Государственное образовательное учреждение высшего профессионального образования

РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Истомин Е.П., Слесарева Л.С.

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ МАТЕМАТИЧЕСКИХ ЗАДАЧ

Учебное пособие

Санкт-Петербург, 2015 г.

Истомин Е.П., Слесарева Л.С. Алгоритмизация и программирование математических задач. Учебное пособие.- СПб.: ООО «Андреевский издательский дом», 2015 - 58 с.

В учебном пособии представлены алгоритмы и программы, написанные на алгоритмическом языке Pascal наиболее распространенных математических задач, встречающихся в экономических и инженерных расчетах.

Учебное пособие предназначено для студентов, обучающихся по специальности «Прикладная информатика».

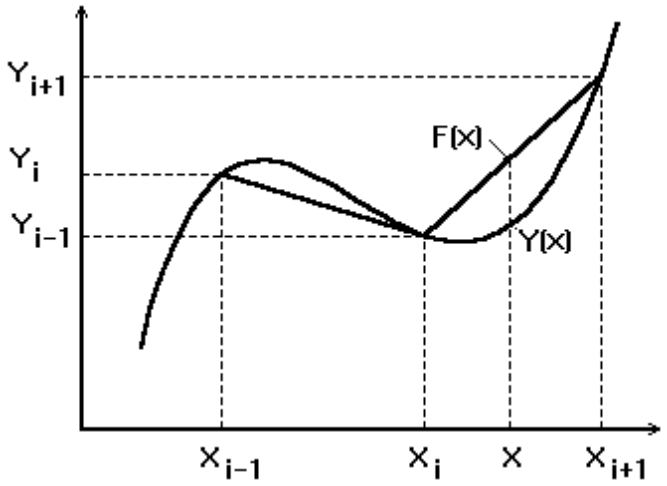
Истомин Е.П., Слесарева Л.С..
Алгоритмизация математических задач.
ООО «Андреевский издательский дом»
197738, Санкт-Петербург, пос. Репино, Приморское шоссе, д. 394
E-mail: **biom@nm.ru**
Подписано в печать 15.10.2015 г.
Печ. листов 3,65. Тираж 200 экз.

1. ПРИБЛИЖЕНИЕ ФУНКЦИЙ

Одной из важнейших задач в процессе математического моделирования является вычисление значений функций, входящих в математическое описание модели. Часто функция задается табличным способом, когда известны отдельные значения функции и возможно ее производные при дискретных значениях аргумента $\{X_i, Y_i, Y_i^1, ..., Y_i^{(n)} / i = \overline{0,n}\}$. Здесь и в дальнейшем предполагается упорядоченность дискретных значений аргумента $X_1 < X_2 < X_3 < ... < Xn$. Также используются обозначения $Y(X_i) = Y_i$, $Y^1(X_i) = Y_i^1$, $Y^{(2)}(X_i) = Y_i^{(2)}$,... ,$Y^{(n)}(X_i) = Y_i^{(n)}$. Требуется найти значение неизвестной функции Y(X) для любого значения $X \in [X_0, X_n]$.

Эта задача решается путем приближенной замены функции Y(X) более простой функцией F(X), легко вычисляемой на ЭВМ, которую используют не только для определения численных значений Y(X), но и для аналитических выкладок при теоретическом исследовании моделируемого процесса. Если удается подобрать такую функцию F(X), что во всех точках $i = \overline{0,n}$ выполняется равенство $Y(X_i) = F(X_i)$, то функцию F(X) называют *интерполяционной*, а множество $\{X_i, Y_i / i = \overline{0,n}\}$ - *узлами* интерполяции.

1.1. Линейная интерполяция



Простейшим методом приближения является линейная интерполяция. В этом методе используется кусочно-линейная функция F(X) соединяющая интерполяционные узлы.

Исходными данными для метода линейной интерполяции является таблица:

$\{X_i, Y_i / i = \overline{0,n}\}$. (1-1)

Требуется найти значение неизвестной функции Y(X) в заданной точке $X \in [X_i, X_{i+1}]$, где $X_i < X < X_{i+1}$.

