Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Тепловые электрические станции»

ИНФОРМАТИКА

Лабораторный практикум
для студентов специальностей
1-43 01 04 «Тепловые электрические станции»,
1-53 01 04 «Автоматизация и управление энергетическими процессами», 1-43 01 08 «Паротурбинные установки атомных электрических станций»

В 2 частях

Ч а с т ь 2

Минск
БНТУ
2011

УДК 004(076.5) (075.8)
ББК 32.81 я 7
И 74

Составители:
Л.А. Тарасевич, *Е.В. Пронкевич*

Рецензенты:
В.А. Булат, *В.Б. Козловская*

И 74 Информатика: лабораторный практикум для студентов специальностей 1-43 01 04 «Тепловые электрические станции», 1-53 01 04 «Автоматизация и управление энергетическими процессами», 1-43 01 08 «Паротурбинные установки атомных электрических станций»: в 2 ч. – Ч. 2 / сост.: Л.А. Тарасевич, Е.В. Пронкевич. – Минск: БНТУ, 2010. – 86 с.

ISBN 978-985-525-654-1 (Ч. 2).

В лабораторном практикуме рассматриваются основы программирования на языке TURBO PASCAL 7.0 с использованием стандартного модуля CRT, основные приемы численных методов, относящиеся к решению нелинейных уравнений, систем нелинейных и линейных алгебраических уравнений, интегрированию функций, решению задач аппроксимации функций, обыкновенных дифференциальных уравнений. Значительное внимание уделяется вопросам алгоритмизации методов.

Часть 1 (авторы Л.А. Тарасевич, Е.В. Пронкевич, Ю.Б. Попова) издана в БНТУ в 2007 г.

ISBN 978-985-525-654-1 (Ч. 2)
ISBN 978-985-479-784-7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данном лабораторном практикуме приведены основы программирования на языке TURBO PASCAL 7.0 с использованием модуля CRT, основы численных методов решения нелинейных уравнений, систем линейных алгебраических уравнений, систем нелинейных алгебраических уравнений, вычисления определенных интегралов, обыкновенных дифференциальных уравнений 1-го порядка. Дополнительно рассматриваются интерполирование и аппроксимация.

Лабораторный практикум может быть использован при выполнении лабораторных и курсовых работ. Каждая лабораторная работа содержит теоретическое обоснование, блок-схемы рассматриваемых методов и упражнения для самостоятельной работы и завершается контрольными вопросами по данной теме.

Лабораторная работа № 1

СТАНДАРТНЫЙ МОДУЛЬ CRT

В стандартном модуле CRT сосредоточены процедуры и функции, обеспечивающие управление текстовым режимом работы экрана. С помощью входящих в модуль подпрограмм можно перемещать курсор в произвольную позицию экрана, менять цвет выводимых символов и окружающего их фона, создавать окна.

Кроме того, в модуль включены процедуры Sound, NoSound, Delay, которые позволяют программировать звуковой генератор ПК.

Программирование клавиатуры

Дополнительные возможности управления реализуются двумя функциями: KeyPressed и ReadKey.

Функция KeyPressed возвращает значение типа Boolean, указывающее состояние буфера клавиатуры: False означает, что буфер пуст, а True – что в буфере есть хотя бы один символ, еще не прочитанный программой.

В MS DOS реализуется так называемый асинхронный буферизованный ввод с клавиатуры. По мере нажатия на клавиши соответствующие коды помещаются в особый буфер, откуда они могут быть затем прочитаны программой. Стандартная длина буфера рассчитана на хранение до 16 кодов символов. Если программа достаточно долго не обращается к клавиатуре, а пользователь нажимает клавиши, буфер может оказаться переполненным. В этот момент раздается звуковой сигнал и «лишние» коды теряются. Чтение из буфера обеспечивается процедурами Read / Readln и функцией ReadKey. Обращение к функции KeyPressed не задерживает исполнение программы: функция немедленно анализирует буфер и возвращает то или иное значение, не дожидаясь нажатия клавиши.

Функция ReadKey возвращает значение типа Char. При обращении к этой функции анализируется буфер клавиатуры: если в нем есть хотя бы один непрочитанный символ, код этого символа берется из буфера и возвращается в качестве значения функции, в противном случае функция будет ожидать нажатия на любую клавишу.

Пусть, например, в какой-то точке программы необходимо игнорировать все ранее нажатые клавиши, коды которых еще не прочитаны из буфера, т. е. необходимо очистить буфер.

Этого можно достичь следующим образом:

```
Uses CRT;
Var C: Char;
Begin
While KeyPressed do
C:=ReadKey;
. . .
End.
```

При использовании процедуры ReadKey необходимо учесть, что в клавиатурный буфер помещаются так называемые расширенные коды нажатых клавиш. Если нажимается любая алфавитно-цифровая клавиша, расширенный код совпадает с ASCII кодом соответствующего символа. Например, если нажимается клавиша с латинской буквой «a» (в нижнем регистре), функция ReadKey возвращает значение chr(97), а если «A» (в верхнем регистре) – значение chr(65). При нажатии функциональных клавиш F1–F10, клавиш управления курсором, клавиш Ins, Home, Del, End, PgUp, PgDn в буфер помещается двухбайтная последовательность: сначала символ # 0, а затем расширенный код клавиши. Таким образом, значение # 0, возвращаемое функцией ReadKey, используется исключительно для того, чтобы указать программе на генерацию расширенного кода. Получив это значение, программа должна еще раз обратиться к функции, чтобы прочитать расширенный код клавиши, т. е. код сканирования клавиши. (Этот код определяется порядком, в соответствии с которым микропроцессор клавиатуры Intel 8042 периодически опрашивает (сканирует) состояние клавиш).

Следующая программа позволяет определить расширенный код любой клавиши:

```
Uses CRT;
   Var
     C:Char;
   Begin
```

```
    repeat
    C:=ReadKey;
    If C<># 0 then
    WriteIn (ord (C))
  else
    WriteIn ('0', ord (ReadKey):8)
    until C=# 27 {27 – расширенный код клавиши Esc}
End.
```

Для завершения работы программы нужно нажать клавишу Esc.

Если воспользоваться этой программой, то обнаружится, что нажатие на некоторые клавиши игнорируется функцией ReadKey. Это, прежде всего, так называемые сдвиговые клавиши – Shift, Ctrl, Alt. Сдвиговые клавиши в MS DOS обычно используются для переключения регистров клавиатуры и нажимаются в сочетании с другими клавишами. Именно таким способом, например, различается ввод прописных и строчных букв.

Текстовой вывод на экран

Используемое в ПК устройство визуального отображения информации – дисплей – состоит из двух основных частей: монитора, содержащего экран с необходимыми компонентами (устройствами развертки изображения), и блока управления, который чаще называют адаптером. Обычно оба устройства согласуются друг с другом, но в отдельных случаях этого согласования может не быть (например, цветной монитор может работать с монохромным адаптером, и наоборот). Здесь будем считать оба устройства согласованными, поэтому, говоря о различных дисплеях, будем говорить только о различных адаптерах, так как именно в них сосредоточены основные отличия дисплеев друг от друга.

1. Процедура TextMode используется для задания одного из возможных текстовых режимов работы адаптера. Заголовок процедуры:

```
Procedure TextMode (Mode: Word);
```

Здесь Mode – код текстового режима. В качестве значения этого выражения могут использоваться следующие константы, определенные в модуле CRT:

```
const
    BW40=0; {Черно-белый режим 40x25}
    Co40=1; {Цветной режим 40x25}
    BW80=2; {Черно-белый режим 80x25}
    Co80=3; {Цветной режим 80x25}
    Mono=7; {Используется с MDA}
    Font8x8=256; {Используется для загружаемого шрифта
                  в режиме 80x43 или 80x50 с адаптерами EGA
                  или VGA}
```

Код режима, установленного с помощью процедуры TextMode, запоминается в глобальной переменной LastMode модуля CRT и может использоваться для восстановления начального состояния экрана.

Следующая программа иллюстрирует использование этой процедуры в различных режимах. Отметим, что при вызове TextMode сбрасываются все ранее сделанные установки цвета и окон, экран очищается и курсор переводится в его левый верхний угол.

```
Uses CRT;
Procedure Print (S: String);
{Выводит сообщение S и ждет инициативы пользователя}
begin
    WriteLn (S); {Выводим сообщение}
    WriteLn ('Нажмите клавишу Enter…');
    ReadLn {Ждем нажатия клавиши Enter}
end; {Print}
var
    LM:Word; {Начальный режим экрана}
Begin
    LM:=LastMode; {Запоминаем начальный режим работы
                   дисплея}
TextMode (Co40);
Print ('Режим 40x25');
TextMode (Co80);
Writeln ('Режим 80x25');
TextMode (Co40+Font8x8);
Writeln ('Режим Co40+Font8x8');
TextMode (Co80+Font8x8);
```

```
Writeln ('Режим Co80+Font8x8'); {Восстанавливаем исходный
                                 режим работы}
TextMode (LM);
End.
```

2. Процедура TextColor определяет цвет выводимых символов. Заголовок процедуры:

```
Procedure TextColor (Color: Byte);
```

Параметр Color определяет цвет выводимого символа, представляет собой выражение целого типа (табл. 1.1).

3. ***Процедура TextBackGround*** определяет цвет фона. Заголовок:

```
Procedure TextBackGround (Color: Byte);
```

Параметр Color – выражение целого типа, обозначающее устанавливаемый цвет фона, на котором выводятся символы.

Таблица. 1.1

Константы, соответствующие различным цветам

Цвет символов	Цвет фона	Название	Обозначение
Черный	Черный	Black	0
Синий	Синий	Blue	1
Зеленый	Зеленый	Green	2
Бирюзовый	Бирюзовый	Cyan	3
Красный	Красный	Red	4
Сиреневый	Сиреневый	Magenta	5
Коричневый	Коричневый	Brown	6
Белый	Светло-серый	LightGray	7
Серый		Gray	8
Голубой		LightBlue	9
Светло-зеленый		LightGreen	10
Светло-бирюзовый		LightCyan	11
Светло-красный		LightRed	12
Светло-сиреневый		LightMagenta	13
Желтый		Yellow	14
Ярко-белый		White	15
Мерцание символа		Blink	128

*4. **Процедура ClrScr*** очищает экран или окно. После обращения к ней экран (окно) заполняется цветом фона и курсор устанавливается в его левый верхний угол. Например:

```
Uses CRT;
var
    C: Char;
Begin
    TextBackGround (red); {Заполняем экран красным цветом}
    ClrScr;
    WriteLn ('Нажмите любую клавишу…');
    C:=ReadKey; {Ждем нажатия любой клавиши}
    TextBackGround (Black);
    ClrScr {Восстанавливаем черный фон экрана}
End.
```

*5. **Процедура Window*** определяет текстовое окно – область экрана, которая в дальнейшем будет рассматриваться процедурами вывода как весь экран. Сразу после вызова процедуры курсор помещается в левый верхний угол окна, а само окно очищается (заполняется цветом фона). По мере вывода курсор, как обычно, смещается вправо и при достижении правой границы окна переходит на новую строку, а если он к этому моменту находился на последней строке, содержимое окна сдвигается вверх на одну строку, то есть осуществляется «прокрутка» окна. Заголовок процедуры:

```
Procedure Window (X1, Y1, X2, Y2: Byte);
```

Здесь X1, Y1 – координаты левого верхнего угла окна; X2, Y2 – правого нижнего угла окна. Они задаются в координатах экрана, причем левый верхний угол экрана имеет координаты (1, 1), горизонтальная координата увеличивается слева направо, а вертикальная сверху вниз.

*6. **Процедура GotoXY*** переводит курсор в нужное место экрана или текущего окна. Заголовок процедуры:

```
Procedure GotoXY (X, Y: Byte);
```

Здесь X, Y – новые координаты курсора. X – столбец, Y – строка. Координаты задаются относительно границ экрана (окна), т. е. оператор

```
GotoXY (1, 1);
```

означает перевести курсор в левый верхний угол экрана (или окна, если к тому моменту на экране определено окно). Обращение к процедуре игнорируется, если новые координаты выходят за границы экрана (окна).

Рассмотрим процедуру черчения рамок.

```
Procedure Frame (X1, Y1, X2, Y2: integer);
{X1, Y1, X2, Y2 – координаты соответственно левого верхнего и правого нижнего угла рамки}
const
{для черчения двойной линии}
a=#186; b=#187; c=#188;
d=#200; e=#201; f=#205;
{для черчения одинарной линии}
a=#179; b=#191; c=#217;
d=#192; e=#218; f=#198;

var i,j:integer;
Begin
    GotoXY(X1,Y1);
    write(e);
    for i:=(X1+1) to (X2-1) do write(f);
    write(b);
    for i:=(Y1+1) to (Y2-1) do
    begin
        GotoXY(X1,i);
        write(a);
        GotoXY(X2,i);
        write(a);
    end;
    GotoXY(X1,Y2);
    write(d);
```

```
    for i:=(X1+1) to (X2-1) do
    write(f);
    write(c);
End.
```

Рассмотрим процедуру выделения ограниченного рамкой окна с заданным фоном.

```
Procedure FrameFon(V1,V2,V3,V4,Fon:Byte);
Begin
    Frame (V1,V2,V3,V4);
    Window(V1+1, V2+1, V3-1, V4-1);
    TextBackGround(Fon);
    ClrScr;
End.
```

7. ***Функции*** *WhereX* ***и*** *WhereY*. С помощью этих функций типа Byte можно определить текущие координаты курсора: WhereX возвращает его горизонтальную, а WhereY – вертикальную координаты.

Три следующие процедуры без параметров могут оказаться полезными при разработке текстовых редакторов.

8. Процедура ClrEOL стирает часть строки от текущего положения курсора до правой границы окна (экрана). Положение курсора не меняется.

9. Процедура DelLine уничтожает всю строку с курсором в текущем окне (или на экране, если окно не создано). При этом все строки ниже удаляемой (если они есть) сдвигаются вверх на одну строку.

10. Процедура InsLine вставляет строку: строка с курсором и все строки ниже ее сдвигаются вниз на одну строку. Строка, вышедшая за нижнюю границу окна (экрана), безвозвратно теряется. Текущее положение курсора не меняется.

11. ***Процедуры*** *LowVideo, NormVideo* ***и*** *HighVideo.* С помощью этих процедур без параметров можно устанавливать соответственно пониженную, нормальную и повышенную яркость символов. Например:

```
Uses CRT;
Begin
```

```
    LowVideo;
    WriteLn ('Пониженная яркость');
    NormVideo;
    WriteLn ('Нормальная яркость');
    HighVideo;
    WriteLn ('Повышенная яркость');
End.
```

На практике нет разницы между пониженной и нормальной яркостью изображения.

12. Процедура AssignCRT. Связывает текстовую файловую переменную F с экраном с помощью непосредственного обращения к видеопамяти (т. е. к памяти, используемой адаптером для создания изображения на экране). В результате вывод в такой текстовой файл осуществляется значительно (в 3–5 раз) быстрее, чем если бы этот файл был связан с экраном стандартной процедурой Assign. Заголовок процедуры:

```
Procedure AssignCRT (F: Text);
```

Программирование звукового генератора

В модуль CRT включены три процедуры, с помощью которых можно запрограммировать произвольную последовательность звуков.

1. Процедура Sound заставляет динамик звучать с нужной частотой. Заголовок процедуры:

```
Procedure Sound (F: Word);
```

Здесь F – выражение типа Word, определяющее частоту звука в герцах.

После обращения к процедуре включается динамик и управление немедленно возвращается в основную программу, в то время как динамик будет звучать до вызова процедуры NoSound.

2. ***Процедура NoSound*** выключает динамик. Если он к этому моменту не был включен, вызов процедуры игнорируется.

3. ***Процедура Delay*** обеспечивает задержку работы программы на заданный интервал времени. Заголовок процедуры:

```
Procedure Delay (T: Word);
```

Здесь T – выражение типа Word, определяющее интервал времени (в милисекундах), в течение которого задерживается выполнение следующего оператора программы.

С помощью процедур Sound, Delay, NoSound и операторов цикла можно создать самые разнообразные звуковые эффекты. Для этого используется набор частот или элементы массива, соответствующие нотам различных октав (табл. 1.2).

Таблица. 1.2

Частота звучания нот

Нота	Большая октава	Малая октава	Первая октава	Вторая октава
До	130,81	261,63	523,25	1046,50
Ре	146,83	293,66	587,33	1174,07
Ми	164,81	329,63	659,26	1318,05
Фа	174,61	349,23	698,46	1396,09
Соль	196,00	392,00	784,99	1568,00
Ля	220,00	440,00	880,00	1760,00
Си	246,94	493,00	987,77	1975,00

Следующая программа воспроизводит простую музыкальную гамму. Используемый в ней массив F содержит частоты всех полутонов в первой октаве от «до» до «си». При переходе от одной октавы к соседней частоты изменяются в два раза.

```
Uses CRT;
const
F:array [1..12] of Real = (130.8, 138.6, 146.8, 155.6, 164.8, 174.6,
185.0, 196.0, 207.7, 220.0, 233.1, 246.9); {Массив частот первой октавы}
Temp=100; {Темп исполнения}
var
k, n:integer;
Begin
{Восходящая гамма}
```

```
for k:=1 to 3 do
for n:=1 to 12 do
begin
Sound (Round(F[n]*(1 shl k))); Delay (Temp);
NoSound
end;
{Нисходящая гамма}
for k:=3 downto 0 do
for n:=12 downto 1 to
begin
    Sound (Round (F[n]*(1 shl k)));
    Delay (Temp);
    NoSound
end
End.
```

Меню

Меню – это перечисление возможностей системы, из которого пользователь выбирает нужную в текущий момент.

