имеет порядок n, т. е. значительно меньше количества шагов в методе простого перебора вершин, где k может быть порядка 2^n (рис. 7.11, б).

Пусть нужно найти такие неотрицательные значения проектных параметров $x_1, x_2, \ldots, x_n$, которые минимизируют линейную целевую функцию

$$f = c_0 + c_1x_1 + c_2x_2 + \ldots + c_nx_n \qquad (108)$$

при заданных ограничениях в виде системы линейных уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{1n}x_n = b_1, \\ \text{----------}, \\ a_{m1}x_1 + a_{m2}x_2 + a_{mn}x_n = b_m. \end{cases} \qquad (109)$$

Если в качестве ограничений заданы неравенства, то их можно преобразовать в равенства путем введения дополнительных неотрицательных переменных, называемых **балансовыми**.

Например, имея некоторое неравенство $a_1x_1 + a_2x_2 + \ldots + a_nx_n \le b$, можно всегда ввести новую переменную $x_{n+1} > 0$, такую, что после ее прибавления к левой части данного неравенства получим равенство

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + x_{n+1} = b.$$

Будем считать, что все уравнения системы (109) линейно независимы, т. е. ни одно из них нельзя получить как линейную комбинацию других. В противном случае линейно зависимые уравнения можно отбросить. И, кроме того, эта система совместна, т. е. среди уравнений системы нет противоречивых.

При этих предположениях число уравнений m меньше числа переменных n, поскольку при m = n система (109) имеет единственное значение, что исключает всякую оптимизацию; при m > n не выполняется сделанные выше предположения.

Выразим первые m переменных $x_1, x_2, \ldots, x_m$ через остальные с помощью уравнений (109):

$$
\begin{aligned}
x_1 &= p_1 + q_{1,\,m+1} \cdot x_{m+1} + q_{1,\,m+2} \cdot x_{m+2} + \ldots + q_{1n} \cdot x_n, \\
x_2 &= p_2 + q_{2,\,m+1} \cdot x_{m+1} + q_{2,\,m+2} \cdot x_{m+2} + \ldots + q_{2n} \cdot x_n, \\
&\text{– –}, \\
x_m &= p_m + q_{m,\,m+1} \cdot x_{m+1} + q_{2,\,m+2} \cdot x_{m+2} + \ldots + q_{mn} \cdot x_n, \\
p_i &\geq 0, \quad i = 1, 2, \ldots, m.
\end{aligned}
\tag{110}
$$

Переменные $x_1, x_2, \ldots, x_m$ называются **базисными**, а **вектор** $\{x_1, x_2, \ldots, x_m\}$ – **базисным**; $x_{m+1}, x_{m+2}, \ldots, x_n$ – **свободные переменные**.

Используя соотношения (110), можно выразить линейную целевую функцию f (108) через свободные перемещенные:

$$f = d_0 + d_{m+1} \cdot x_{m+1} + \ldots + d_n x_n. \tag{111}$$

Процесс оптимизации начнем с некоторого начального (опорного) решения, например, при нулевых значениях свободных переменных. Тогда получим

$$x_1 = p_1, \ldots, x_m = p_m, x_{m+1} = 0, \ldots, x_n = 0. \tag{112}$$

При этом целевая функция (111) примет значение $f^{(0)} = d_0$.

Дальнейшее решение задачи симплекс-методом распадается на ряд этапов, заключающихся в том, что от одного решения нужно перейти к другому с таким условием, чтобы целевая функция не возрастала. Это достигается выбором нового базиса и значений свободных переменных.

Выясним, является ли опорное решение (112) оптимальным. Для этого проверим, можно ли уменьшить соответствующее этому решению значение целевой функции $f = d_0$, при изменение каждой свободной переменной.

Так как $x_i \geq 0$, то можно лишь увеличивать их значения.

Если коэффициенты $d_{m+1}, \ldots, d_n$ в формуле (111) неотрицательны, то при увеличении любой свободной переменной $x_{m+1}, \ldots, x_n$ целевая функция не может уменьшиться. В этом случае **решение** (112) окажется **оптимальным**.