В качестве решения за искомую величину Y(X) принимается значение функции F(X), которое вычисляется по формуле:

$$F(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)\ ,\ i = \overline{0,n}. \qquad (1\text{-}2)$$

Пример 100. Разработать алгоритм и программу линейной интерполяции для заданных значений $\{X_i,\ Y_i\ /\ i = \overline{0,n}\}$. Построение алгоритма начнем с таблицы имен, содержащей наименование величин в вычислительной схеме, их смысл, тип и соответствующее имя в структурограмме и программе.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число промежутков интерполирования	integer
x i	X[i]	Значение узлов интерполирования	real
y i	Y[i]	Значение функции в узлах интерполир.	real
x	XP	Произвольное значение X ∈ [X0, Xn]	real
F(x)	YP	Значение интерполируемой функции в точке x	real
-	Q	Символьная переменная, значения которой Y/N определяют продолжение Y или окончание N вычислений	char

Структурограмма

Ввод N, Xi, Yi, где i = 0 .. n.
Пока Q равно 'Y' или 'y' решать задачу интерполяции:
Ввод значения XP.
Анализ: XP ∈ [X0, Xn]
False: Вывод сообщения: 'X - вне интервала интерполяции'
True: Начнем анализ с интервала [X0, X1], то есть I = 0
Пока XP не опадет в интервал [Xi, Xi+1] делать:
Взять следующий интервал: I = I + 1
Используя уравнение прямой рассчитать YP; Вывести найденное значение F(X) на монитор.
Запрос на повторное вычисление функции.

На первом этапе работы программы вводится объём выборки N и массивы X и Y. После ввода исходных данных пользователь должен указать значение ар-

гумента X для которого будет вычисляться интерполяционная функция F(X). Значение X нужно проверить на $X \in [X_0, Xn]$. Если это условие выполнено, то находится подинтервал $[X_i, X_{i+1}]$, которому принадлежит $X \in [X_i, X_{i+1}]$. Используя (8-2), вычисляется значение F(X) и выводиться на экран монитора.

```
PROGRAM PR100;        {ЛИНЕЙНАЯ ИНТЕРПОЛЯЦИЯ}
CONST G = 20;     {МАКСИМАЛЬНОЕ ЧИСЛО УЗЛОВ ИНТЕРПОЛЯЦИИ}
VAR   I, N: INTEGER; XP, YP: REAL; Q: CHAR; X, Y: ARRAY[0..G] OF
REAL;
BEGIN
WRITELN( 'ВВЕДИТЕ ЧИСЛО УЗЛОВ ИНТЕРПОЛЯЦИИ, N'); READ(N);
WRITELN( 'ВВЕДИТЕ ЗНАЧЕНИЯ УЗЛОВ ИНТЕРПОЛЯЦИИ X, Y' );
N:=N-1;
FOR I := 0 TO N DO READ (X[I], Y[I]); Q := 'Y';
WHILE (Q = 'Y') OR (Q = 'y')
DO     BEGIN
WRITELN( 'ВВЕДИТЕ ЗНАЧЕНИЯ X' ); READLN(XP);
IF (X[0] <= XP) AND (XP <= X[N])
THEN BEGIN  I := 0;      {ПОИСК ИНТЕРВАЛА XI < XP < XI + 1}
 WHILE  NOT((XP <= X[I+1]) AND (XP >= X[I]))  DO I := I + 1;
 YP := Y[I] + (Y[I+1] - Y[I]) / (X[I+1]-X[I]) * (XP - X[I]);
 WRITELN('X = ', XP:8:5,', F(X) = ',YP:8:5)
 END   {THEN}
ELSE WRITELN( 'X-ВНЕ ИНТЕРВАЛА ИНТЕРПОЛЯЦИИ.' );
WRITELN( ' БУДУТ ЕЩЕ РАСЧЕТЫ: Y/N? ' ); READLN(Q)
END {WHILE}
END.
```