Пример программы вертикального меню

Вертикальное меню в примере представляется следующим образом:

```
VVOD {Ввод данных}
POISK {Вычисление и вывод результата}
EXIT {Выход}
```

С помощью ↑ или ↓ выбирается режим и нажимается Enter для выполнения.

```
Program menu;
uses crt;
const
 pm=3; Dim=15;
var
```

```
  c:byte;
  Key:char;
  i:byte;
  prexit:Boolean;
  menupunkts:array [1..pm] of string;

   Procedure vvod;
    Begin
     writeln('Ввод данных');
    End;

   Procedure poisk;
    Begin
     writeln('Вычисление');
     readln;
    End;

   Procedure exit;
     Begin
      Exit;
     End;

BEGIN

  menupunkts[1]:='vvod';
  menupunkts[2]:='poisk';
  menupunkts[3]:='exit';

  c:=1;
  Prexit:=False;
 Repeat
  clrscr;
  GotoXY(1,5);

  For i:=1 to pm do
   Begin
    TextColor(8);
    if i=c then TextColor(5)
```

```
          else TextColor(8);
    writeln (menupunkts[i]);
   End;

   key:=Readkey;
  case ORD(key) of
   13:
     Begin
      clrscr;
        case c of
          1: vvod;
          2: poisk;
          3: prexit:=not prexit;
        End
     End;
    72:Dec(c);
    80:Inc(c);
   End;
    if c<1 then c:=c;
    if c>pm then c:=pm;
  until prexit;
END.
```

Лабораторная работа № 2

РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Пусть задана непрерывная функция $f(x)$ и требуется найти корни уравнения

$$f(x) = 0 \qquad (2.1)$$

на всей числовой оси или на некотором интервале $a < x < b$.

Всякое значение $x^* \in (a;b)$, удовлетворяющее условию $f(x^*) = 0$, называется ***корнем уравнения***, а способ нахождения этого значения x^* и есть ***решение уравнения*** (2.1).

Метод бисекции

Алгоритм решения нелинейного уравнения

1. $f(a) \cdot f(b) < 0$;

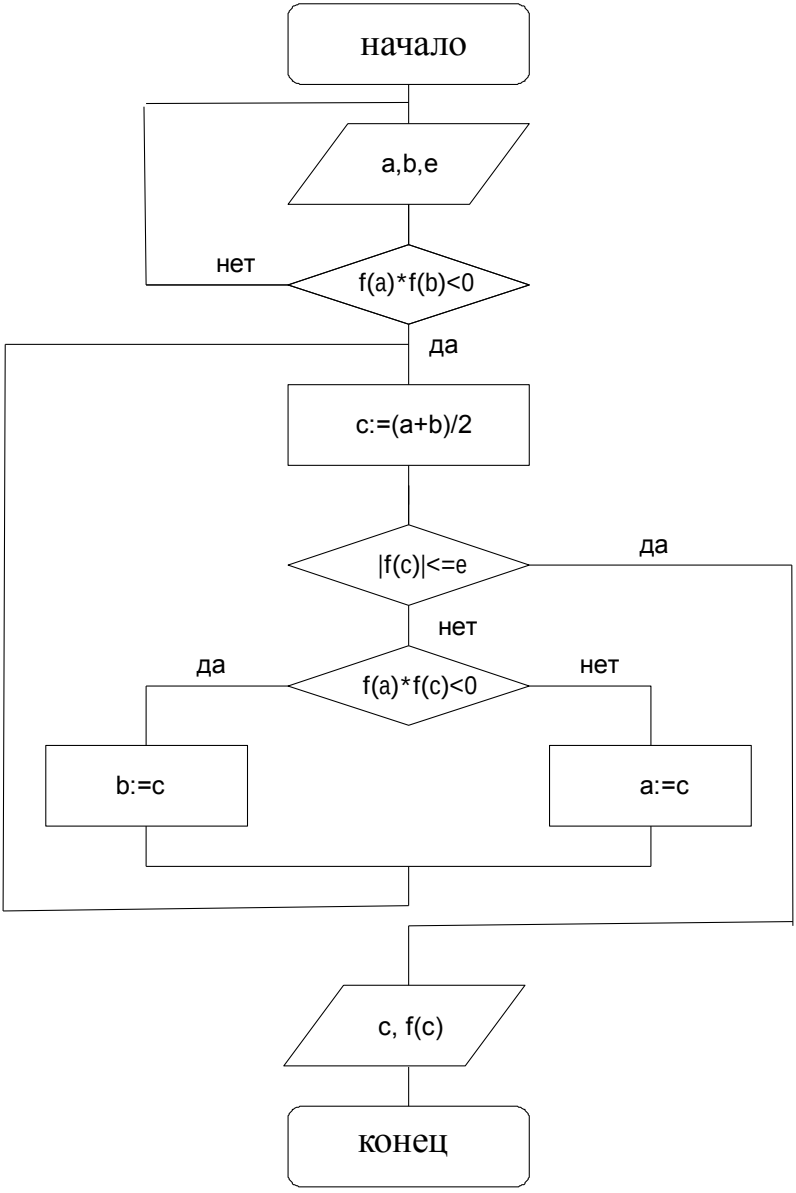
2. $c = \dfrac{a+b}{2}$;

3. Если $f(a) \cdot f(c) < 0$, то $b = c$.
   Если $f(b) \cdot f(c) < 0$, то $a = c$.

4. Критерий окончания счета:

$$\begin{cases} |f(c)| \le \varepsilon \\ |a-b| \le \varepsilon \end{cases}.$$

Блок-схема алгоритма решения нелинейных уравнений методом бисекции



Метод хорд

Алгоритм решения нелинейного уравнения

1. Если $f(a)\cdot f(b)<0$, то на отрезке $[a;b]$ существует корень.

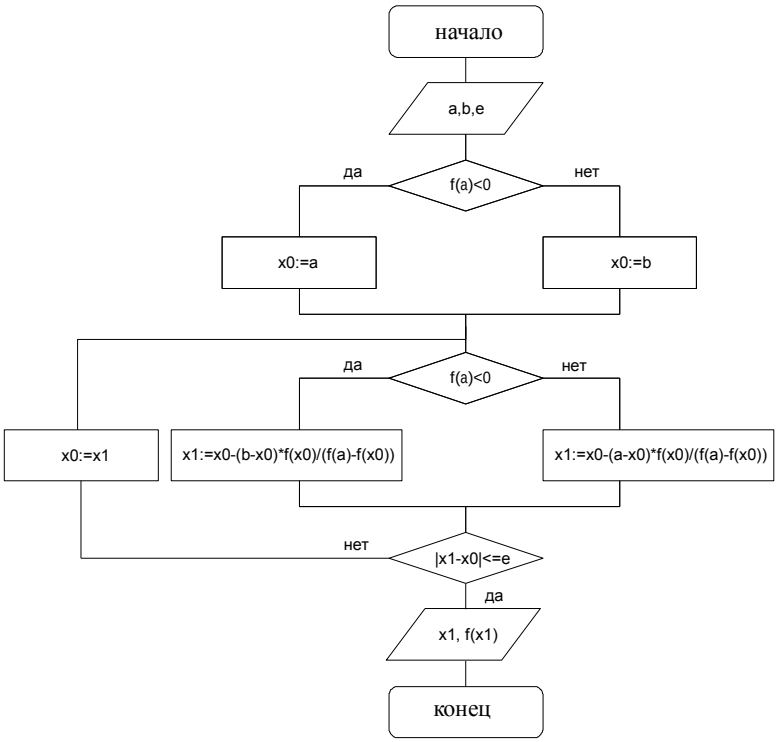
2. Если $f(a)<0$, то $x_0=a$ и $x_{n+1}=x_n-\dfrac{b-x_n}{f(b)-f(x_n)}\cdot f(x_n)$, $n=0,1,\ldots$

Если $f(a)>0$, то $x_0=b$ и $x_{n+1}=x_n-\dfrac{a-x_n}{f(a)-f(x_n)}\cdot f(x_n)$, $n=0,1,\ldots$

3. Критерий окончания счета:

$$\begin{cases}|f(x_{n+1})|\le\varepsilon\\ |x_{n+1}-x_n|\le\varepsilon\end{cases}.$$

Блок-схема алгоритма решения нелинейных уравнений методом хорд



Метод простой итерации

Постановка задачи

Дано нелинейное уравнение

$$f(x) = 0. \qquad (2.2)$$

Корень отделен $x^* \in [a; b]$. Требуется уточнить корень с точностью ε.

Уравнение (2.2) преобразуем к эквивалентному виду:

$$x = \varphi(x). \qquad (2.3)$$

Выберем начальное приближение $x_0 \in [a; b]$.

Вычислим последовательно новые приближения корня:

$$x_1 = \varphi(x_0);$$

$$x_2 = \varphi(x_1); \qquad (2.4)$$

$$\ldots \ldots \ldots$$

$$x_i = \varphi(x_{i-1}),$$

где i – номер итерации; $i = 1, 2, \ldots$.

Последовательное вычисление значений x_i по формуле (2.4) называется итерационным процессом метода простых итераций, а сама формула – формулой итерационного процесса метода.

Если $\lim\limits_{i \to \infty} x_i = x^*$, то итерационный процесс сходящийся.

Условие сходимости:

$$\left|\varphi'_{(x)}\right| < 1 \quad \forall x \in [a; b]. \qquad (2.5)$$

Точное решение x^* получить невозможно, так как требуется бесконечный итерационный процесс.

Итерационный процесс заканчивается при выполнении условия

$$\left|x_i - x_{i-1}\right| \le \varepsilon, \qquad (2.6)$$

где ε – заданная точность; i – номер последней итерации.

Условие завершения итерационного процесса (2.6) обеспечивает близость значения x_i к точному решению:

$$\left|x^* - x_i\right| \le \varepsilon. \qquad (2.7)$$

Рассмотрим геометрическую иллюстрацию метода простых итераций.

Уравнение (2.3) представим на графике в виде двух функций: $y_1 = x$ и $y_2 = \varphi(x)$.

Возможные случаи взаимного расположения графиков функций и, соответственно, видов итерационного процесса показаны на рис. 1.1–1.4.

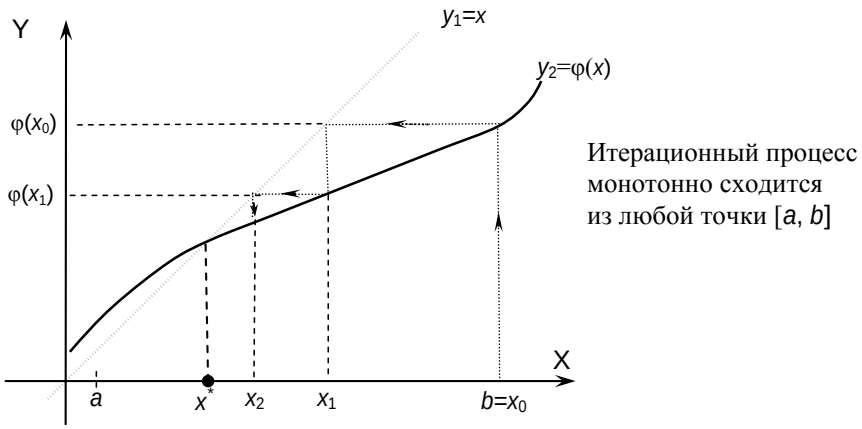
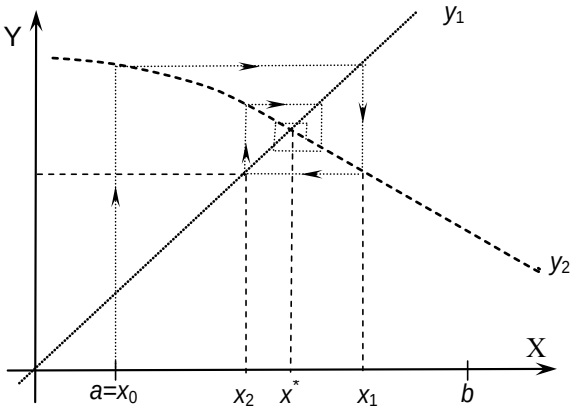
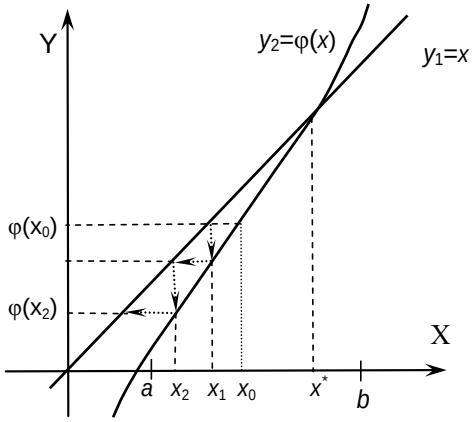


Рис. 1.1. Итерационный процесс для случая $0 < \varphi'_x < 1 \ \ \forall \ \ x \in [a; b]$



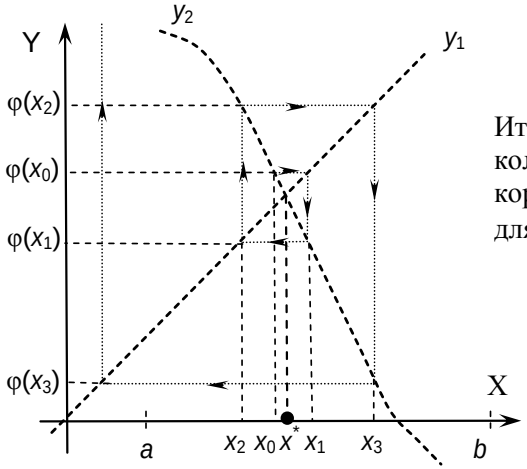
Итерационный процесс колебательно (около корня x^*) сходится из любой точки $[a, b]$

Рис. 1.2. Итерационный процесс для случая $-1 < \varphi'_x < 1 \quad \forall \quad x \in [a; b]$



Итерационный процесс монотонно расходится для любого $x_0 \in [a, b]$

Рис. 1.3. Итерационный процесс для случая $\varphi'_x > 1 \quad \forall \quad x \in [a; b]$



Итерационный процесс колебательно (относительно корня x^*) расходится для любого $x_0 \in [a; b]$

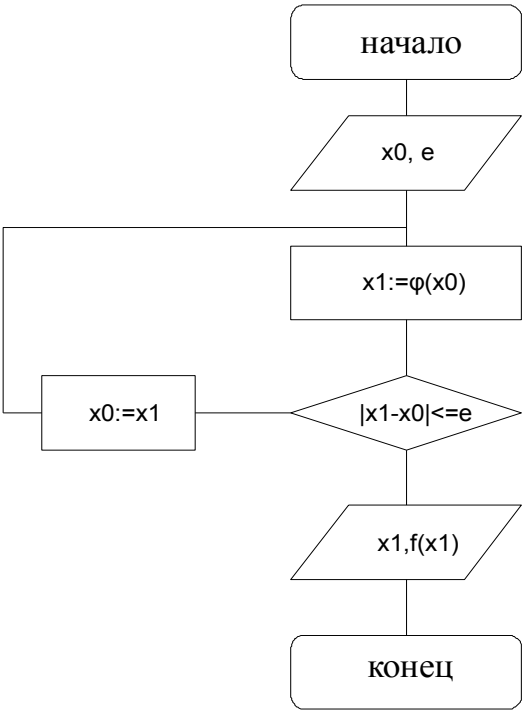
Рис. 1.4. Итерационный процесс для случая $\varphi'_x \le -1 \ \ \forall \ \ x \in [a; b]$

Из анализа графиков следует, что скорость сходимости растет при уменьшении значения $|\varphi'_x|$.

Алгоритм нахождения корня методом простых итераций

1. Если $\varphi'(x) > 0$, то $x_0 = b$.
   Если $\varphi'(x) < 0$, то $x_0 = a$.
2. Уравнение $f(x) = 0$ преобразуем к виду:
   $x_{n+1} = \varphi(x_n)$
   или
   $x_{n+1} = x_n + c \cdot f(x_n)$,
   где $x_n + c \cdot f(x_n) = \varphi(x_n)$,
3. Условие сходимости метода простой итерации:
   $|\varphi'(x_0)| < 1$.
4. Критерий окончания счета:
   $$\begin{cases} |f(x_{n+1})| \le \varepsilon \\ |x_{n+1} - x_n| \le \varepsilon \end{cases}.$$

Блок-схема алгоритма решения нелинейных уравнений методом простой итерации



Метод Ньютона (метод касательных)

Постановка задачи

Дано нелинейное уравнение (2.8):

$$f(x) = 0\,. \qquad (2.8)$$

Корень отделен $x^* \in [a; b]$.

Метод основан на стратегии постепенного уточнения корня. Формулу уточнения можно получить из геометрической иллюстрации идеи метода (рис. 1.5).

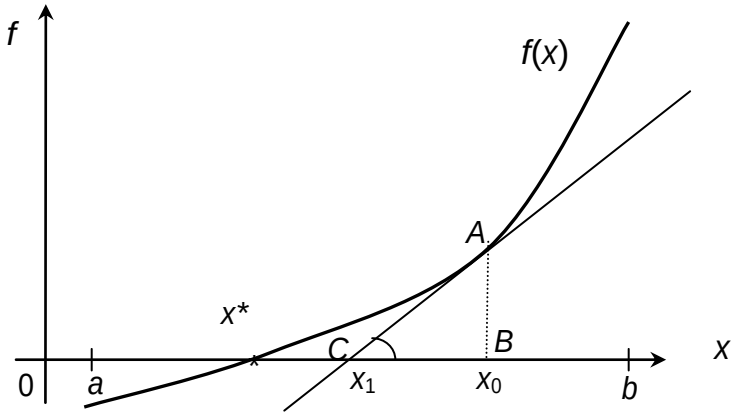


Рис. 1.5. Геометрическая иллюстрация метода Ньютона

На отрезке существования корня выбирается начальное приближение x_0. К кривой $f(x)$ в точке A с координатами $(x_0, f(x_0))$ проводится касательная. Абсцисса x_1 точки пересечения этой касательной с осью OX является новым приближением корня.