Пусть теперь среди коэффициентов формулы (111) хотя бы один отрицательный, например $d_{m+1} < 0$. Это означает, что при увеличении переменной x_{m+1} целевая функция уменьшается по сравнению со значением d_0, соответствующим решению (112). Поэтому в качестве нового опорного выбирается решение при следующих значениях свободных параметров:

$$x_{m+1} = x^{(1)}_{m+1}, x_{m+2} = 0, \ldots, x_n = 0. \tag{113}$$

При этом базисные перемещенные, вычисляемые по (110), равны

$$x_i = p_i + q_{i,\,m+1} \cdot x^{(1)}_{m+1}, \; i = 1, 2, \ldots, m. \tag{114}$$

Если все коэффициенты $q_{i,\,m+1}$ неотрицательны, то x_{m+1} можно увеличивать неограниченно; в этом случае не существует оптимального решения задачи. Но такие случаи на практике не встречаются.

Обычно среди коэффициентов $q_{i,\,m+1}$ имеются отрицательные, что влечет за собой угрозу сделать некоторые базисные перемещенные x_i в (114) отри-

цательными из-за большого значения $x^{(1)}_{m+1}$. Следовательно, переменную x_{m+1} можно увеличивать лишь до тех пор, пока базисные переменные остаются неотрицательными. Это является **условием выбора $x^{(1)}_{m+1}$:**

$$p_i + q_{i,\, m+1} \cdot x^{(1)}_{m+1} \geq 0,\ i = 1, 2, \ldots, m. \qquad (115)$$

Среди всех отрицательных коэффициентов $q_{i,\, m+1}$ найдем наибольший по модулю. Пусть его значение равно Q, а соответствующее ему значение p_i равно P. Тогда из (115) получим максимально возможное значение переменной x_{m+1} на данном шаге оптимизации:

$$x^{(1)}_{m+1} = -\,P/Q\ (P > 0,\ Q < 0).$$

Новое опорное решение запишем в виде

$$\begin{gathered} x_1 = p_1 - \frac{P}{Q} q_{1,\, m+1}, \ldots, x_m = p_m - \frac{P}{Q} q_{m,\, m+1}, \\ x_{m+1} = -\frac{P}{Q},\ x_{m+2} = 0, \ldots, x_n = 0\,. \end{gathered} \qquad (116)$$

Новая целевая функция при этих значениях проектных параметров равна

$$f^{(1)} = d_0 - d_{m+1} \frac{P}{Q}\,. \qquad (117)$$

Полученное значение целевой функции $f^{(1)}$ меньше предыдущего, поскольку в данной формуле второй член правой части больше нуля ($d_{m+1} < 0$, $Q < 0$, $P > 0$). На этом заканчивается первый шаг оптимизации.

Теперь нужно сделать второй шаг, используя аналогичную процедуру. Для этого необходимо выбрать новый базис, принимая в качестве базисных переменных параметры $x_1, \ldots, x_{m-1}, x_{m+1}$. После второго шага мы либо найдем новые оптимальные значения переменных и соответствующее им значение целевой функции $f^{(2)} < f^{(1)}$, либо покажем, что решение (116) является оптимальным. В любом случае после конечного числа шагов мы придем к оптимальному решению.

Таким образом, в отличие от метода перебора симплекс-метод дает возможность вести поиск целенаправленно, уменьшая на каждом шаге значение целевой функции.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Крупенников О. Г. Решение задач машиностроения средствами Turbo Pascal: учебное пособие / О. Г. Крупенников, С. И. Рязанов, Ю. В. Псигин, Д. В. Кравченко. – Ульяновск: УлГТУ, 2004. – 107 с.

2. Карев Е. А. Технологическая информатика: методические указания к выполнению курсовой работы для студентов специальности 120100 «Технология машиностроения» / Е. А. Карев. – Ульяновск: УлГТУ, 2002. – 52 с.

3. Турчак Л. И. Основы численных методов: учебное пособие / Л. И. Турчак, П. В. Плотников. – М.: ФИЗМАТЛИТ, 2002. – 304 с.

4. Корн Г. Справочник по математике (для научных работников и инженеров). Определения, термины, формулы / Г. Корн, Т. Корн. – СПб.: Издательство «Лань», 2003. – 832 с.

5. Фурунжиев Р. И. Применение математических методов и ЭВМ: учебное пособие для вузов / Р. И. Фурунжиев, Ф. М. Бабушкин, В. В. Варавко. – Минск: Выш. шк., 1988. – 191 с.

6. Мудров А. Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль: учебное пособие / А. Е. Мудров. – Томск: МП «РАСКО», 1991. – 272 с.

7. Фаронов В. В. Турбо Паскаль 7.0. Начальный курс: учебное пособие / В. В. Фаронов. – М.: «Нолидж», 1997. – 616 с.