1.2. Интерполирование полиномом Лагранжа

Основным недостатком линейной интерполяции является разрыв первой производной в узлах. Этот недостаток устраняется, если в качестве интерполяционной функции взять полином. Доказано, что для всякой таблицы вида $\{X_i, Y_i / i = \overline{0,n}\}$ существует единственный полином степени не выше n, проходящий через все узлы таблицы:

$$F(X) = A_0 + A_1 X + A_2 X^2 + ... + A_n X^n,$$

(1-3)

Лагранж предложил представить этот полином (1-3) в следующем виде:

$$F(x) = \sum_{i=0}^{n} L_i \cdot y_i \text{ ; где } L_i = \prod_{\substack{i,j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \qquad (1\text{-}4)$$

Формулу (1-4) называют полиномом Лагранжа.

Пример 101. Разработать алгоритм и программу интерполяции полиномом Лагранжа таблицы $\{X_i, Y_i / i = \overline{0,n}\}$.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число промежутков интерполирования	integer
x i	X[i]	Значение узлов интерполирования	real
y i	Y[i]	Значение функции в узлах интерполир.	real
x	XP	Произвольное значение $X \in [X0, Xn]$	real
F(x)	YP	Значение интерполируемой функции в точке x	real
-	Q	Символьная переменная, значения которой Y/N определяют продолжение Y или окончание N вычислений	char
Li	SL	Множитель Лагранжа	real

Структурограмма

Ввод N, Xi, Yi, где i = 0 .. n; Q = 'Y'

Пока Q равно 'Y' или 'y' решать задачу интерполяции:

Ввод значения XP.

Анализ: $XP \in [X_0, Xn]$

False — Вывод сообщения: 'X-вне интервала интерполяции'

True — Расчет формулы Лагранжа: YP = 0.

Для I от 0 до N с шагом 1 делать:

Расчет множителя Лагранжа: SL = 1.

Для J от 0 до N с шагом 1 делать:

I равно J

False — SL = SL * (XP - X[J])/(X[I] - X[J])

YP = YP + SL * Y[I]

Вывести найденное значение F(X) на монитор

Запрос на повторное вычисление функции

Отличие алгоритма в том, что не нужно находить интервал $X \in [X_i, X_{i+1}]$, так как полином F(X) проходит через все точки таблицы. Достаточно убедиться в $X \in [X_0, Xn]$, после чего рассчитать F(X) по формуле (1-4).

```
PROGRAM PR101;        {ИНТЕРПОЛЯЦИЯ ПОЛИНОМОМ ЛАГРАНЖА}
CONST  G = 20;    {МАКСИМАЛЬНОЕ ЧИСЛО УЗЛОВ ИНТЕРПОЛЯЦИИ}
VAR  I, J, N: INTEGER; XP, YP, SL: REAL; Q: CHAR;
X, Y: ARRAY[0..50] OF REAL;
BEGIN
WRITELN( 'ВВЕДИТЕ ЧИСЛО УЗЛОВ ИНТЕРПОЛЯЦИИ, N' ); READ(N);
WRITELN ('ВВЕДИТЕ ЗНАЧЕНИЯ УЗЛОВ ИНТЕРПОЛЯЦИИ X, Y' );
N:=N-1;
FOR I := 0 TO N DO READ (X[I],Y[I]);
Q := 'Y';              {ФЛАГ: Y - БУДУТ ЕЩЕ ВЫЧИСЛЕНИЯ, N - НЕТ}
WHILE (Q = 'Y') OR (Q = 'y')
DO BEGIN
    WRITELN('ВВЕДИТЕ ЗНАЧЕНИЕ X'); READLN(XP);
    IF  (X[0]<=XP)  AND  (XP<=X[N])   {ПРОВЕРКА НА ПРИНАДЛЕЖ-
НОСТЬ}
    THEN BEGIN                {ИНТЕРВАЛУ ИНТЕРПОЛЯЦИИ}
     YP := 0;
      FOR I := 0 TO N
          DO BEGIN
              SL := 1;
              FOR  J:=0  TO N
              DO IF  I <> J
                   THEN  SL := SL * (XP - X[J]) / (X[I] - X[J]);
              YP := YP + SL * Y[I]
              END; {FOR I}
          WRITELN('X= ',XP:8:5,', F(X)= ',YP:8:5)
          END   {THEN}
    ELSE WRITELN('X - ВНЕ ИНТЕРВАЛА ИНТЕРПОЛЯЦИИ.');
    WRITELN('БУДУТ ЕЩЕ РАСЧЕТЫ: Y/N?'); Q:=' '; READLN(Q)
    END  {WHILE}
END.
```