Из рисунка следует, что $x_1 = x_0 - CB$.

Из ΔABC: $CD = \dfrac{AB}{\mathrm{tg}\angle ACB}$. Но $\mathrm{tg}\angle ACB = f'(x_0)$, $AB = f(x_0)$.

Следовательно, $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$

Аналогично для i-го приближения можно записать формулу итерационного процесса метода Ньютона:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}, \qquad (2.9)$$

где $x_0 \in [a; b]$; $i = 1, 2, \ldots$.

Условие окончания расчета:

$$|\delta| \le \varepsilon, \qquad (2.10)$$

где $\delta = x_{i-1} - x_i = \dfrac{f(x_{i-1})}{f'(x_{i-1})}$ – корректирующее приращение или поправка.

Условие сходимости итерационного процесса:

$$\left| f(x) \cdot f''(x) \right| < \left(f'(x) \right)^2 \; \forall x \in [a;b]. \qquad (2.11)$$

Если на отрезке существования корня знаки $f'(x)$ и $f''(x)$ не изменяются, то начальное приближение, обеспечивающее сходимость, нужно выбрать из условия:

$$f(x_0) \cdot f''(x_0) > 0, \; x_0 \in [a; b]. \qquad (2.12)$$

Т. е. в точке начального приближения знаки функций и ее второй производной должны совпадать (рис. 1.6).

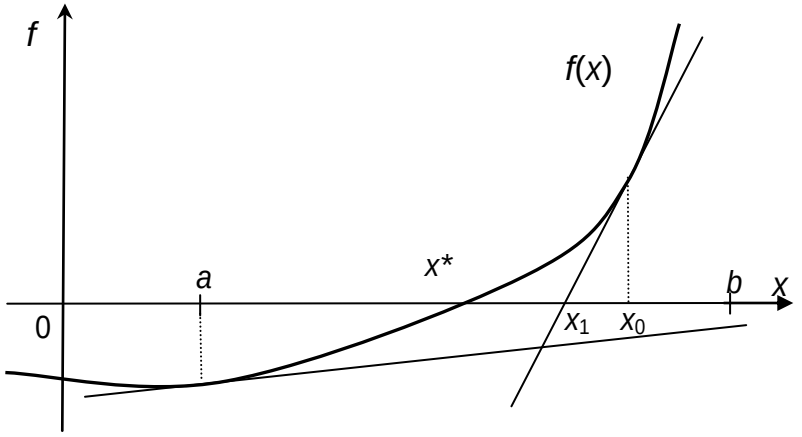


Рис. 1.6. Геометрическая иллюстрация выбора начального приближения: график $f(x)$ вогнутый, $f''(x) > 0$, тогда $x_0 = b$, т. к. $f(b) > 0$

Если же выбрать $x_0 = a$, то итерационный процесс будет сходиться медленнее или даже расходиться (см. касательную для $x_0 = a$ на рис. 1.7).

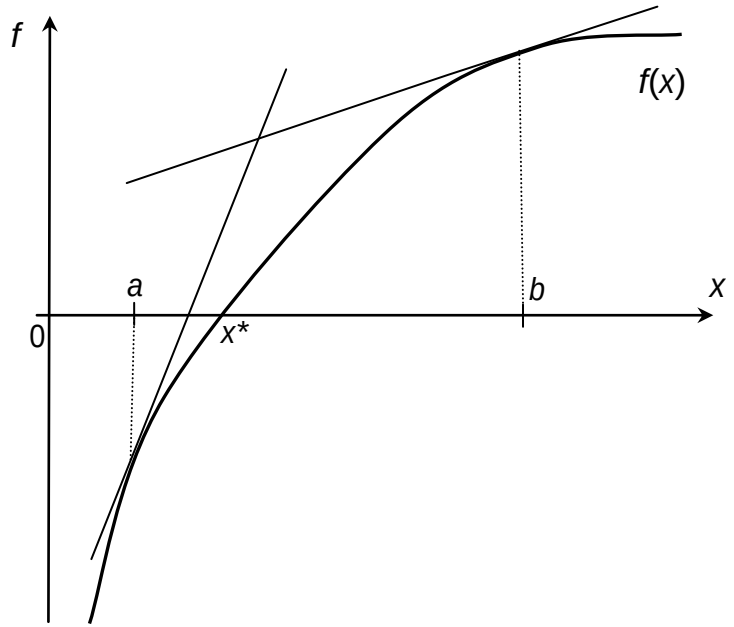


Рис. 1.7. Геометрическая иллюстрация выбора начального приближения: график *f(x)* выпуклый, $f''(x) < 0$, тогда $x_0 = a$, т. к. $f(a) < 0$

Метод Ньютона в отличие от ранее рассмотренных методов использует свойства функции в виде значения производной, что значительно ускоряет итерационный процесс. При этом, чем больше значение модуля производной в окрестности корня (чем круче график функции), тем быстрее сходимость.

Алгоритм нахождения корня методом Ньютона

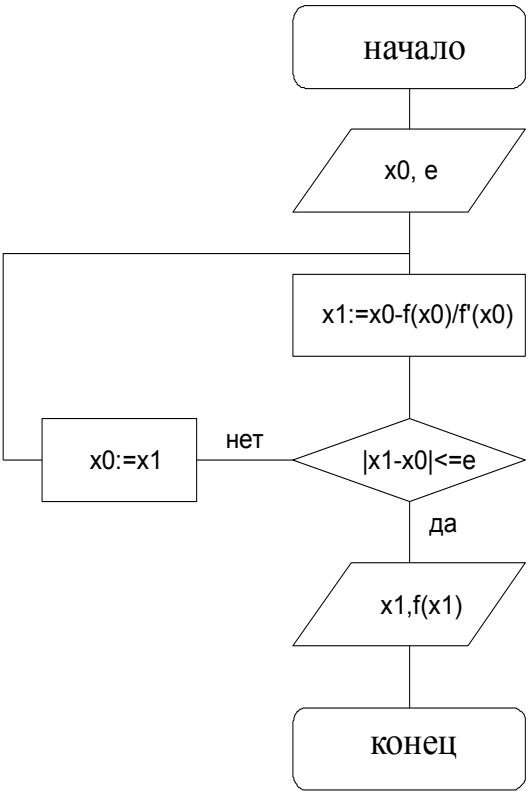
1. За начальное приближение принимается такое значение $x_0 \in [a;b]$, для которого выполняется условие Фурье: $f(x_0) \cdot f''(x_0) > 0$;

2. $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

3. Критерий окончания счета:

$$\begin{cases} |f(x_{n+1})| \le \varepsilon \\ |x_{n+1} - x_n| \le \varepsilon \end{cases}.$$

Блок-схема алгоритма решения нелинейных уравнений методом Ньютона



Задание

Решить нелинейные уравнения методами:

1) бисекции;
2) хорд;
3) простой итерации;
4) Ньютона.

Варианты заданий

№ варианта	Уравнение	Интервал изоляции корня
1	$x^2 - 5\sin x = 0$	[1,57; 3,14]
2	$e^x - 10x = 0$	[0; 1]
3	$x + \sqrt{x} + \sqrt[3]{x} - 2{,}5 = 0$	[0,4; 1]
4	$\sin x - x + 0{,}15 = 0$	[0,5; 1]
5	$x - \sqrt{9 + x} + x^2 - 4 = 0$	[2; 3]
6	$0{,}1x - x\ln x = 0$	[1; 2]
7	$x^4 - 26x^3 + 131x^2 - 226x + 120 = 0$	[19,5; 21,2]
8	$x^4 - 0{,}486x^3 - 5{,}792x^2 + 0{,}486x + 4{,}792 = 0$	[2; 3]
9	$0{,}1\sin x + x^3 - 1 = 0$	[0,8; 1,0]
10	$0{,}1e^x - \sin^2 x + 0{,}5 = 0$	[-2; 1]
11	$e^x - x - 1{,}25 = 0$	[0,618; 0,667]
12	$\sin x - 2x + 0{,}5 = 0$	[0,4; 0,5]
13	$e^x - 2(x-1)^2 = 0$	[0; 1]
14	$x - 1{,}25\ln x - 1{,}25 = 0$	[2,2; 2,4]
15	$3\sin\sqrt{x} + 0{,}35x - 3{,}8 = 0$	[2; 3]
16	$\sqrt{1-x} - \mathrm{tg}x = 0$	[0; 1]

Контрольные вопросы

1. В чем сущность методов бисекции, хорд, простой итерации, Ньютона?

2. Как выбирается начальное приближение в методах простой итерации и Ньютона?

3. К какому виду нужно преобразовать уравнение для метода простой итерации?

Лабораторная работа № 3

ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Методы решения систем линейных алгебраических уравнений (СЛАУ) можно разделить на две группы:

1) точные методы;
2) методы последовательных приближений.

С помощью ***точных методов***, проделав конечное число операций, можно получить точные значения неизвестных. При этом предполагается, что коэффициенты и правые части системы известны точно, а все вычисления проводятся без округлений. Чаще всего решение задач такими методами осуществляется поэтапно. На первом этапе систему преобразуют к тому или иному простому виду, на втором – решают упрощенную систему и получают значения неизвестных.

Методы ***последовательных приближений*** характеризуются тем, что с самого начала решение задается какими-то приближенными значениями неизвестных, которые затем многократно улучшаются тем или иным способом.

Система m линейных уравнений с m неизвестными имеет вид:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mm}x_m = b_m, \end{cases} \qquad (3.1)$$

где a_{ij}, b_i (i = 1, 2, ..., m; j = 1, 2, ..., m) – произвольные числа, называемые соответственно коэффициентами при переменных и свободными членами уравнений.

Система уравнений называется *совместной*, если она имеет хотя бы одно решение, и *несовместной*, если она не имеет решений.

Методы исключения неизвестных (метод Гаусса)

Наиболее распространенным методом решения СЛАУ является метод Гаусса, в основе которого лежит идея последовательного исключения неизвестных. С помощью элементарных преобразований система уравнений приводится к равносильной системе ступенчатого (или треугольного) вида, из которой последовательно, начиная с последних (по номеру) переменных, находятся все остальные переменные.

Рассмотрим решение системы (3.1) m линейных уравнений с m неизвестными в общем виде.

Предположим, что в системе (3.1) $a_{11} \neq 0$ (этого всегда можно добиться перестановкой уравнений или переменных с изменением их номеров).

Метод Гаусса состоит в последовательном исключении неизвестных из этой системы.

Предположим, что $a_{11} \neq 0$. Последовательно умножая первое уравнение на $-\frac{a_{i1}}{a_{11}}$ и складывая с i-м уравнением, исключим x_1 из всех уравнений, кроме первого. Получим систему:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1, \\ a_{22}^{(1)}x_2 + \ldots + a_{2m}^{(1)}x_m = b_2^{(1)}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m2}^{(1)}x_2 + \ldots + a_{mm}^{(1)}x_m = b_m^{(1)}, \end{cases}$$

где $a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}$; $b_i^{(1)} = b_i - \frac{a_{i1}b_1}{a_{11}}$; $i, j = 2, 3, \ldots, m$.

Аналогичным образом из полученной системы исключим x_2. Последовательно, исключая все неизвестные, получим систему треугольного вида:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \quad \ldots \quad + a_{1k}x_k + a_{1m}x_m = b_1, \\ a_{22}^{(1)}x_2 + \quad \ldots \quad + a_{2k}^{(1)}x_k + a_{2m}^{(1)}x_m = b_2^{(1)}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m-1,m-1}^{(m-1)}x_{m-1} + a_{m-1,m}^{(m-1)}x_m = b_m^{(m-1)}, \\ a_{m,m}^{(m-1)}x_m = b_m^{(m-1)}. \end{cases}$$

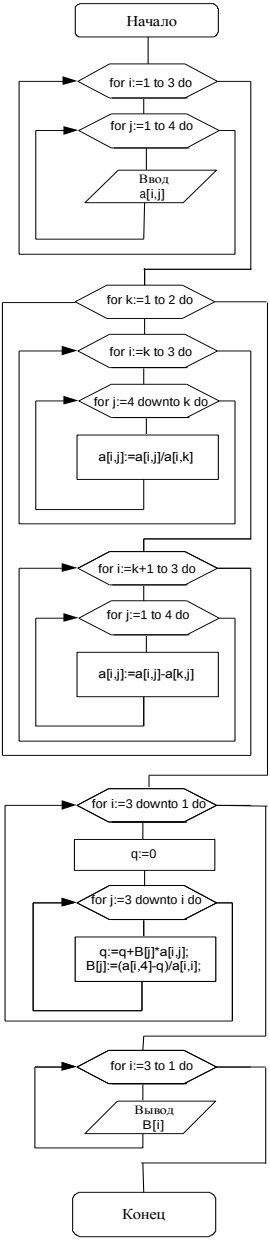
Описанная процедура называется прямым ходом метода Гаусса. Ее выполнение возможно при условии, что все a_{ii}^{l}, $(l = 1, 2, \ldots, m)$ не равны нулю.

Выполняя последовательные подстановки в последней системе (начиная с последнего уравнения), можно получить все значения неизвестных.

$$x_m = \frac{b_m^{(m-1)}}{a_{m,m}^{(m-1)}}, \quad x_i = \frac{1}{a_{i,i}^{(i-1)}}\left(b_{i,i}^{(i-1)} - \sum_{j=i-1}^{m} a_{ij}^{(i-1)}x_j \right).$$

Эта процедура получила название обратный ход метода Гаусса.

Блок-схема алгоритма решения СЛАУ методом Гаусса



Задание

Решить системы ЛАУ методом Гаусса.

Варианты заданий

№ варианта	Система уравнений
1	$\begin{cases} x+y+z=2 \\ 2x-y-z=-1 \\ x-y+z=0 \end{cases}$
2	$\begin{cases} 2,7x+3,3y+1,3z=2,1 \\ 3,5x-1,7y+2,8z=1,7 \\ 4,1x+5,8y-1,7z=0,8 \end{cases}$
3	$\begin{cases} 2x-y-z=0 \\ y+z=5 \\ 0,5x+y-z=2 \end{cases}$
4	$\begin{cases} 3,1x+2,8y+1,9z=0,2 \\ 1,9x+3,1y+2,1z=2,1 \\ 7,5x+3,8y+4,8z=5,6 \end{cases}$
5	$\begin{cases} 6x+y-z=10 \\ x-y+2z=5 \\ -x-y+6z=40 \end{cases}$
6	$\begin{cases} 3,6x+1,8y-4,7z=3,8 \\ 2,7x-3,6y+1,9z=0,4 \\ 1,5x+4,5y+3,3z=-1,6 \end{cases}$
7	$\begin{cases} 0,1x-y-z=2 \\ x+y+z=1 \\ x+y-z=-8 \end{cases}$

№ варианта	Система уравнений
8	$\begin{cases} 2,7x+0,9y-1,5z=3,5 \\ 4,5x-2,8y+6,7z=2,6 \\ 5,1x+3,7y-1,4z=-0,14 \end{cases}$
9	$\begin{cases} x+2y+z=1 \\ -x-2y+2z=1 \\ y+z=2 \end{cases}$
10	$\begin{cases} 3,8x+6,7y-1,2z=5,2 \\ 6,4x+1,3y-2,7z=3,8 \\ 2,4x+4,5y+3,5z=-0,6 \end{cases}$
11	$\begin{cases} 0,5x-y+0,5z=3 \\ x+y+z=1 \\ -x+0,5y-z=-1 \end{cases}$
12	$\begin{cases} 2,74x-1,18y+3,17z=2,18 \\ 1,12x+0,83y-2,16z=-1,15 \\ 0,81x+1,27y+0,76z=3,23 \end{cases}$
13	$\begin{cases} -2x-y=0,333 \\ -x+2y-z=1 \\ y+2z=-0,333 \end{cases}$
14	$\begin{cases} 10x+y+z=12 \\ 2x+10y+z=13 \\ 2x+2y+10z=14 \end{cases}$
15	$\begin{cases} 10x+y+z=10 \\ 2x+10y+z=13 \\ 2x+y+z=5 \end{cases}$

Контрольные вопросы

1. К какому виду приводится матрица в методе Гаусса?
2. В каком случае нельзя применить метод Гаусса?
3. Что нужно предусмотреть при использовании метода Гаусса?

Итерационные методы

Метод называется ***приближенным***, если при точном выполнении всех требуемых действий и при точных коэффициентах мы получаем, как правило, лишь приближенный результат.

Впрочем, на практике и при применении точного метода результат оказывается приближенным по двум причинам: во-первых, коэффициенты уравнений обычно бывают приближенными числами; во-вторых, промежуточные вычисления обычно невозможно выполнять с полной точностью. Значит погрешность окончательного результата складывается из неустранимой погрешности задания исходных данных (коэффициентов) и погрешностей округления в промежуточных действиях.

Итерационные методы, например, метод простой итерации или метод Зейделя, применяют для очень больших систем, когда метод Гаусса становится неэффективным.

Одношаговый линейный стационарный итерационный процесс называется ***методом простой итерации*** (***МПИ***).

Пусть дана система ЛАУ $A \cdot x = f$ с неособенной матрицей. В методе простой итерации ее предварительно приводят к виду:

$$x = B \cdot x + C.$$

Метод простой итерации

Вычисления по *методу простой итерации* начинаются с произвольного вектора $X^0 = \{x_1^0, x_2^0, ..., x_n^0\}$. Итерационный процесс осуществляется по формуле:

$$x_i^{j+1} = x_i^j + \frac{1}{a_{ii}} \cdot \left(b_i - \sum_{k=1}^{n} a_{ik} \cdot x_k^j \right); \; i = 1, 2, ..., n; \; j = 0, 1, ...,$$

т. е. *все* неизвестные на следующей итерации вычисляются *только* через неизвестные на предыдущей итерации.

Условия завершения итерационного процесса:

$$\delta \le \varepsilon,$$

где ε – требуемая точность;

δ – оценка достигнутой точности.

$$\delta = \max_{i=\overline{1,n}} \left| x_i^{(k)} - x_i^{(k-1)} \right|$$

или

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \left| x_i^{(k)} - x_i^{(k-1)} \right|.$$

Условие сходимости итерационного процесса (условие преобладания диагональных коэффициентов):

$$|a_{ii}| > \sum_{j=1}^{n} |a_{ij}|; \quad i = \overline{1,n}.$$

Метод Зейделя

Метод Зейделя отличается от простой итерации тем, что найдя какое-то приближение для компоненты, мы сразу же используем его для отыскания следующей компоненты.

В *методе Зейделя* используется итерационная формула

$$x_i^{j+1} = x_i^j + \frac{1}{a_{ii}} \cdot \left(b_i - \sum_{k=1}^{i-1} a_{ik} \cdot x_k^{j+1} - \sum_{k=i}^{n} a_{ik} \cdot x_k^j \right); \; i = 1, 2, \ldots, n; \; j = 0, 1, \ldots,$$

в которой при вычислении очередного неизвестного используются *последние* найденные значения остальных неизвестных. Вычисления заканчиваются, когда невязки системы становятся достаточно малыми. Итерационные методы сходятся не для всякой матрицы.

Достаточным условием сходимости является *положительная определенность* матриц.

Задание

Решить системы линейных алгебраических уравнений методами:
1) простой итерации;
2) Зейделя.

Варианты заданий

№ варианта	Система уравнений	Точность ε
1	$\begin{cases} 4x+0{,}24y-0{,}08z=8 \\ 0{,}09x+3y-0{,}15z=9 \\ 0{,}04x-0{,}08y+4z=20 \end{cases}$	10^{-5}
2	$\begin{cases} 2x-y+z=-3 \\ 3x+5y-2z=1 \\ x-4y+10z=0 \end{cases}$	10^{-5}
3	$\begin{cases} 10x+y+z=12 \\ 2x+10y+z=13 \\ 2x+2y+10z=14 \end{cases}$	10^{-5}
4	$\begin{cases} 2x-y=0{,}333 \\ -x+2y-z=1 \\ -y+2z=-0{,}333 \end{cases}$	10^{-5}
5	$\begin{cases} 7{,}6x+0{,}5y+2{,}4z=1{,}9 \\ 2{,}2x+9{,}1y+4{,}4z=9{,}7 \\ -1{,}3x+0{,}2y+5{,}8z=-1{,}4 \end{cases}$	10^{-5}
6	$\begin{cases} 25x+y-3{,}5z=5 \\ 9{,}4y-3{,}4z=-3 \\ x-y+7{,}3z=0 \end{cases}$	10^{-5}

№ варианта	Система уравнений	Точность ε
7	$\begin{cases} 2,7x+3,3y+1,3z=2,1 \\ 3,5x-1,7y+2,8z=1,7 \\ 4,1x+5,8y-1,7z=0,8 \end{cases}$	10^{-5}
8	$\begin{cases} 3,1x+2,8y+1,9z=0,2 \\ 1,9x+3,1y+2,1z=2,1 \\ 7,5x+3,8y+4,8z=5,6 \end{cases}$	10^{-5}
9	$\begin{cases} 3,6x+1,8y-4,7z=3,8 \\ 2,7x-3,6y+1,9z=0,4 \\ 1,5x+4,5y+3,3z=-1,6 \end{cases}$	10^{-5}
10	$\begin{cases} 2,7x+0,9y-1,5z=3,5 \\ 4,5x-2,8y+6,7z=2,6 \\ 5,1x+3,7y-1,4z=-0,14 \end{cases}$	10^{-5}
11	$\begin{cases} 3,8x+6,7y-1,2z=5,2 \\ 6,4x+1,3y-2,7z=3,8 \\ 2,4x-4,5y+3,5z=-0,6 \end{cases}$	10^{-5}
12	$\begin{cases} 1,02x-0,05y-0,1z=0,795 \\ -0,11x+1,03y-0,05z=0,849 \\ -0,11x-0,12y+1,04z=1,398 \end{cases}$	10^{-5}
13	$\begin{cases} 6x+2y+z=12 \\ 2x+4y-z=16 \\ x-y+5z=15 \end{cases}$	10^{-5}
14	$\begin{cases} 10x-2y-2z=6 \\ -x+10y-2z=7 \\ -x-y+10z=8 \end{cases}$	10^{-5}
15	$\begin{cases} 2x-y+z=-3 \\ 3x+5y-2z=1 \\ x-4y+10z=0 \end{cases}$	10^{-5}

Контрольные вопросы

1. Чем отличаются точные методы от итерационных?
2. Как выбираются начальные приближения в методах простой итерации и Зейделя?
3. Каково условие прекращения итерации в методе простой итерации и в методе Зейделя для решения систем линейных алгебраических уравнений?

Лабораторная работа № 4

ИНТЕРПОЛИРОВАНИЕ

Системы функций Чебышева

Рассмотрим линейное множество R действительных функций, определенных на отрезке $[a; b]$, и некоторую конечную или счетную систему линейно независимых функций $\{\varphi_i(x)\}$ из этого множества.

Линейную комбинацию

$$\tau(x) = \sum_{i=0}^{n} c_i \varphi_i(x)$$

с действительными коэффициентами c_i называют обобщенным многочленом по системе функций $\varphi_k(x)$ $(k = 0, 1, ..., n)$.

Определение. *Совокупность функций* $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$ *называется системой Чебышева на отрезке* $[a; b]$, *если любой обобщенный многочлен по этой системе, у которого хотя бы один из коэффициентов отличен от нуля, имеет на* $[a; b]$ *не более* n *корней.*

Пусть на отрезке $[a; b]$ в некоторых попарно различных точках $x_0, x_1, ..., x_n$ известны значения функций $f(x)$.

Задача интерполирования функции $f(x)$ состоит в том, чтобы найти значение $f(x), x \neq x_i$ $(i = 0,1,...,n)$, если известны узлы интерполирования $x_0, x_1, ..., x_n$ и значения функции $f(x)$ в этих узлах.

Решается задача интерполирования следующим образом: выбирается система функций $\varphi_i(x)$, строится обобщенный многочлен $\tau(x) = \sum_{i=0}^{n} c_i \varphi_i(x)$, а коэффициенты c_i задаются таким образом, чтобы в узлах интерполирования значения обобщенного многочлена совпадали со значениями данной функции $f(x)$:

$$\tau(x_k) = \sum_{i=0}^{n} c_i \varphi_i(x_k) = f(x_k), \quad k = 0,1,...,n.$$

Обобщенный многочлен, обладающий таким свойством, называется ***обобщенным интерполяционным многочленом***. За приближенное значение $f(x)$ принимают значение $\tau(x)$.

Выясним, когда задача интерполирования решается однозначно.

Теорема 1. *Для того чтобы для любой функции* $f(x) \in R$, *определенной на отрезке* $[a; b]$, *и любого набора* $n+1$ *узлов* $x_0, x_1, ..., x_n$ $(x_i \neq x^j$ *при* $i \neq j$, $x_i \in [a; b])$ *существовал и был бы единственным обобщенный интерполяционный многочлен* $\tau(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + ... + c_n\varphi_n(x)$, *необходимо и достаточно, чтобы система функций* $\varphi_i(x)$ $(i = 0,1, ..., n)$ *являлась системой Чебышева на* $[a; b]$.

На практике чаще всего используются следующие системы:

1) $1, x, x^2, ..., x^n, ...$;

2) $1, \sin x, \cos x, ..., \sin nx, \cos nx, ...$;

3) $e^{\alpha_0 x}, e^{\alpha_1 x}, ..., e^{\alpha_n x}, ...$, где α_i – некоторая числовая последовательность попарно различных действительных чисел.

В случае 1 интерполирование называется алгебраическим, в случае 2 – тригонометрическим (применяется для приближения 2π-периодических функций), в случае 3 – экспоненциальным.

Метод Лагранжа

Пусть $a = x_0 < x_1 < \ldots < x_n = b$ – набор различных точек (узлов) на отрезке $[a; b]$, в котором заданы значения достаточно гладкой функции $f(x)$ так, что $f_i = f(x_i)$, $i = 0, 1, \ldots, n$. Требуется построить многочлен $L_n(x)$ степени не выше n, принимающий в точках x_i значения f_i, и оценить погрешность приближения функции этим многочленом на всем отрезке $[a; b]$.

Введем в явном виде вспомогательные многочлены $\Omega_i(x)$ степени $n-1$, удовлетворяющие условиям $\Omega_i(x_j) = \begin{cases} 1, & \text{если } i = j \\ 0, & \text{если } i \neq j \end{cases}$, по формуле:

$$\Omega_i(x) = \prod_{\substack{j=0 \\ i \neq j}}^{n} \frac{x - x_j}{x_i - x_j}. \qquad (4.1)$$

Тогда интерполяционный многочлен $L_n(x)$ можно задать по формуле Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} \Omega_i(x) f_i, \qquad (4.2)$$

при этом $\Omega_i(x)$ удобно преобразовать к виду:

$$\Omega_i(x) = \frac{\omega(x)}{(x - x_i)\omega'(x_i)}, \qquad (4.3)$$

где $\omega(x) = \prod_{i=0}^{n} (x - x_i)$.

Разность $r_n(x) = f(x) - L_n(x)$ называется ***погрешностью интерполирования***, или остаточным членом интерполирования. В узлах интерполирования погрешность $r_n(x)$ обращается в нуль, в остальных точках она отлична от нуля, но если $f(x)$ – многочлен степени

k, $k < n+1$, то $r_n(x) \equiv 0$. Если функция $f(x)$ имеет непрерывную $(n+1)$-ю производную, то остаточный член можно представить в виде:

$$r_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x), \qquad (4.4)$$

где ξ – некоторая точка, лежащая на отрезке, содержащем узлы $x_0, x_1, \ldots, x_n$ и точку x.

Пример 1. Построить многочлен наименьшей степени, принимающий в данных точках следующие значения:

x	–1	0	3	6
y	–14	–5	22	175

По формуле (4.1) вычислим вспомогательные многочлены:

$$\Omega_0(x) = \frac{(x-0)(x-3)(x-6)}{(1-0)(1-3)(1-6)} = \frac{x^3-9x^2+18x}{-28},$$

$$\Omega_1(x) = \frac{(x-(-1))(x-3)(x-6)}{(0-(-1))(0-3)(0-6)} = \frac{x^3-8x^2+9x+18}{18},$$

$$\Omega_2(x) = \frac{(x-(-1))(x-0)(x-6)}{(3-(-1))(3-0)(3-6)} = \frac{x^3-5x^2-6x}{-36},$$

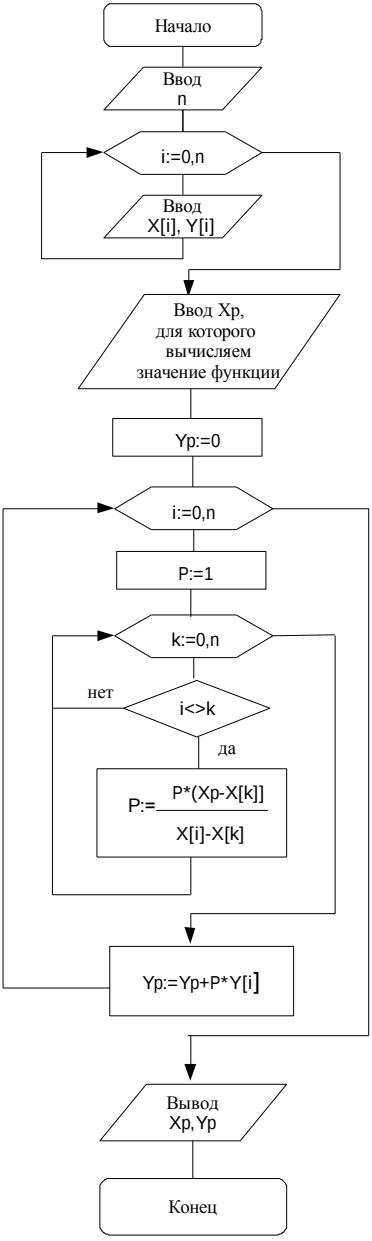
$$\Omega_3(x) = \frac{(x-(-1))(x-0)(x-3)}{(6-(-1))(6-0)(6-3)} = \frac{x^3-2x^2-3x}{126}.$$

Затем по формуле (4.2) построим интерполяционный многочлен Лагранжа:

$$L_3(x) = -14\frac{x^3-9x^2+18x}{-28} + (-5)\frac{x^3-8x^2+9x+18}{18} + 22\frac{x^3-5x^2-6x}{-36} +$$

$$+175\frac{x^3-2x^2-3x}{126} = x^3-2x^2+6x-5.$$

Блок-схема алгоритма интерполирования методом Лагранжа



Линейная интерполяция

Линейная интерполяция состоит в том, что заданные точки $M(x_i, y_i)$ $(i = 0, 1, ..., n)$ соединяются прямолинейными отрезками и функция $f(x)$ приближается к ломаной с вершинами в данных точках. Поскольку имеется n интервалов (x_{i-1}, x_i), то для каждого из них в качестве уравнения интерполяционного полинома используется уравнение прямой, проходящей через две точки. В частности для i-го интервала можно записать уравнение прямой, проходящей через точки (x_{i-1}, y_{i-1}) и (x_i, y_i), в виде:

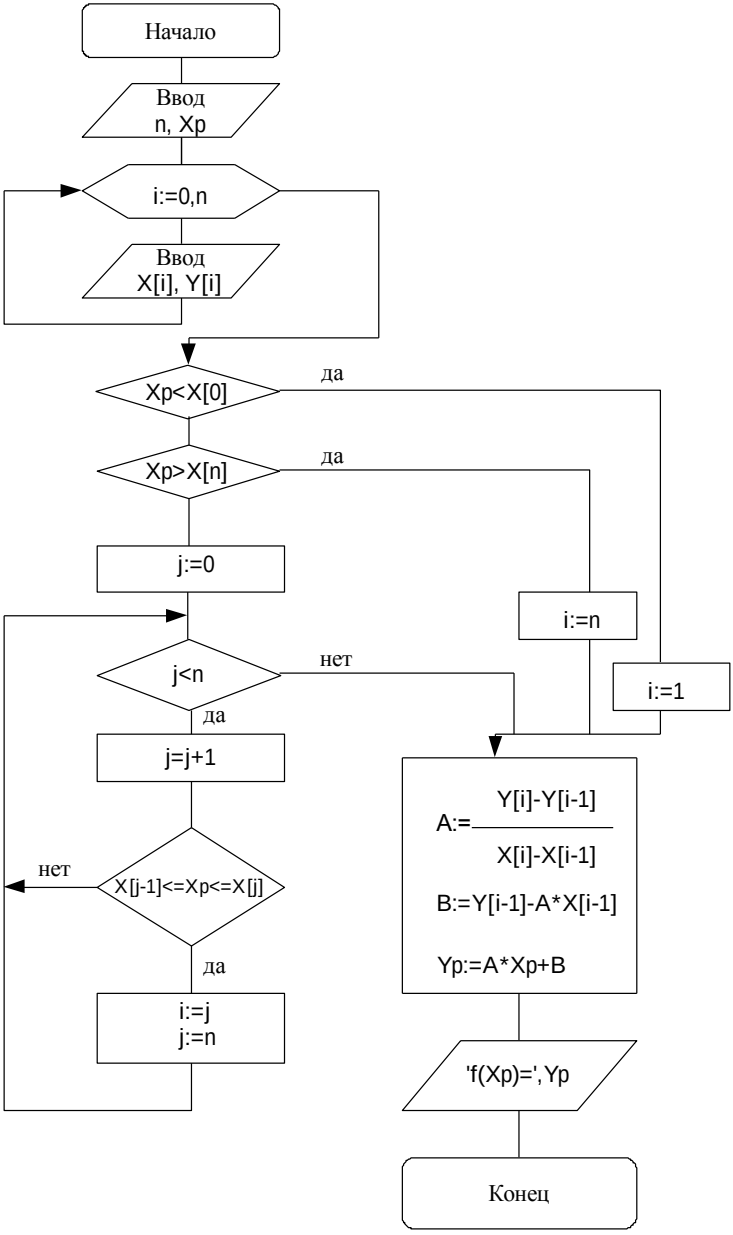
$$\frac{y - y_{i-1}}{y_i - y_{i-1}} = \frac{x - x_{i-1}}{x_i - x_{i-1}}. \qquad (4.5)$$

Отсюда

$$y = a_i x + b_i, \quad x_{i-1} \le x \le x_i,$$
$$a_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \quad b_i = y_{i-1} - a_i x_{i-1}. \qquad (4.6)$$

Следовательно, при использовании линейной интерполяции сначала нужно определить интервал, в который попадает значение аргумента x, а затем подставить его в формулу (4.6) и найти приближенное значение функции в этой точке.

Блок-схема алгоритма линейной интерполяции



Задание

Решить задания методами:
1) линейной интерполяции;
2) Лагранжа.