1.3. Выбор и нахождение параметров эмпирической формулы

Иногда таблицы экспериментальных данных получены со значительными погрешностями и нет смысла использовать интерполяцию для обработки результатов. В этом случае для заданной таблицы $\{x_i, y_i / i = \overline{0,n}\}$ требуется подобрать простую эмпирическую формулу F(x, a, b), имеющую параметры 'a' и 'b' такую, чтобы значения y_i весьма незначительно отличались от соответствующих значе-

ний $F(x_i, a, b)$. В данном пособии для оценки степени близости функций $Y(x)$ и $F(x, a, b)$ использован критерий - наименьший суммарный квадрат отклонения:

$$G(a,b) = \sum_{i=1}^{n}\left(y_i - F(xi,a,b)\right)^2 = \sum_{i=1}^{n}\varepsilon_i^2 \rightarrow \min \qquad (1\text{-}8)$$

Если взять линейную функцию

$F(X, A_1, A_2) = A_1 X + A_2$,

(1-9)

то аналитическое решение этой задачи имеет вид под номером (1-10):

$$A_1 = -\frac{\sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} y_i - n\sum_{i=1}^{n} y_i \cdot x_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}; A_2 = \frac{\sum_{i=1}^{n} y_i - A_1 \cdot \sum_{i=1}^{n} x_i}{n}. \qquad (1\text{-}10)$$

Чтобы распространить это решение на более широкий класс нелинейных функций используют метод выравнивания. Суть этого метода состоит в преобразовании нелинейного уравнения к виду (8-9) путем введения новых переменных $X = \varphi(x, y)$, $Y = \chi(x, y)$ таким образом, чтобы $\chi(x, y) = A_1 \varphi(x, y) + A_2$ или $Y = A_1 X + A_2$. Для этого уравнения находятся A_1 и A_2 по формулам (1-10), а затем вычисляются параметры a, b исходной нелинейной функции. Эквивалентные преобразования для восьми формул приведены в табл. 1.

Выравнивание зависимостей **Таблица 1**

№	Эмпирическая формула	Способ выравнивания	
1	$y = a \cdot x + b$	X = x, Y = y	$a = A_1$, $b = A_2$
2	$y = a \cdot b^x$	X = x, Y = ln(y)	$a = \exp(A_2)$, $b = \exp(A_1)$
3	$y = \frac{1}{(a \cdot x + b)}$	X = x, Y = 1 / y	$a = A_1$, $b = A_2$
4	$y = a \cdot \ln(x) + b$	X = ln(x), Y = y	$a = A_1$, $b = A_2$
5	$y = a \cdot x^b$	X = ln(x), Y = ln(y)	$a = \exp(A_2)$, $b = A_1$
6	$y = a + \frac{b}{x}$	X = 1 / x, Y = y	$a = A_2$, $b = A_1$
7	$y = \frac{x}{a \cdot x + b}$	X = x, Y = x / y	$a = A_1$, $b = A_2$
8	$y = a \cdot e^{b \cdot x}$	X = x, Y = ln(y)	$a = \exp(A_2)$, $b = A_1$

Алгоритм следует начать с ввода таблицы $\{x_i, y_i / i = 1..\ n\}$. Затем нужно указать L номер формулы из таблицы 1. Для каждой из восьми формул существует своя ветвь вычислений, но суть действий одинакова. На экран выводиться внешний вид формулы. Далее, используя метод выравнивания, $\{x_i, y_i / i = \overline{0,n}\}$ преоб-

разуют в линейную таблицу $\{X_i, Y_i / i = \overline{0,n}\}$. Для этой таблицы находят A_1 и A_2 с помощью процедуры NMKP, в которой рассчитываются формулы (1-10). После нахождения A_1 и A_2 вычисляются параметры a и b. И, последнее, в выбранной ветви находится множество значений $\{F(x_i, a, b) / i = \overline{0,n}\}$, которые помещаются в массив Y2. Далее находится сумма квадратов отклонений по формуле (1-8). Найденное значение G, а также значения параметров a и b выводятся на экран монитора. Процесс повторяется до тех пор, пока пользователь при выборе формулы не введет 0, что означает конец расчетов.