Варианты заданий

№ варианта	Условие								*f(x)*
1	*x*	93	96,2	100	104,2	108,7			*f*(102)
	f(x)	11,38	12,8	14,7	17,07	19,91			
2	*x*	0	2	3	6	7	9		*f*(5)
	f(x)	658 503	704 969	729 000	804 357	830 584	884 736		
3	*x*	2	2,3	2,5	3,0	3,5	3,8	4	*f*(3,75)
	f(x)	5,848	6,127	6,3	6,694	7,047	7,243	7,368	
4	*x*	14	17	31	35				*f*(20)
	f(x)	68,7	64	44	39,1				
5	*x*	10	15	17	20				*x* при *f*(*x*)=10
	f(x)	3	7	11	17				
6	*x*	1	2	2,5	3				*x* при *f*(*x*)=0
	f(x)	–6	–1	5,625	16				
7	*x*	4	6	8	10				*x* при *f*(*x*)=20
	f(x)	11	27	50	83				
8	*x*	1	1,1	1,2	1,3	1,4			*f*(1,13)
	f(x)	1,1752	1,33565	1,50946	1,69838	1,9043			

№ вари-анта	Условие								f(x)
9	x	1,5	1,6	1,7	1,8				f(1,75)
	f(x)	2,12928	2,37587	2,64563	2,94217				
10	x	1	1,1	1,2	1,3	1,4			f(1,23)
	f(x)	0,6827	0,7287	0,7699	0,8064	0,8385			
11	x	1,5	1,6	1,6	1,7	1,8			f(1,61)
	f(x)	0,8664	0,8904	0,9109	0,9281	0,9426			
12	x	0,3	0,4	0,5	0,6	0,7			f(0,55)
	f(x)	0,2913	0,3799	0,4621	0,5380	0,6044			
13	x	0,8	0,9	1,0	1,1				f(0,87)
	f(x)	0,664	0,7163	0,7616	0,8005				
14	x	1	1,1		1,2	1,3	1,4		f(1,25)
	f(x)	1	0,95135		0,91817	0,98747	0,88726		
15	x	2	2,2		2,4	2,6			x при f(x)=0,1
	f(x)	0,224	0,1104		0,0025	–0,0968			
16	x	0,1	0,15		0,19	0,25			x при f(x)=0,2
	f(x)	1,1052	1,1618		1,2092	1,284			
17	x	0,2	0,24		0,26	0,29			f(0,21)
	f(x)	1,2214	1,2712		1,2969	1,3364			
18	x	0,1	0,13	0,17	0,2				f(0,15)
	f(x)	0,0998	0,1296	0,1692	0,1987				

Контрольные вопросы

1. В каких случаях прибегают к интерполяции?
2. В чем заключается метод линейной интерполяции?
3. В чем заключается метод Лагранжа?

Лабораторная работа № 5

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Метод Ньютона

Метод Ньютона применяется к решению систем уравнений вида

$$\begin{cases} f(x,y)=0 \\ g(x,y)=0 \end{cases}. \tag{5.1}$$

$J(x,y)=\begin{bmatrix} f'_x(x,y) & f'_y(x,y) \\ g'_x(x,y) & g'_y(x,y) \end{bmatrix}$ – матрица Якоби.

Тогда последовательные приближения по методу Ньютона вычисляются по формуле:

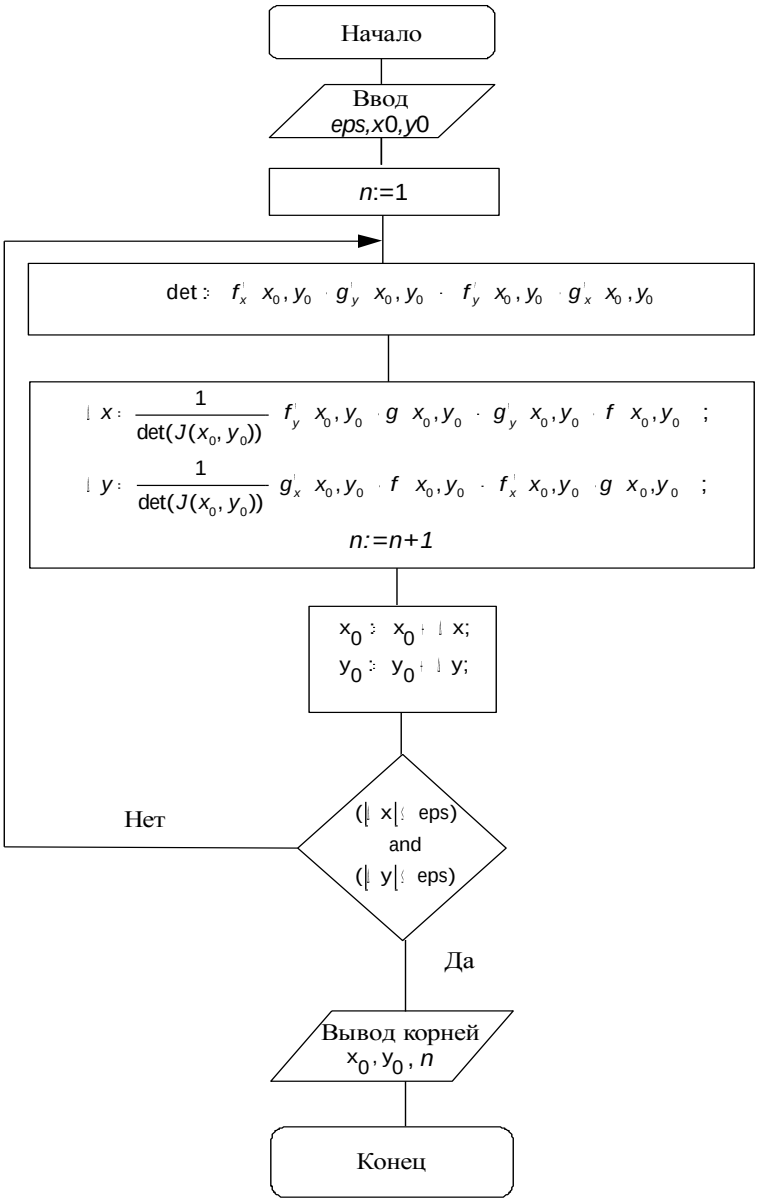
$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix}=\begin{pmatrix} x_n \\ y_n \end{pmatrix}-J^{-1}(x_n,y_n)\begin{pmatrix} f(x_n,y_n) \\ g(x_n,y_n) \end{pmatrix}, \tag{5.2}$$

где $J^{-1}(x_n,y_n)=\dfrac{1}{\det J(x_n,y_n)}\begin{vmatrix} g'_y(x_n,y_n) & -f'_y(x_n,y_n) \\ -g'_y(x_n,y_n) & f'_x(x_n,y_n) \end{vmatrix}$.

Тогда

$$\begin{cases} x_{n+1}=x_n-\dfrac{1}{\det J(x_n,y_n)}\left(g'_y(x_n,y_n)\cdot f(x_n,y_n)-f'_y(x_n,y_n)\cdot g(x_n,y_n)\right) \\ y_{n+1}=y_n-\dfrac{1}{\det J(x_n,y_n)}\left(-g'_x(x_n,y_n)\cdot f(x_n,y_n)+f'_x(x_n,y_n)\cdot g(x_n,y_n)\right) \end{cases}.$$

Блок-схема алгоритма решения систем нелинейных уравнений методом Ньютона



Задание

Решить системы нелинейных уравнений методом Ньютона.

Варианты заданий

№ варианта	Система уравнений	Точность ε
1	$\begin{cases}\sin x - y = 0\\ \cos y - x + 1{,}386 = 0\end{cases}$	10^{-3}
2	$\begin{cases}x^7 - 5x^2y^2 + 1510 = 0\\ y^5 - 3x^4y - 105 = 0\end{cases}$	10^{-5}
3	$\begin{cases}5x - 6y + 20\lg x + 16 = 0\\ 2x - y - 10\lg y - 4 = 0\end{cases}$	10^{-3}
4	$\begin{cases}x^3 - y^2 - 1 = 0\\ xy^3 - y - 4 = 0\end{cases}$	10^{-5}
5	$\begin{cases}\sin(x+1) - y = 0\\ 2x + \cos y = 2\end{cases}$	10^{-3}
6	$\begin{cases}x^2y^2 - 3x^3 - 6y^3 + 8 = 0\\ x^2 - 9y + 2 = 0\end{cases}$	10^{-5}
7	$\begin{cases}\cos\frac{1}{3}(x-y) - 2y = 0\\ \sin\frac{1}{3}(x+y) - 2x = 0\end{cases}$	10^{-3}
8	$\begin{cases}e^{xy} = x^2 - y + 1{,}1\\ (x+0{,}5)^{1{,}1} + y^2 = 0{,}1\end{cases}$	10^{-5}
9	$\begin{cases}\sin(x-y) - 1{,}3x = 0{,}1\\ x^2 - y^2 = \frac{3}{4}\end{cases}$	10^{-5}

№ варианта	Система уравнений	Точность ε
10	$\begin{cases}\sin(x+y)-1{,}3x=0{,}1\\ x^2+y^2=1\end{cases}$	10^{-2}
11	$\begin{cases}\sin(x+y)-1{,}3x=0{,}1\\ x^2+y^2=1\end{cases}$	10^{-5}
12	$\begin{cases}x^{10}+y^{10}=1024\\ e^x-e^y=1\end{cases}$	10^{-2}
13	$\begin{cases}x^3-y^2=1\\ xy^3-y-4=0\end{cases}$	10^{-3}
14	$\begin{cases}\sin(x-2y)-xy+1=0\\ x^2-y^2=1\end{cases}$	10^{-2}
15	$\begin{cases}x+3\lg x-y^2=0\\ 2x^2-xy-5x+1=0\end{cases}$	10^{-5}

Контрольные вопросы

1. Как выбрать начальное приближение в методе Ньютона для решения систем нелинейных уравнений?
2. Как определяется матрица Якоби?
3. Какое условие сходимости должно выполняться в методе Ньютона?

Лабораторная работа № 6

РЕШЕНИЕ ЗАДАЧИ АППРОКСИМАЦИИ

Аппроксимация – это процесс определения аналитического вида функции, заданной таблично.

Задача аппроксимации сводится к нахождению свободных параметров функции заданного вида, которые обеспечивают наилучшее приближение функции, заданной таблично.

Наиболее распространенным методом аппроксимации полинома является аппроксимация методом наименьших квадратов.

Метод наименьших квадратов (полином второй степени)

Пусть дана функция $y_i = f(x_i)$ $(i = \overline{1, n})$, которая задается таблично.

Пусть полином, наилучшим образом описывающий таблицу (x_i, y_i), равен:

$$P_m = \sum_{i=0}^{m} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m . \qquad (6.1)$$

Для нахождения коэффициентов $a_0, a_1, \ldots, a_m$ используем метод наименьших квадратов.

За меру отклонения полинома $P_m(x)$ от данной функции $f(x)$ на множестве $x_1, x_2, \ldots, x_n$ принимают величину

$$S_m = \sum_{i=1}^{n} (P_m(x_i) - f(x_i))^2 , \qquad (6.2)$$

равную сумме квадратов отклонений полинома $P_m(x)$ от функции $f(x)$ на заданной системе точек.

Критерием метода наименьших квадратов является минимизация функции многих переменных S_m.

Из условия минимума формируется система линейных уравнений относительно $a_0, a_1, \ldots, a_m$:

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i) = 0, \\ \dfrac{\partial S}{\partial a_0} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i) x_i = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \dfrac{\partial S}{\partial a_0} = 2\sum\limits_{i=1}^{n}(a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i) x_i^m = 0. \end{cases}$$

После преобразований система имеет вид

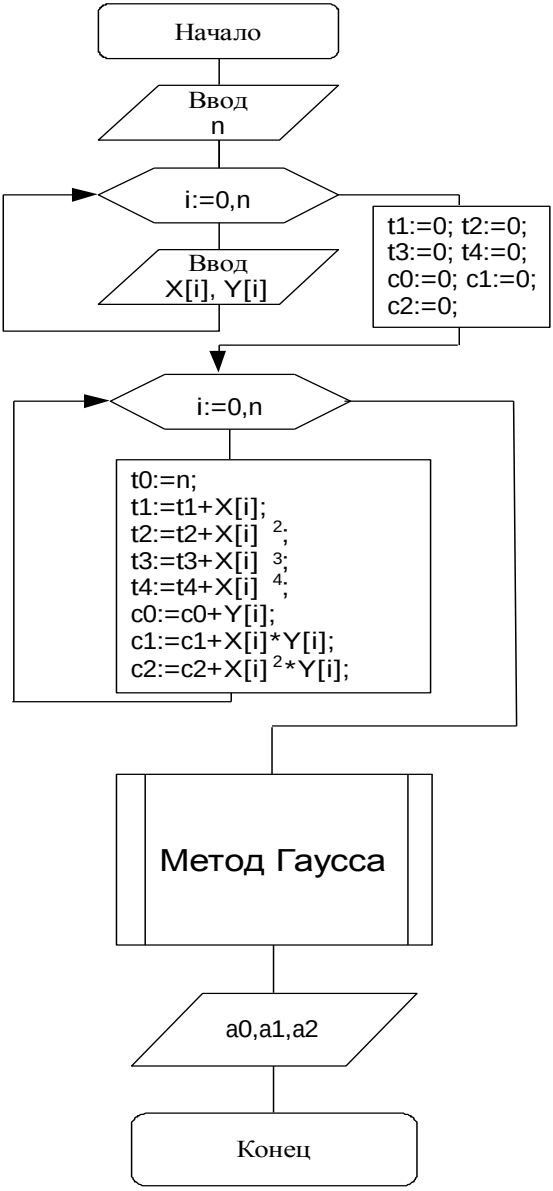
$$\begin{cases} \sum\limits_{i=1}^{n}(a_0 + a_1 x_i + \ldots + a_m x_i^m) = \sum\limits_{i=1}^{n} y_i, \\ \sum\limits_{i=1}^{n}(a_0 x_i + a_1 x_i^2 + \ldots + a_m x_i^{m+1}) = \sum\limits_{i=1}^{n} x_i y_i, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum\limits_{i=1}^{n}(a_0 x_i^m + a_1 x_i^{m+1} + \ldots + a_m x_i^{2m}) = \sum\limits_{i=1}^{n} x_i^m y_i. \end{cases}$$

После введения обозначений $t_0 = n$; $t_k = \sum\limits_{i=1}^{n} x_i^k$, $k = \overline{0, 2m}$; $c_k = \sum\limits_{i=1}^{n} x_i^k \cdot y_i$, $k = \overline{0,m}$ линейная система имеет вид

$$\begin{cases} t_0 a_0 + t_1 a_1 + \ldots + t_m a_m = c_0, \\ t_1 a_0 + t_2 a_1 + \ldots + t_{m+1} a_m = c_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ t_m a_0 + t_{m+1} a_1 + \ldots + t_{2m} a_m = c_m. \end{cases}$$

Эта линейная система может быть решена любым методом (методом Гаусса или итерационными методами).

Блок-схема алгоритма решения задачи аппроксимации методом наименьших квадратов



Задание

Выполнить задания методом наименьших квадратов.

Варианты заданий

№ варианта	Выборка							
1	*x*	0	1,5	3	4,5	6		
	y	0	2	3	3,5	3,8		
2	*x*	0	0,9	1,5	1,7	1,7	1,6	
	y	0	1	2	3	4	5	
3	*x*	2	2,4	3,2	4,5	6,5		
	y	0	1	2	3	4		
4	*x*	3	4,1	4,5	4,3	3,3	1,8	
	y	3	4	5	6	7	8	
5	*x*	2,5	3	3,5	4	4,5	5	5,5
	y	0	0,6	1,2	1,8	2,9	4,5	6,5
6	*x*	0	1	2	3		4	
	y	3	4,5	5	3,6		2,1	
7	*x*	0	1	2	3	4		
	y	6,3	3,8	2,5	1,85	1,4		
8	*x*	4	5	6	7	8		
	y	6	5,5	5,2	5	4,9		

№ варианта	Выборка						
9	*x*	1,6	2	3	4	5	6
	y	6	3,4	2,8	3,3	4,5	5,6
10	*x*	*4*	5	6	7	8	
	y	*1*	0,1	0,7	1,75	3,1	
11	*x*	0	0,1	0,2	0,3		0,4
	y	1	–1,2	–2,4	–3		– 3,5
12	*x*	0	0,1	0,2	0,3		0,4
	y	3	0,2	0,6	1,75		3,1
13	*x*	0,3	0,4	0,5	0,6		0,7
	y	–1	1,1	1,65	1,25		– 0,5
14	*x*	0,1	0,2	0,3	0,4		0,5
	y	1,75	2	2,55	4,35		5,5
15	*x*	0,3	0,4	0,5	0,6		0,7
	y	–2,85	–2,55	–2,1	–1,4		0
16	*x*	0,4	0,5	0,6	0,7		
	y	5	3	2,1	1,5		
17	*x*	0,1	0,2	0,3	0,4	0,5	
	y	4	2	0,9	0	–0,25	
18	*x*	0	0,1	0,2	0,3	0,4	0,5
	y	–3	–2	–1,2	–1,1	–1,5	–2

Контрольные вопросы

1. Что такое аппроксимация?
2. В чем заключается задача аппроксимации?
3. В чем заключается критерий метода наименьших квадратов?

Лабораторная работа № 7

ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННОГО ИНТЕГРАЛА

Необходимость вычисления значений определенных интегралов при моделировании возникает достаточно часто.

Формула Ньютона–Лейбница

$$I^* = \int_a^b f(x)dx = F(b) - F(a) \qquad (7.1)$$

имеет ограниченное применение, т. к.:

1) не позволяет вычислить интегралы от таблично заданной подынтегральной функции $f(x)$;

2) не всякая подынтегральная функция имеет первообразную $F(x)$.

Численные методы интегрирования универсальны: позволяют вычислить значение определенного интеграла непосредственно по значениям подынтегральной функции $f(x)$ независимо от способа ее задания или вида аналитического выражения.

Геометрический смысл определенного интеграла – площадь криволинейной трапеции, ограниченной осью OX, кривой $f(x)$ и прямыми $x = a$ и $x = b$ (рис. 7.1).

Численные методы интегрирования основаны на различных способах оценки этой площади, поэтому полученные формулы численного интегрирования называются ***квадратурными*** (формулами вычисления площади).

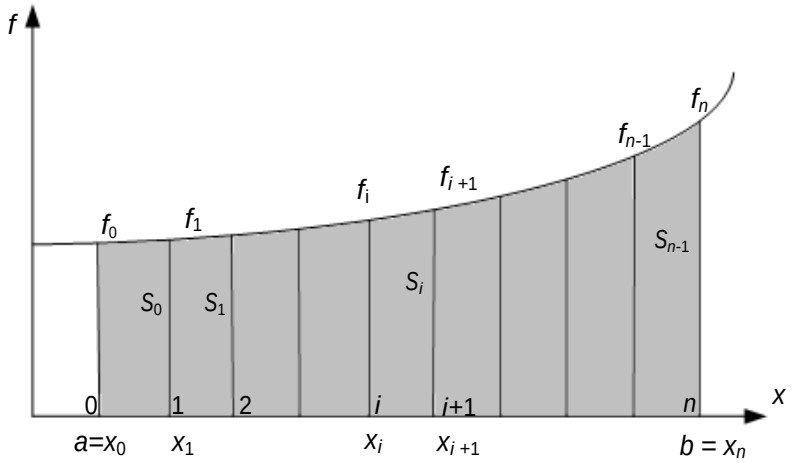


Рис. 7.1. Геометрический смысл определенного интеграла

Отрезок $[a; b]$ делят на n равных частей – ***элементарных отрезков***. Принято такое деление отрезка называть сеткой, а точки $x_0, x_1, \ldots, x_n$ – узлами сетки.

Если сетка равномерная, то $h = \dfrac{b-a}{n}$ – шаг сетки, при интегрировании – шаг интегрирования, а координата i-го узла вычисляется по формуле:

$$x_i = a + i \cdot h, \quad i = \overline{0,n}. \tag{7.2}$$

Полная площадь криволинейной трапеции состоит из n элементарных криволинейных трапеций – элементарных площадей:

$$I^* = \sum_{i=0}^{n-1} S_i. \tag{7.3}$$

Квадратурные формулы отличаются друг от друга способом оценки значения S_i – площади элементарной криволинейной трапеции.

Рассмотрим получение простейших формул для часто используемой равномерной сетки.

Метод прямоугольников

Площадь i-й элементарной трапеции можно оценить (приближенно вычислить) как площадь прямоугольника со сторонами $x_{i+1} - x_i = h$ и f_i (рис. 7.2). Тогда $S_i \approx f_i \cdot h$ и значение интеграла

$$I^* = \sum_{i=0}^{n-1} S_i \approx \sum_{i=0}^{n-1} S_i \cdot h = h \cdot \sum_{i=0}^{n-1} f_i = h \cdot \sum_{i=0}^{n-1} f(x_i) = h \cdot \sum_{i=0}^{n-1} f(a + i \cdot h). \quad (7.4)$$

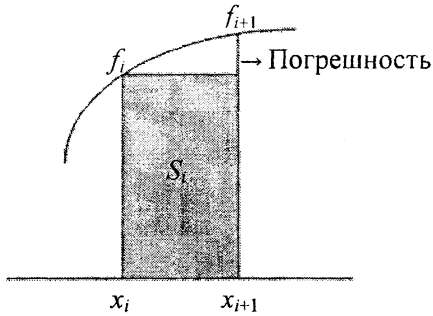


Рис. 7.2. Оценка элементарной площади S_i левым прямоугольником

Полученная формула называется формулой ***левых прямоугольников***, т. к. для оценки площади использовалось левое основание элементарной криволинейной трапеции.

Аналогично можно получить формулу ***правых прямоугольников*** (рис. 7.3):

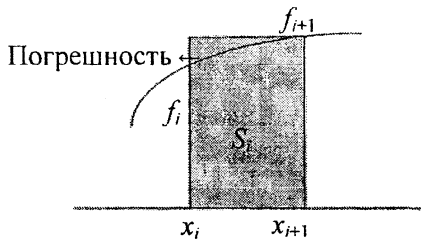


Рис. 7.3. Оценка элементарной площади S_i правым прямоугольником

Для данного случая $S_i \approx f_{i+1} \cdot h$ и тогда значение интеграла

$$I^* = \sum_{i=0}^{n-1} S_i \approx \sum_{i=0}^{n-1} f_{i+1} \cdot h = h \cdot \sum_{i=0}^{n-1} f_i = h \cdot \sum_{i=0}^{n-1} f(x_i) = h \cdot \sum_{i=0}^{n-1} f(a + i \cdot h). \quad (7.5)$$

Эти формулы не находят широкого применения, т. к. имеют большую погрешность, пропорциональную величине шага $\delta_м \approx O(h)$.

Как появляется эта погрешность, видно на рис. 7.2 и 7.3.

Для повышения точности площадь S_i можно оценить, используя прямоугольник со стороной, равной значению подынтегральной функции в середине элементарного отрезка $\tilde{f}_i = f(x_i + \frac{h}{2})$ (рис. 7.4).

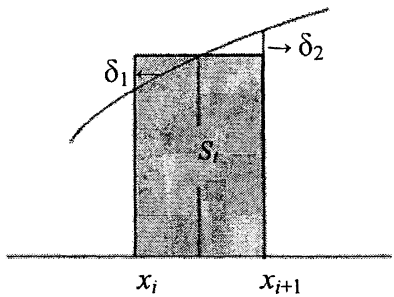


Рис. 7.4. Оценка элементарной площади S_i центральным прямоугольником

Для данного случая $S_i \approx \tilde{f}_i \cdot h$ и формула центральных прямоугольников имеет вид

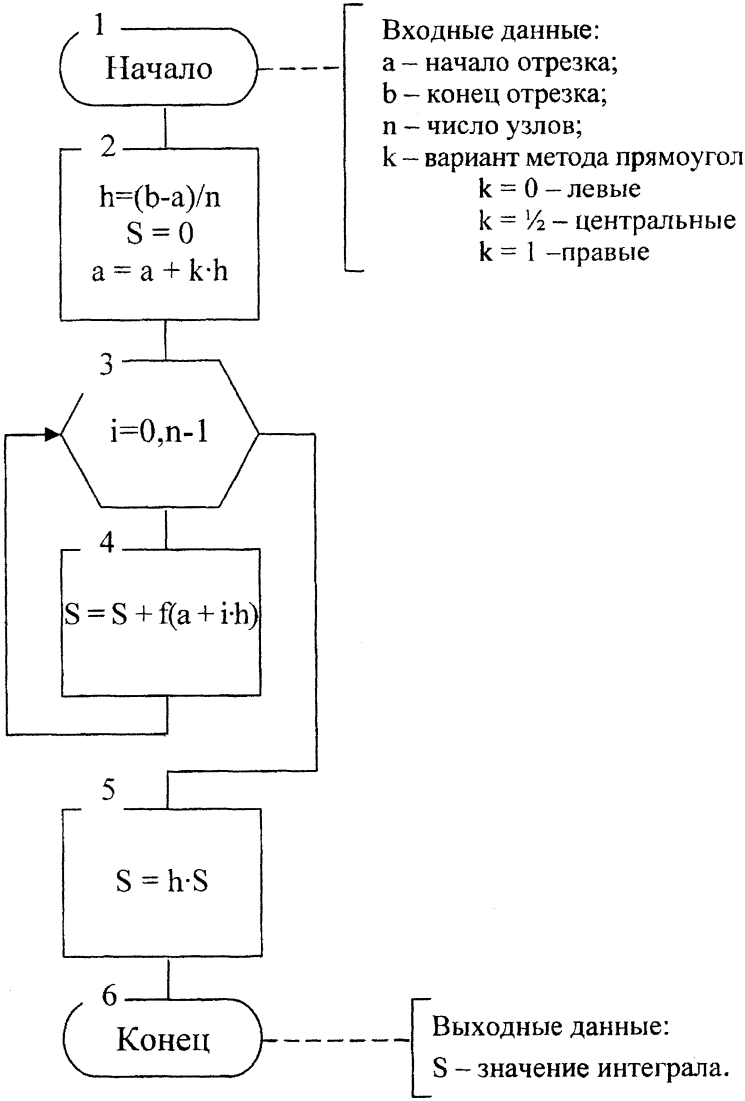
$$I^* = \sum_{i=0}^{n-1} S_i \approx \sum_{i=0}^{n-1} \tilde{f}_i \cdot h = h \cdot \sum_{i=0}^{n-1} \tilde{f}_i = h \cdot \sum_{i=0}^{n-1} f(x_i + \frac{h}{2}) = h \cdot \sum_{i=0}^{n-1} f(a + i \cdot h + \frac{h}{2}). \quad (7.6)$$

Как видно из рис. 7.4., погрешность в оценке площади S_i в данном случае существенно меньше, чем в двух предыдущих (погрешность оценивается разницей площадей δ_1 и δ_2).

Погрешность метода пропорциональна квадрату величины шага $\delta_м \approx O(h^2)$.

Формула центральных прямоугольников на порядок точнее предыдущих формул.

Блок-схема алгоритма вычисления определенного интеграла методом прямоугольников



Метод трапеций

В данном методе дуга $f(x)$ заменяется хордой CD (рис. 7.5).

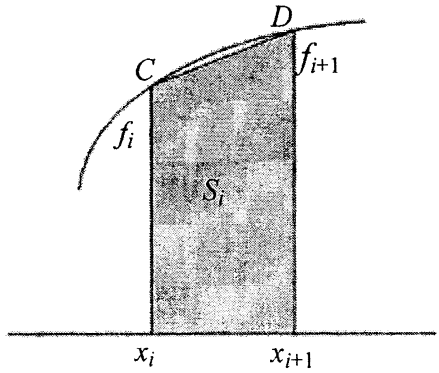


Рис. 7.5. Оценка элементарной площади S_i трапецией

Из рис. 7.5. видно, что $S_i \approx \dfrac{f_i + f_{i+1}}{2} \cdot h.$

Отсюда

$$I^* = \sum_{i=0}^{n-1} S_i \approx \sum_{i=0}^{n-1} \frac{f_i + f_{i+1}}{2} \cdot h = h \cdot \left(\frac{f_0 + f_1}{2} + \frac{f_1 + f_2}{2} + \frac{f_2 + f_3}{2} + \ldots + \right.$$

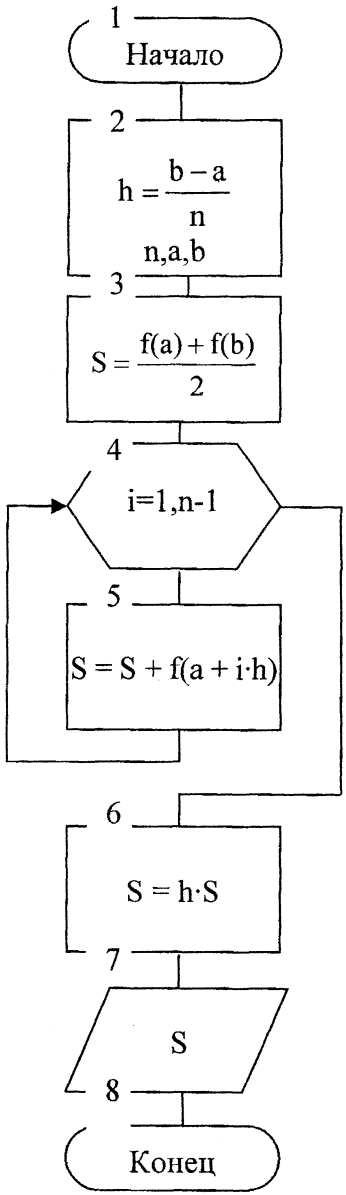
$$\left. + \frac{f_{n-2} + f_{n-1}}{2} + \frac{f_{n-1} + f_n}{2} \right) = h \cdot \left(\frac{f_0 + f_n}{2} + \sum_{i=1}^{n-1} f_i \right) = \qquad (7.7)$$

$$= h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(a + i \cdot h) \right).$$

Погрешность формулы трапеций пропорциональна квадрату шага h $\delta_м \approx O(h^2)$, т. е. **формулы центральных прямоугольников и трапеций имеют близкую точность.**

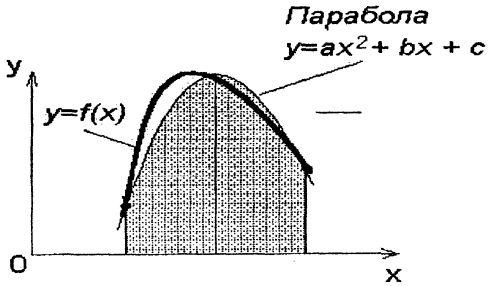
Знак погрешности легко объяснить по геометрической иллюстрации применения формулы.

Блок-схема алгоритма вычисления определенного интеграла методом трапеций



Метод Симпсона

Графическое представление метода Симпсона



На каждом элементарном отрезке подынтегральная функция *f(x)* заменяется квадратичной параболой, построенной по трем точкам: концам элементарного отрезка (x_i, f_i), (x_{i+1}, f_{i+1}) и его середине $(x_i + \frac{h}{2}, \tilde{f}_i)$.

Площадь полученной криволинейной трапеции служит оценкой элементарной площади S_i:

$$S_i \approx \frac{h}{6}\cdot\left(f_i + 4\tilde{f}_i + f_{i+1}\right) = \frac{h}{6}\cdot\left(f(x_i) + 4f(x_i + \frac{h}{2}) + f(x_{i+1})\right).$$

Тогда значение интеграла

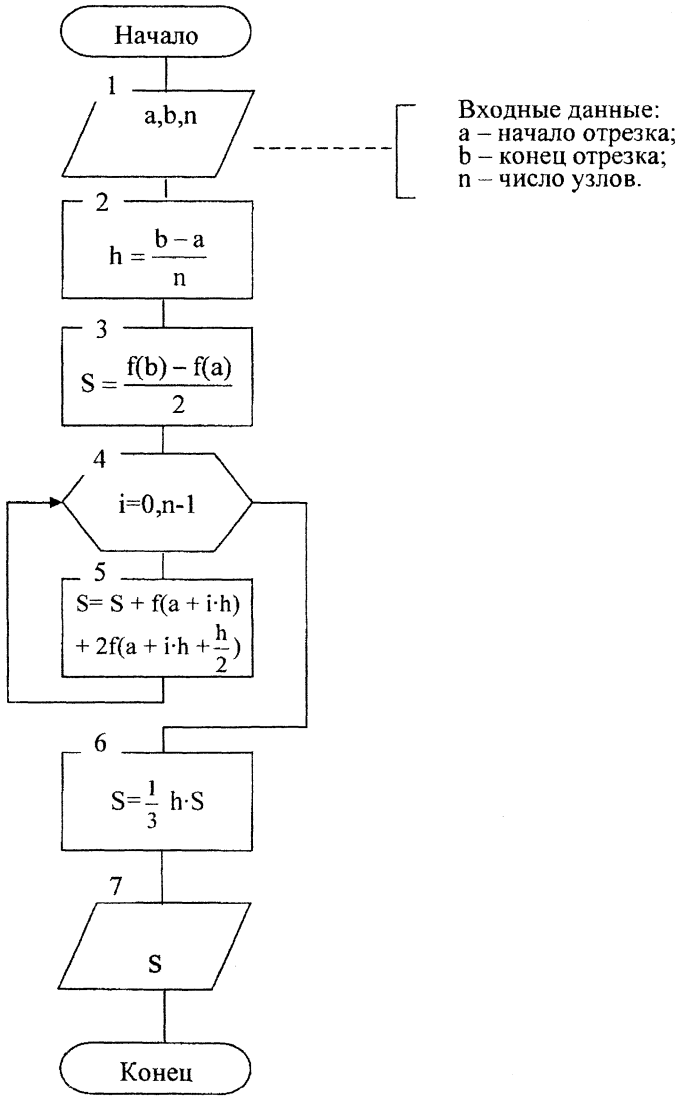
$$I^* \approx \sum_{i=0}^{n-1}\frac{h}{6}\cdot\left(f_i + 4\tilde{f}_i + f_{i+1}\right) = \frac{h}{6}\left(f_0 + 4\tilde{f}_0 + f_1 + f_1 + 4\tilde{f}_1 + f_2 + \ldots + \right.$$
$$\left. + f_{n-2} + 4\tilde{f}_{n-2} + f_{n-1} + f_{n-1} + 4\tilde{f}_{n-1} + f_n\right)$$

Преобразуем данную формулу:

$$I^* \approx \frac{h}{3}\cdot\left[\frac{f_n - f_0}{2} + \sum_{i=0}^{n-1}(f_i + 2\tilde{f}_i)\right] =$$
$$= \frac{h}{3}\cdot\left[\frac{f(b) - f(a)}{2} + \sum_{i=0}^{n-1}\left(f(a+ih) + 2f(a+ih+\frac{h}{2})\right)\right]. \qquad (7.8)$$

Формула Симпсона имеет высокую точность, так как погрешность метода $\delta_м \approx O(h^3)$.

Блок-схема алгоритма вычисления определенного интеграла методом Симпсона



Точность и сходимость методов прямоугольников, трапеций, Симпсона

Формулы для оценки погрешности численного интегрирования методом:

1) прямоугольников

$$\left|R_{\text{пр}}\right| \le \frac{(b-a)^3}{24n^2} M_2; \tag{7.9}$$

2) трапеций

$$\left|R_{\text{тр}}\right| \le \frac{(b-a)^3}{24n^2} M_2; \tag{7.10}$$

3) Симпсона

$$\left|R_{\text{с}}\right| \le \frac{(b-a)^5}{2880n^4} M_4, \tag{7.11}$$

где $M_2 = \max\limits_{x\in[a;\,b]} \left|f''(x)\right|$, $M_4 = \max\limits_{x\in[a;\,b]} \left|f^{(4)}(x)\right|$.