Пример 103. Разработать алгоритм и программу нахождения параметров эмпирических формул для табл. 1.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Объем выборки	integer
x_i	X[I]	Исходные данные: аргумент x_i;	real
y_i	Y[I]	функция y_i.	real
X_i	X1[I]	Исходные данные, преобразованные к	real
Y_i	Y1[I]	линейному виду методом выравнивания	real
$F(x_i, a, b)$	Y2[I]	Значения $F(x_i, a, b)$, найденные по эмпирической формуле	real
G(a, b)	F	Суммарный квадрат отклонения	real
a, b	A, B	Параметры эмпирической формулы	real
A_1, A_2	A1,A2	Коэффициенты приведенного линейного уравнения	real
$\Sigma_i y_i$	SY		real
$\Sigma_i x_i$	SX	Суммы необходимые для расчета A1 и	
$\Sigma_i x_i \cdot y_i$	SXY	A2 по формулам (8-10)	
$\Sigma_i x_i^2$	SX2		
номер формулы	L	Номер эмпирической формулы	integer

Структурограмма

Ввод N, X[I], Y[I], где I = 1 .. N	
	Ввод номера L эмпирической зависимости
	Анализ L

L = 1	L = 2		L = 8
Вывод: Y = A * X + B;	Вывод: $y = a \cdot b^x$	. . .	Вывод: $y = a \cdot b^x$;
Расчет X1, X2;	Расчет X1, X2;	. . .	Расчет X1, X2;
процедура NMKP;	процедура NMKP;	. . .	процедура NMKP;
Расчет a, b;	Расчет a, b;	. . .	Расчет a, b;
Расчет Y2	Расчет Y2	. . .	Расчет Y2
Расчет F; Вывод на монитор a, b, F			
Делать до тех пор, пока L не станет равным 0			

```
PROGRAM PR103;          {Поиск параметров эмпирической формулы }
TYPE XT = ARRAY [1..50] OF REAL;
VAR   X, Y, X1, Y1, Y2: XT; I, N, L: INTEGER;
        A, B, A1, A2, F, SY, SX, SXY, SX2: REAL;
PROCEDURE NMKP;     { Метод наименьших квадратов }
BEGIN
SX := 0; SY := 0; SX2 := 0; SXY := 0;
FOR I := 1 TO N
DO BEGIN
     SXY := SXY + X1[I] * Y1[I];
     SX   := SX + X1[I];
     SY   := SY + Y1[I];
     SX2 := SX2 + X1[I] * X1[I]
     END;
A1 := (N * SXY - SX * SY)/(N * SX2 - SX * SX);
A2 := (SY - A1 * SX)/N
END;  { PROCEDURE }
FUNCTION  POW(C, D: REAL):REAL;    { Показательная функция, }
BEGIN  POW := EXP(D * LN(C)) END;    { D - Степень; C - Основание}
BEGIN
WRITELN ('Введите объем выборки'); READLN (N);
WRITELN ('Введите выборку X, Y');
FOR I := 1 TO N DO READLN (X[I], Y[I]);
REPEAT
WRITELN ('Укажите вид зависимости 1..8, 0 - конец');
READLN (L);
CASE L OF           {Выравнивание зависимостей и нахождение}
1: BEGIN            {Параметров эмпирических формул}
   WRITELN ('Y = AX + B'); X1 := X; Y1 := Y;
```

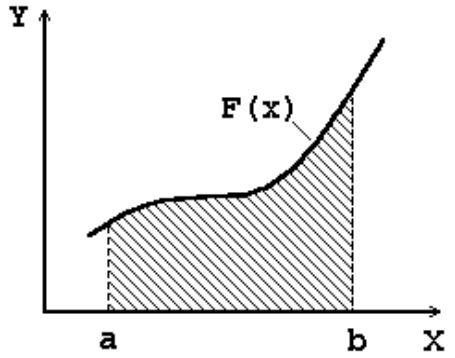
```
   NMKP; A := A1; B := A2;
   FOR I := 1 TO N DO Y2[I] := A * X[I] + B    END;
2: BEGIN     WRITELN ('Y = AB ^ X'); X1 := X;
   FOR I := 1 TO N DO Y1[I] := LN(Y[I]);
   NMKP; A := EXP(A2); B := EXP(A1);
   FOR I := 1 TO N DO Y2[I] := A * POW(B, X[I])    END;
3: BEGIN   WRITELN ('Y=1/(AX + B)'); X1:=X;
   FOR I := 1 TO N DO Y1[I] := 1/Y[I];
   NMKP; A := A1; B := A2;
   FOR I := 1 TO N DO Y2[I] := 1/(A * X[I] + B)     END;
4: BEGIN   WRITELN ('Y = A * LN(X) + B');
   FOR I := 1 TO N DO X1[I]:=LN(X[I]);
   Y1 := Y; NMKP; A := A1; B := A2;
   FOR I := 1 TO N DO Y2[I]:=A * LN(X[I]) + B   END;
5: BEGIN   WRITELN ('Y = A * X ^ B');
   FOR I := 1 TO N DO BEGIN X1[I] := LN(X[I]); Y1[I] := LN(Y[I]) END;
   NMKP; B := A1; A := EXP(A2);
   FOR I := 1 TO N DO Y2[I] := A * POW(X[I], B)   END;
6: BEGIN   WRITELN ('Y = A + B/X');
   FOR I := 1 TO N DO X1[I] := 1/X[I]; Y1 := Y;
   NMKP; A := A2; B := A1;
   FOR I := 1 TO N DO Y2[I] := A + B/X[I]   END;
7: BEGIN   WRITELN ('Y = X/(A * X + B');
   X1 := X; FOR I := 1 TO N DO Y1[I] := X[I]/Y[I];
   NMKP; A := A1; B := A2;
   FOR I := 1 TO N DO Y2[I] := X[I]/(A * X[I] + B)    END;
8: BEGIN    WRITELN ('Y = A * EXP(B * X)');
   X1:=X; FOR I := 1 TO N DO Y1[I] := LN(Y[I]);
   NMKP; A := EXP(A2); B := A1;
   FOR I := 1 TO N DO Y2[I] := A * EXP(B * X[I])   END
   END;  {CASE}
   IF  L IN [1..8]
   THEN BEGIN            {Вычисление суммы квадратов отклонения}
           F := 0.0;
           FOR I := 1 TO N DO F := F + (Y2[I] - Y[I])*(Y2[I] - Y[I]);
           WRITELN ('A = ', A:8:5, ', B = ', B:8:5, ', G(A, B) = ', F:10:5);
           END;
UNTIL L = 0
```

END.

Выбор одной из восьми формул в программе не предусмотрен. Пользователь этой программы сам назначает номера формул, получает на экране значения квадрата отклонения G(A, B) для каждой из формул и выбирает наименьшее, то есть наилучшее приближение (1-8).

2. ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

Определенный интеграл функции F(X) с пределами интегрирования **a** и **b** находиться по формуле Ньютона - Лейбница:



$$J = \int_a^b F(X)dX = \Phi(b) - \Phi(a), \qquad (2\text{-}1)$$

где Ф(Y) - первообразная функция от F(X). Определенный интеграл можно трактовать как площадь фигуры (рис. 30) ограниченной ординатами F(a) и F(b), осью абсцисс X и графиком подынтегральной функции F(X). Если функция Ф(X) известна, то задача становится тривиальной, и сводится к расчету по формуле (9-1). Если Ф(X) неизвестна, то применяются методы численного интегрирования, суть которых состоит в оценке площади с помощью формул численного интегрирования, которые называются квадратурой.