Формула Симпсона обладает повышенной точностью, т. к.:

1) она оказывается точной для $f(x)$, являющихся полиномами до третьей степени включительно, т. к. для этих случаев производная четвертого порядка равна нулю;

2) для достижения той же точности, что и в формуле трапеций, в формуле Симпсона можно брать меньшее число n отрезков разбиения.

Задание

Вычислить определенный интеграл методами:

1) трапеций;

2) прямоугольников;

3) Симпсона.

Варианты заданий

№ варианта	Подынтегральная функция	Интервал интегрирования $[a; b]$	Кол-во частей разбиения: n_1, n_2, n_3	Первообразная функция $F(x)$
1	$\frac{x}{(x+3)^2}$	[2; 5]	40, 80, 200	$\frac{3}{x+3}+\ln(x+3)$
2	$\frac{\sqrt{4-x^2}}{x}$	[3; 7]	80, 150, 400	$\sqrt{4-x^2}-2\ln\frac{2+\sqrt{4-x^2}}{x}$
3	$x\cdot\sin 2x$	[0,9; 3,1]	20, 100, 500	$\frac{\sin 2x}{4}-\frac{x\cos 2x}{2}$
4	2^{3x}	$[0, 2; \frac{\pi}{4}]$	50, 180, 400	$\frac{2^{3x}}{3\ln 2}$
5	$\frac{\ln^2 x}{x}$	[0,8; 1,9]	50, 200, 1000	$\frac{\ln^3 x}{3}$
6	$e^{2x}\sin x$	[1; 5]	30, 500, 1200	$\frac{e^{2x}}{5}(2\sin x-\cos x)$
7	$\frac{x^2}{2x+3}$	[2; 6]	100, 300, 2000	$\frac{1}{8}(2x^2-6x+9\ln(2x+3))$
8	$x^2\sqrt{x+2}$	[1; 3]	50, 400, 2500	$\frac{2((15x^2-24x+32)}{105}\sqrt{(x+2)^3}$
9	$\frac{x}{\sin^2 3x}$	[0,8; 4,5]	25,150, 1000	$-\frac{x}{3}\operatorname{ctg}3x+\frac{1}{9}\ln\sin 3x$
10	$x\cdot e^{0,8x}$	[2; 3]	40, 300, 2000	$\frac{e^{0,8x}}{0,64}(0,8x-1)$
11	$\frac{1}{x\sqrt{x^2+0,25}}$	[1,7; 3,2]	50, 250, 500	$-2\ln(\frac{0,5+\sqrt{x^2+0.25}}{x})$
12	$\frac{x^2}{(2x+0,3)^2}$	[2,1; 4,2]	80, 300, 2000	$0,25x-0,075\ln(2x+3)-\frac{0,01125}{2x+0,3}$

№ варианта	Подынтегральная функция	Интервал интегрирования $[a; b]$	Кол-во частей разбиения: $n1, n2, n3$	Первообразная функция $F(x)$
13	$\frac{x}{0{,}5x+0{,}1}$	[3; 5]	50, 500, 4000	$\frac{8(0{,}5x-0{,}2)}{3}\sqrt{0{,}5x+0{,}1}$
14	$x^2 \sin x$	[2; 3,1]	40, 200, 5000	$2x\sin x-(x^2-2)\cos x$
15	$x\cdot 2^{3x}$	[2; 4]	60, 180, 3500	$\frac{x\cdot 2^{3x}}{3\ln 2}-\frac{2^{3x}}{9(\ln 2)^2}$

Контрольные вопросы

1. Объяснить геометрический смысл определенного интеграла.
2. Какой зависимостью связан шаг интегрирования с количеством интервалов?
3. Какой из методов вычисления определенного интеграла является самым точным и как это определяется?

Лабораторная работа № 8

РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА

Дифференциальные уравнения

Общий вид дифференциального уравнения 1-го порядка:

$$F(x, y, y') = 0\,. \tag{8.1}$$

Если это уравнение разрешимо относительно y', то

$$y' = f(x, y) \quad \text{или} \quad dy = f(x, y)dx\,. \tag{8.2}$$

Общим решением уравнения (8.1) называется функция

$$y = \varphi(x, C) \tag{8.3}$$

от x и произвольной постоянной C, обращающая это уравнение в тождество.

Частным решением уравнения (8.1) называется решение, полученное из общего решения (8.3) при фиксированном значении С:

$$y = \varphi(x, C_0)\,, \tag{8.4}$$

где C_0 – фиксированное число.

Задача Коши

Найти решение $y = f(x, y)$ дифференциального уравнения (8.1), удовлетворяющее заданным начальным условиям: $y = y_0$ при $x = x_0$. Другими словами, найти интегральную кривую уравнения (8.1), проходящую через точку $M_0(x_0, y_0)$.

Метод Эйлера (метод Рунге–Кутта 1-го порядка)

Разобьем $[a, b]$ на n равных частей – ***элементарных отрезков***; $x_0, x_1, \ldots, x_n$ будем называть ***узлами сетки***; $h = (b - a) / n$ – ***шаг сетки***.

Очевидно, что $x_i = a + i \cdot h$, $i = \overline{0,1,...,n}$; $x_0 = a$, $x_n = b$.

Заменим в уравнении (8.1) y' в точке x_i ее приближенной оценкой – отношением приращений (это следует из определения производной):

$$y_i' \approx \frac{\Delta y_i}{\Delta x_i} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{y_{i+1} - y_i}{h}.$$

Тогда получаем

$$\frac{y_{i+1} - y_i}{h} \approx f(x_i, y_i).$$

Отсюда формула Эйлера:

$$y_{i+1} \approx y_i + h \cdot f(x_i, y_i). \qquad (8.5)$$

$x_i = a + i \cdot h$, $i = \overline{0,1,...,n-1}$ – номер узла.

Зная y_0 в точке x_0 (начальное условие) можно найти y_1, затем, используя уже известные значения x_1 и y_1, вычислить x_2 и y_2 и так далее.

Если функция $|f| \le M_1$, $\left|\frac{df}{dx}\right| \le M_2$, $\left|\frac{df}{dy}\right| \le M_3$ для $x \in [a,b]$, $y \in [y_0, Y]$, то имеет место неравенство

$$|y(x_i) - y_i| \le \frac{M_4 h}{2M_3} \cdot e^{M_3(x_i - x_0)},$$

где $M_4 = M_2 + M_1 M_3$, $h = \max\limits_{0 \le i \le n-1} h_i$.

Оценка имеет лишь теоретическое значение. На практике чаще всего пользуются двойным пересчетом на ЭВМ: расчет на отрезке $[x_i;\ x_{i+1}]$ (повторяют с шагом $h_i/2$) и погрешность более точного решения y^*_{i+1} (при шаге $h_i/2$) оценивают по формуле:

$$\left|y^*_{i+1} - y(x_{i+1})\right| \approx \left|y^*_{i+1} - y_{i+1}\right|. \qquad (8.6)$$

Рассмотрим геометрическую иллюстрацию метода Эйлера (рис. 8.1). В координатах (x; y) отобразим известные данные: отрезок [a; b] на оси X и начальное условие y_0 – точка A с координатами (a; y_0). Отрезок [a; b] разобьем на n равных частей, получим узлы равномерной сетки $a = x_0, x_1, x_2, \ldots, x_n = b$. Вычислим значения первой производной искомой функции в точке A, используя координату этой точки и исходное уравнение (8.3):

$$y'(x_0) = f(x_0, y_0) = \mathrm{tg}\alpha_0 .$$

Полученное значение позволяет построить касательную к искомой функции в точке A. Эту касательную можно использовать для вычисления приближенного значения искомой функции в новом узле x_1 (кривую $y(x)$ заменяем отрезком AB на элементарном отрезке $[x_0, x_1]$).

Зная (x_1; y_1), можно аналогично получить новую точку (x_2; y_2) и т. д.

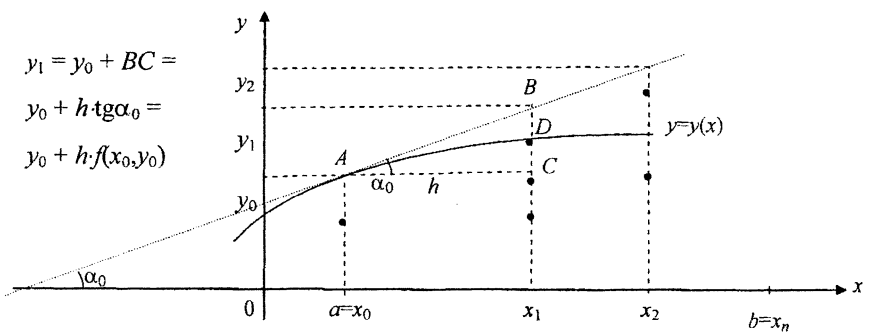


Рис. 8.1. Геометрическая иллюстрация метода Эйлера

Из геометрической иллюстрации следует, что:

1. На каждом шаге есть погрешность (на рис. 8.1 это отрезок *BD*). Погрешность тем больше, чем больше шаг.

2. Ошибка может накапливаться.

Формула Эйлера (8.3) имеет погрешность метода $\delta_м \approx O(h^2)$.

Для практического выбора h с целью обеспечения заданной точности решения задачи ε применяется следующий прием.

Выполняются 2 расчета: с n и $2n$ узлами. Если полученные значения функции во всех узлах отличаются не более чем на ε, задача считается решенной. Если нет, число узлов вновь удваивают и опять сравнивают полученные значения функций.

Таким образом, расчет продолжается до достижения условия

$$\delta = \max_{i=1,n} | y_i^n - y_i^{2n} | \le \varepsilon . \qquad (8.7)$$

Значение n может достигать большой величины – более 1000. Чтобы не печатать столько значений функции, в алгоритме решения ОДУ методом Эйлера нужно предусмотреть печать не всех рассчитанных значений, а только части их, например, 10 значений, распределенных равномерно по всему отрезку.

Пример 1. Дано уравнение: $y' - 2y + x^2 = 1$.

Найти решение для отрезка [0; 1], если $y(0) = 1$.

Выберем $n = 10$, тогда шаг $h = (1 - 0) / 10 = 0{,}1$.

Запишем уравнение в каноническом виде $y' = f(x, y) = 1 + 2y - x^2$.

Начальная точка $x_0 = 0$, $y_0 = 1$.

Вычислим первую точку:

$$y_1 = y_0 + h \cdot f(x_0; y_0) = 1 + 0{,}1 \cdot f(0; 1) = 1 + 0{,}1 \cdot (1 + 2 \cdot 1 - 0^2) = \\ = 1 + 0{,}1 \cdot 3 = 1{,}3.$$

$$x_1 = x_0 + h = 0 + 0{,}1 = 0{,}1.$$

Вычислим вторую точку:

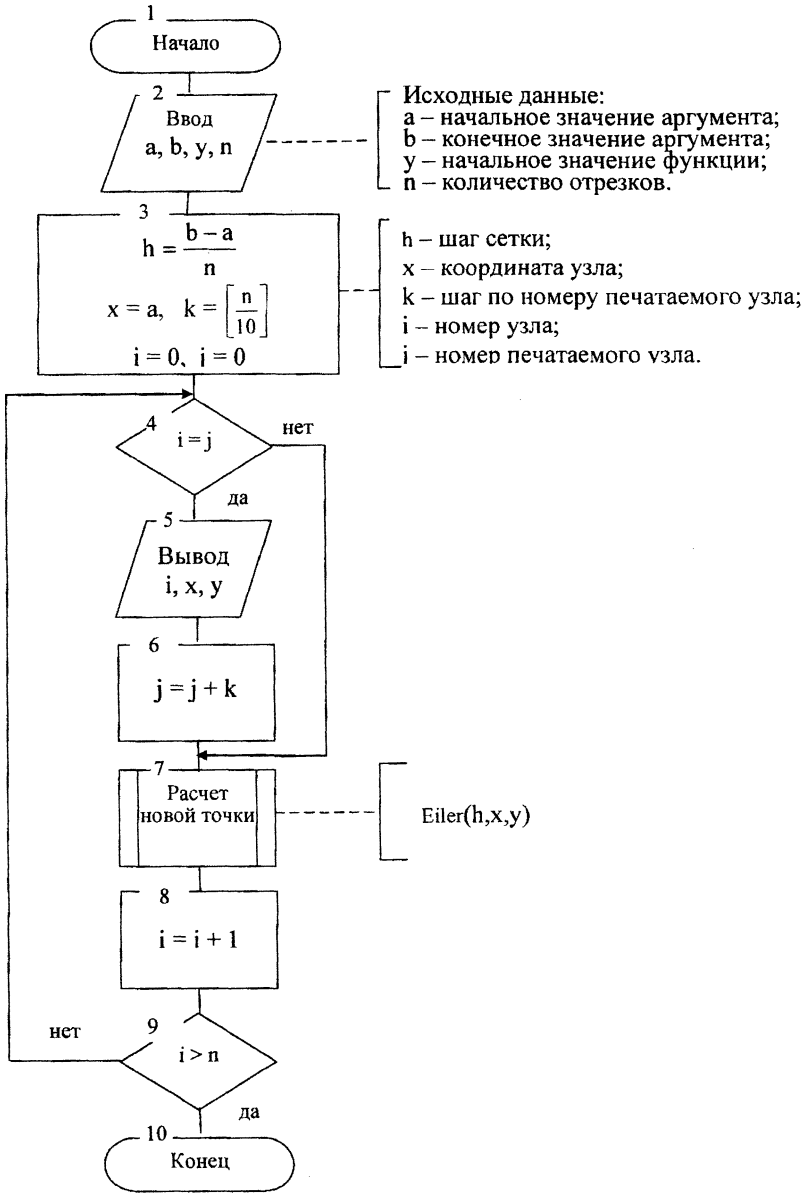
$$y_2 = y_1 + h \cdot f(x_1;\ y_1) = 1{,}3 + 0{,}1 \cdot f(0{,}1;1{,}3) = 1{,}3 + 0{,}1 \cdot (1 + 2{,}6 - 0{,}01) =$$
$$= 1{,}3 + 0{,}1 \cdot 3{,}59 = 1{,}659.$$

$$x_2 = x_1 + h = 0{,}1 + 0{,}1 = 0{,}2\ .$$

Аналогично нужно вычислить еще восемь точек (т. к. выбрано $n = 10$).

Блок-схема алгоритма расчета новой точки методом Эйлера

Блок-схема алгоритма решения ОДУ 1-го порядка методом Эйлера



Модифицированный метод Эйлера (метод Рунге–Кутта 2-го порядка)

Пусть требуется найти решение задачи Коши:

$$y' = f(x,y), \quad a \le x \le b, \quad y(a) = y_0.$$

Как и в методе Эйлера, на отрезке $[a;b]$ зададим конечное множество точек $\{x_i\}_{i=0}^{n}$, ($a = x_0 < x_1 < ... < x_N = b$). По модифицированному методу Эйлера вычисление приближенного решения $y_i \approx y(x_i)$ проводится следующим образом (рис. 8.2).

Вначале вычисляется первое приближение:

$$\tilde{y}_{i+1} = y_i + h_i f(x_i, y_i).$$

Затем находится более точное приближение:

$$y_{i+1} = y_i + h_i \frac{f(x_i, y_i) + f(x_{i+1}, \tilde{y}_{i+1})}{2}.$$

Остаточный член на каждом шаге в модифицированном методе Эйлера имеет порядок $O(h^3)$.

A – начальная точка;

L_1 – касательная к $y(x)$ в точке A;

L_2– касательная к $y(x)$ в середине элементарного отрезка;

L_3 параллельно L_2 через т. A

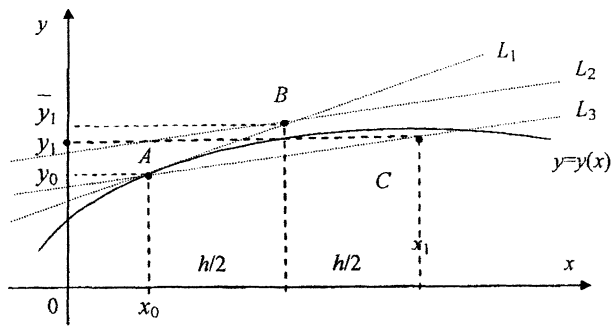


Рис. 8.2. Геометрическая иллюстрация модифицированного метода Эйлера

Оценка погрешности может быть получена с помощью двойного пересчета на ЭВМ. Расчет повторяют с шагом $h_i/2$ и погрешность более точного решения y_i^* оценивают приближенно:

$$\left|y_i^* - y(x_i)\right| \approx \frac{1}{3}\left|y_i^* - y_i\right|.$$

Расчетные формулы:

$\overline{y_1} = y_0 + \frac{h}{2}\cdot f(x_0, y_0)$ – значение функции в середине отрезка $[x_0; x_1]$.

$y_1 = y_0 + h\cdot f(x_0 + \frac{h}{2}, \overline{y_1})$ – значение функции в конце отрезка $[x_0; x_1]$.

Формула модифицированного метода Эйлера

$$y_{i+1} = y_i + h\cdot f(x_i + \frac{h}{2}, y_i + \frac{h}{2}\cdot f(x_i, y_i)), \qquad (8.8)$$

где $I = 0, 1, ..., n-1$ – номер узла;

$x_i = a + i\cdot h$ – координата узла;

$y_0 = y(x_0)$ – начальное условие.