2.1. Программирование интерполяционно-квадратурных формул

В этом учебном пособии ограничимся группой методов, получивших название методы Ньютона - Котеса. В методах этого класса на интервале [a, b] строится равномерная сетка X_0, h, n, где X_0 = a, n - число подинтервалов, на которое разбивается [a, b], h - шаг интегрирования. Величины h и Xi можно найти по формулам:

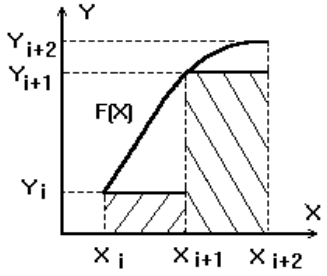
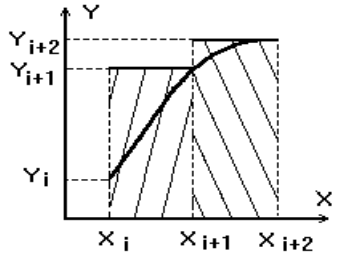
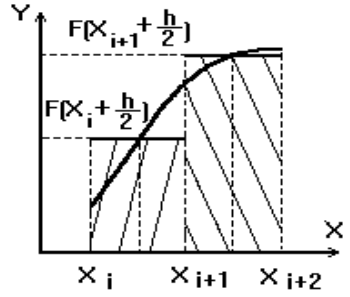
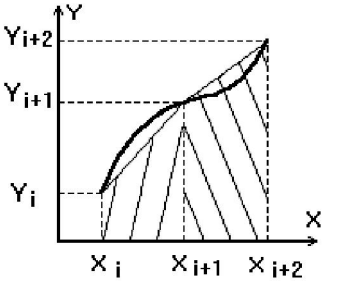
$h = (b - a) / n;\ \ Xi = X_0 + i \cdot h.$

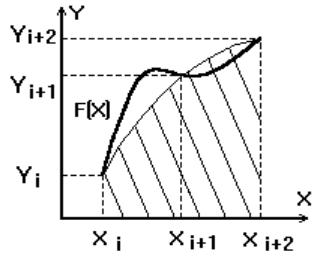
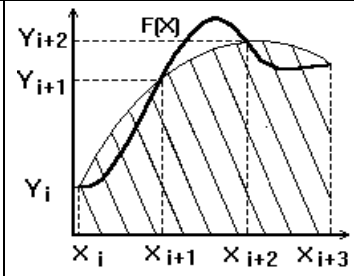
(2-2)

Для точек $\{X_i / i = \overline{0,n}\}$ находятся значения $\{Y_i = F(X_i) / i = \overline{0,n}\}$ и таким образом получается таблица $\{X_i, Y_i / i = \overline{0,n}\}$. Далее строится интерполяционный полином степени S. Площадь полученной геометрической фигуры вычисляется по квадратурным формулам, приведенным в табл. 2. Погрешность интегрирования E зависит от метода и выбранного шага h. В табл. 2 приведена нижняя граница E.

Для программирования квадратурных формул удобно создать пользовательскую библиотеку (модуль - автономно компилируемую программную единицу). В этой библиотеке с именем INTEGRAL интерполяционно-квадратурные формулы оформлены в виде пользовательских функций, на входе которых формальные параметры N - число шагов, нижний A и верхний B пределы интегрирования и имя подынтегральной функции F(X). В свою очередь функция возвращает численное значение интеграла.

Квадратурные формулы **Таблица 2**

Численный метод интегрирования	Графическая иллюстрация
Формула левых прямоугольников, S = 0, E > 0.001. $J = h \cdot \sum_{i=0}^{n-1} Y_i$	
Формула правых прямоугольников, S = 0, E > 0.001 $J = h \cdot \sum_{i=1}^{n} Y_i$	
Формула центральных прямоугольников, S = 0, E > 0.0001 $J = h \cdot \sum_{i=0}^{n-1} F\left(X_i + \frac{h}{2}\right)$	
Формула трапеций, S=1, E > 0.0001. $J = \frac{h}{2} \cdot \sum_{i=0}^{n-1} \left(Y_i + Y_{i+1}\right)$	