Алгоритм решения ОДУ модифицированным методом Эйлера отличается от описанного ранее алгоритма метода Эйлера, представленного на блок-схеме, только алгоритмом расчета новой точки.

Погрешность метода $\delta \approx O(h^3)$.

Усовершенствованный метод Эйлера можно еще более уточнить, применяя **итерационную обработку** каждого значения y_i. Вначале вычисляется

$$y_{i+1}^{(0)} = y_i + h_i f(x_i, y_i),$$

а затем это приближение уточняется по формуле

$$y_{i+1}^{(k+1)} = y_i + h_i/2(f(x_{i+1}, y_{i+1}^{(k)}) + f(x_i, y_i)).$$

Итерации продолжают до тех пор, пока в пределах требуемой точности два последовательных приближения $y_{i+1}^{(k-1)}$ и $y_{i+1}^{(k)}$ не совпадут. После чего $y_{i+1}^{(k)}$ принимается за приближенное значение $y(x_{i+1})$.

Пример 2. Решим ранее рассмотренное уравнение (пример 1) модифицированным методом Эйлера.

$y' - 2y + x^2 = 1$, $x \in [0; 1]$, $y(0) = 1$.

Пусть $n = 10$, $h = (1 - 0) / 10 = 0{,}1$.

Начальная точка $x_0 = 0$, $y_0 = 1$.

Рассчитаем первую точку:

$$y_1 = y_0 + h \cdot f\left(x_0 + \frac{h}{2}; y_0 + \frac{h}{2} \cdot f(x_0; y_0)\right) = 1 + 0{,}1 \cdot f\left(0 + \frac{0{,}1}{2}; 1 + \frac{0{,}1}{2} \cdot f(0; 1)\right) =$$

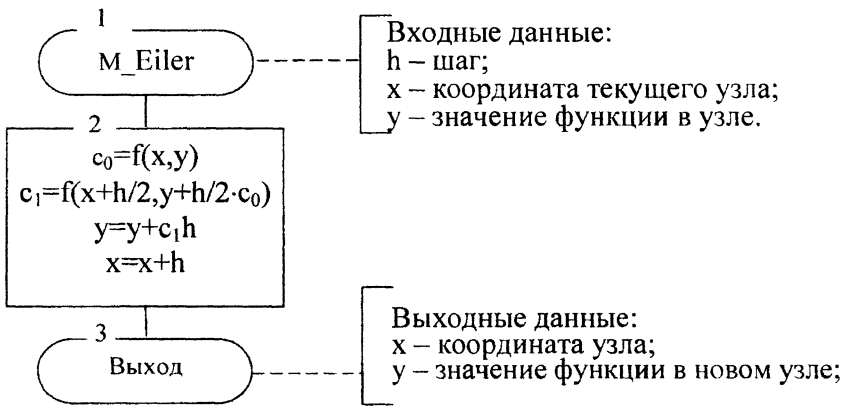
$$= 1 + 0{,}1 \cdot f(0{,}05; 1 + 0{,}05 \cdot (1 + 2 \cdot 1 - 0^2)) = 1 + 0{,}1 \cdot f(0{,}05; 1{,}15) =$$

$$= 1 + 0{,}1 \cdot (1 + 2 \cdot 1{,}15 - 0{,}05^2) = 1{,}32975.$$

$$x_1 = x_0 + h = 0{,}1.$$

Аналогично рассчитаем 2, 3, ..., 10-ю точки.

Блок-схема алгоритма расчета новой точки модифицированным методом Эйлера



Метод усредненных точек

Пусть требуется найти решение задачи Коши:

$$y' = f(x,y), \quad a \le x \le b, \quad y(a) = y_0.$$

Как и в методе Эйлера, на отрезке $[a;b]$ зададим конечное множество точек $\{x_i\}_{i=0}^{n}$, $(a = x_0 < x_1 < \ldots < x_n = b)$. По усовершенствованному методу ломаных сначала вычисляются промежуточные значения

$$x_{i+\frac{1}{2}} = x_i + \frac{h}{2}, \quad y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f_i,$$

а затем полагают, что

$$y_{i+1} = y_i + h f_{i+\frac{1}{2}},$$

где $f_{i+\frac{1}{2}} = f\left(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}\right)$.

В этом методе для повышения точности используется усредненное значение производной на рассматриваемом отрезке:

$$y_{i+1} = y_i + h\frac{y_i' + y_{i+1}'}{2} = y_i + \frac{h}{2}[f(x_i,y_i) + f(x_{i+1},y_{i+1})].$$

В приведенной формуле y_{i+1} входит в обе части уравнения и не может быть выражено явно. Чтобы обойти эту трудность, в правую часть вместо y_{i+1} подставляется значение, рассчитанное по формуле Эйлера (8.5):

$$\overline{y_{i+1}} = y_i + h f(x_i,y_i).$$

Получаем формулу исправленного метода Эйлера

$$y_{i+1} = y_i + \frac{h}{2} \cdot [f(x_i,y_i) + f(x_{i+1}, y_i + h \cdot f(x_i,y_i))], \; i = \overline{0,n-1}, \quad (8.9)$$

где $I = 0, 1, ..., n-1$ – номер узла;

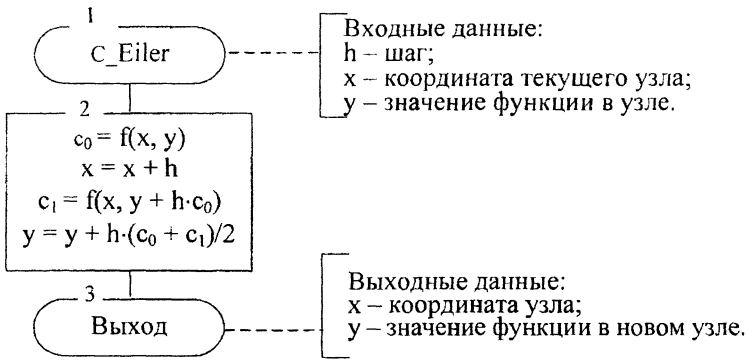
$x_i = a + i \cdot h$ – координата узла;

$y_0 = y(x_0)$ – начальное условие.

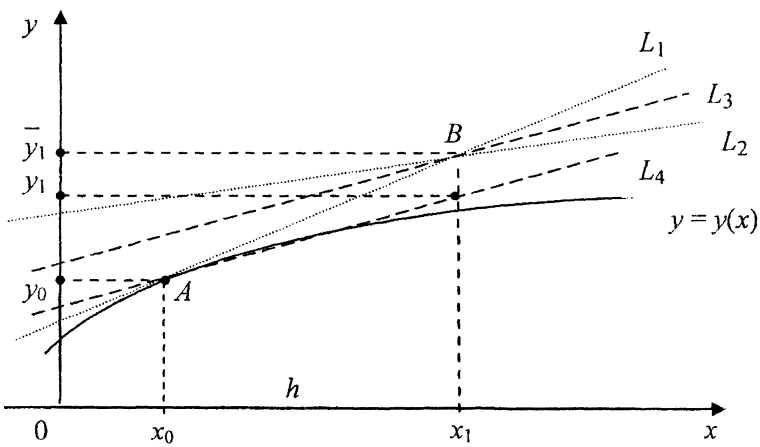
Погрешность исправленного метода Эйлера $\delta_м \approx O(h^3)$.

Алгоритм решения ОДУ методом усредненных точек отличается от описанного ранее алгоритма метода Эйлера, представленного на блок-схеме, только алгоритмом расчета новой точки.

Блок-схема алгоритма расчета новой точки усредненным методом Эйлера



Геометрическая иллюстрация усредненного метода Эйлера



Здесь L_1 – касательная к $y(x)$ в начальной точке A с $\operatorname{tg}\alpha_0 = f(x_0; y_0)$;

т. B – значение $\bar{y}_1$ вычисляется по формуле Эйлера;

L_2 – касательная к $y(x)$ в точке B с $\operatorname{tg}\alpha_1 = f(x_1, \bar{y}_1)$;

L_3 – прямая через B со среднеарифметическим углом наклона;

L_4 – прямая, параллельная L_3, проведенная через точку A.

Пример 3. Решим ранее рассмотренное уравнения (пример 1) усредненным методом Эйлера.

$y' - 2y + x^2 = 1, x \in [0; 1], y(0) = 1.$

Пусть $n = 10$, $h = (1 - 0) / 10 = 0{,}1$.

Начальная точка $x_0 = 0$, $y_0 = 1$.

Рассчитаем первую точку:

$$y_1 = y_0 + \frac{h}{2}\cdot\left[f(x_0; y_0) + f(x_0 + h;\ y_0 + h\cdot f(x_0; y_0))\right] =$$

$$= 1 + \frac{0{,}1}{2}\cdot\left[f(0;1) + f(0 + 0{,}1;\ 1 + 0{,}1\cdot f(0, 1))\right] =$$

$$= 1 + \frac{0{,}1}{2}\cdot[(1 + 2\cdot 1 - 0^2) + f(0{,}1;\ 1 + 0{,}1\cdot(1 + 2 - 0^2))] =$$

$$= 1 + 0{,}05\cdot[3 + f(0{,}1; 1{,}3)] = 1 + 0{,}05\cdot[3 + (1 + 2\cdot 1{,}3 - 0{,}1^2)] =$$

$$= 1 + 0{,}05\cdot[3 + 3{,}59] = 1 + 0{,}05\cdot 6{,}59 = 1{,}3295.$$

$$x_1 = x_0 + h = 0{,}1.$$

Аналогично можно вычислить значения функции во 2, 3, …, 10-й точках.

Метод Рунге–Кутта 4-го порядка

На практике наибольшее распространение получил метод Рунге–Кутта 4-го порядка, в котором усреднение проводится по трем точкам. Формула Эйлера на каждом отрезке используется 4 раза: в начале отрезка, дважды в его середине и в конце отрезка.

Расчетные формулы метода Рунге–Кутта 4-го порядка для дифференциального уравнения (8.3) имеют вид

$$y_{i+1} = y_i + \Delta y_i \ \ (i = 1, 2, \ldots), \ \ \Delta y_i = \frac{1}{6}\left(k_0 + 2k_1 + 2k_2 + k_3\right),$$

где $k_0 = hf(x_i; y_i)$;

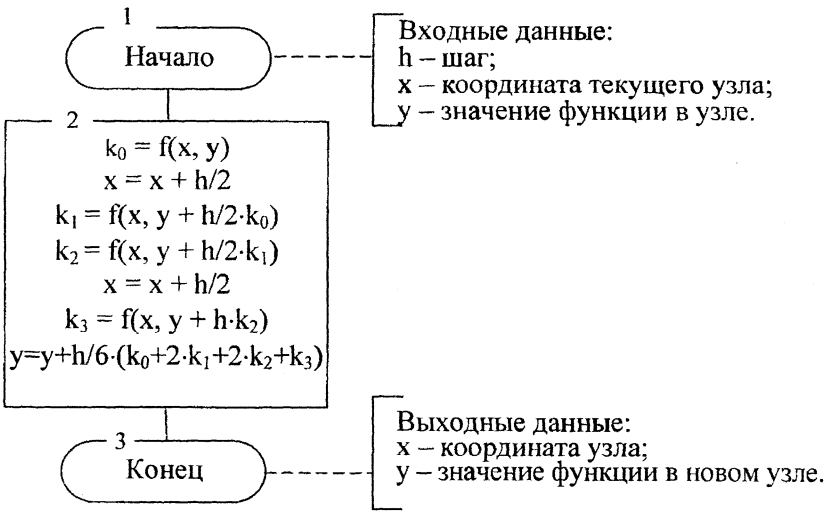
$$k_1 = hf(x_i + \frac{h}{2};\ y_i + \frac{k_0}{2});$$

$$k_2 = hf(x_i + \frac{h}{2}; y_i + \frac{k_1}{2});$$

$$k_3 = hf(x_i + h; y_i + k_3).$$

Погрешность метода $\delta_м \approx O(h^5)$.

Блок-схема алгоритма расчета новой точки методом Рунге–Кутта 4-го порядка



Задание

Решить дифференциальные уравнения 1-го порядка методами:
1) Эйлера;
2) модифицированным методом Эйлера;
3) Рунге–Кутта.

Варианты заданий

№ варианта	Дифференциальное уравнение	Начальные условия	Интервал интегрирования	Точное решение
1	$y' + y \cdot \operatorname{tg} x = \dfrac{1}{\cos x}$	$y(\pi) = 5$	$[\pi; 2\pi]$	$y = -5\cos x + \sin x$
2	$y' + 3y = e^{2x} y^2$	$y(0) = 1$	$[0; 2]$	$y = e^{-2x}$
3	$y' - \dfrac{y}{x-3} = \dfrac{y^2}{x-3}$	$y(1) = -2$	$[1; 1{,}5]$	$y = \dfrac{x-3}{2-x}$
4	$y' = 2y - x + e^x$	$y(0) = -1$	$[0; 1{,}5]$	$y = \dfrac{1}{2}x - e^x + \dfrac{1}{4}\left(1 - e^{2x}\right)$
5	$(x^2 - x)y' + y = x^2(2x-1)$	$y(-2) = 2$	$[-2; -1]$	$y = x^2 - \dfrac{3x}{x-1}$
6	$xy' = x \cdot \sin\dfrac{y}{x} + y$	$y(2) = \pi$	$[2; 3]$	$y = 2x \cdot \operatorname{arctg}(x/2)$
7	$xy' = y(1 + \ln y - \ln x)$	$y(1) = e^2$	$[1; 1{,}5]$	$y = x \cdot e^{2x}$
8	$y' = y/\left(3x - y^2\right)$	$y(0) = 1$	$[0; 1]$	$x = y^2 - y^3$
9	$x(y' - y) = e^x$	$y(1) = 0$	$[1; 2]$	$y = e^x \ln x$
10	$xy' - 2y = 2x^4$	$y(1) = 0$	$[1; 2]$	$y = x^4 - x^2$
11	$y' = 2x(x^2 + y)$	$y(0) = 0$	$[0; 1]$	$y = x^2 + 1 - e^{x^2}$
12	$y' - y = e^x$	$y(0) = 1$	$[0; 1]$	$y = (x+1)e^x$
13	$yx' + x = 4y^3 + 3y^2$	$y(2) = 1$	$[2; 3]$	$x = y^3 + y^2$
14	$x^2 y' + xy + 1 = 0$	$y(1) = 0$	$[1; 2]$	$y = -(\ln x)/x$
15	$y = x(y' - x \cdot \cos x)$	$y(\pi/2) = 0$	$[\pi/2; 1]$	$y = (\sin x - 1)x$
16	$y' + y \cdot \operatorname{tg} x = \sec x$	$y(0) = 0$	$[0; 1]$	$y = \sin x$

Контрольные вопросы

1. В чем заключается задача Коши для решения обыкновенных дифференциальных уравнений 1-го порядка?
2. Насколько модифицированный метод Эйлера точнее простого?
3. В чем отличие метода усредненных точек от модифицированного метода Эйлера?

ЛИТЕРАТУРА

1. Паскаль для персональных компьютеров: справочное пособие / Ю.С. Бородич [и др.]. – Минск: Вышэйшая школа: БФ ГИТМП «НИКА», 1991. – 365 с.

2. Фигурнов, В.Э. IBM PC для пользователя / В.Э. Фигурнов. – 6-е изд., перераб. и доп. – М.: Инфра-М, 1995. – 432 с.

3. Фаронов, В.В. Турбо Паскаль 7.0. Практика программирования / В.В. Фаронов. – СПб.: БХВ-Петербург, 2004. – 1056 с.

4. Демидович, Б.П. Численные методы анализа / Б.П. Демидович, И.А. Марон, Э.З. Шувалова. – М.: Гос. изд-во физ.-мат. лит., 1967. – 140 с.

5. Демидович, Б.П. Основы вычислительной математики / Б.П. Демидович, И.А. Марон. – М.: Гос. изд-во физ.-мат. лит., 1966. – 112 с.

6. Пантелеев, А.В. Методы оптимизации в примерах и задачах: учебное пособие / А.В. Пантелеев, Т.А. Летова. – 2-е изд., испр. – М.: Высшая школа, 2005. – 119 с.

7. Самарский, А.А. Численные методы: учебное пособие для вузов / А.А. Самарский, А.В. Гулин. – М.: Наука, 1989. – 432 с.

8. Березин, И.С. Методы вычислений / И.С. Березин, Н.П. Жидков. – М., 1962. – 217 с.

9. Волков, Е.А. Численные методы: учебное пособие для вузов / Е.А. Волков. – 2-е изд., испр. – М.: Наука,1987. – 248 с.

10. Турчак, Л.И. Основы численных методов / Л.И. Турчак. – М.: Наука, 1987. – 250 с.

11. Марчук, Г.И. Методы вычислительной математики / Г.И. Марчук. – М.: Наука, 1989. – 456 с.

12. Балашевич, В.А. Основы математического программирования / В.А. Балашевич. – Минск: Вышэйшая школа, 1985. – 173 с.

13. Исаков, В.Б. Элементы численных методов: учебное пособие для вузов / В.Б. Исаков. – М.: Академия, 2003. – 192 с.

Учебное издание

ИНФОРМАТИКА

Лабораторный практикум
для студентов специальностей
1-43 01 04 «Тепловые электрические станции»,
1-53 01 04 «Автоматизация и управление энергетическими процессами», 1-43 01 08 «Паротурбинные установки атомных электрических станций»

В 2 частях

Часть 2

Составители:
ТАРАСЕВИЧ Леонид Александрович
ПРОНКЕВИЧ Елена Васильевна

Редактор В.О. Кутас
Компьютерная верстка Н.А. Школьниковой

Подписано в печать 08.11.2011.
Формат $60\times84^{1}/_{16}$. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 5,0. Уч.-изд. л. 3,91. Тираж 100. Заказ 386.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.