Формула Симпсона, S = 2, E > 0.0000001 $J = \frac{h}{3} \cdot \sum_{i=0,2,4}^{n-2} (Y_i + 4Y_{i+1} + Y_{i+2})$	
Формула Ньютона, S = 3, E > 0.0000001 $J = \frac{3h}{8} \sum_{i=0,3,6}^{n-3} (Y_i + 3Y_{i+1} + 3Y_{i+2} + Y_{i+3})$	

Пример 104. Создать пользовательскую библиотеку INTEGRAL, в которой запрограммировать квадратурные формулы из табл. 2.

```
UNIT INTEGRAL;   {Формулы численного интегрирования}
INTERFACE
TYPE FUN = FUNCTION(X: REAL): REAL; MAS = ARRAY[0..20] OF REAL;
FUNCTION LREC(N: INTEGER; A, B: REAL; F: FUN): REAL;
FUNCTION RREC(N: INTEGER; A, B: REAL; F: FUN): REAL;
FUNCTION CREC(N: INTEGER; A, B: REAL; F: FUN): REAL;
FUNCTION TRAP(N: INTEGER; A, B: REAL; F: FUN): REAL;
FUNCTION SIMP( N: INTEGER; A, B: REAL; F: FUN): REAL;
FUNCTION NIUT( N: INTEGER; A, B: REAL; F: FUN): REAL;
FUNCTION TRAP_T(Z: MAS; H: REAL; K: INTEGER): REAL;
FUNCTION SIMP_T(Z: MAS; H: REAL; K: INTEGER): REAL;
IMPLEMENTATION
FUNCTION LREC;        {Формула левых прямоугольников}
VAR I: INTEGER; H, S: REAL;
BEGIN
H := (B-A)/N;  S:=0;
FOR I := 0 TO N-1 DO S := S + F(A+ I * H);
LREC := H * S
END;
FUNCTION RREC;        {Формула правых прямоугольников}
VAR I: INTEGER; H, S: REAL;
BEGIN
H := (B - A)/N; S := 0;
FOR I := 1 TO N DO S := S + F(A + I * H);
```

```
RREC := H * S
END;
FUNCTION CREC;          {Формула центральных прямоугольников}
VAR I: INTEGER; H, S: REAL;
BEGIN
H := (B - A)/N;  S := 0;
FOR I := 0 TO N-1 DO S := S + F(A + (I + 0.5) * H);
CREC :=H * S
END;
FUNCTION TRAP;          {Формула трапеции}
VAR I: INTEGER; H, S: REAL;
BEGIN
H := (B-A)/N;  S:=(F(A) + F(B)) * 0.5;
FOR I := 1 TO N-1 DO S := S + F(A+ I * H);
TRAP := H * S
END;
FUNCTION SIMP; {Формула Симпсона}
VAR I: INTEGER; H, S: REAL;
BEGIN
H := (B-A)/N; S := 0;
FOR I := 0 TO (N-2) DIV 2
DO S := S + F(A +2 * I * H) + 4 * F(A+ (2 * I + 1) * H) + F(A+ (2 * I +2) * H);
SIMP := H * S/3
END;
FUNCTION NIUT; {Формула Ньютона}
VAR I: INTEGER; H, S: REAL;
BEGIN
H := (B-A)/N; S := 0;
FOR I := 0 TO (N-3) DIV 3
DO S := S + F(A+ 3 * I * H)+3 * F(A+ (3 * I +1) * H)+3 * F(A+(3 * I +2) * H)
+F(A+ (3 * I+3) * H);
NIUT := 3 * H * S/ 8
END;
FUNCTION SIMP_T;       {Формула Симпсона для табличной функции}
VAR I: INTEGER; S: REAL;
BEGIN
S := 0;
FOR I := 0 TO (K-2) DIV 2
```

```
DO S := S + Z[2 * I] +4 * Z[2 * I+1] +Z[2 * I+2];
SIMP_T := H * S/3
END;
FUNCTION TRAP_T;     {Формула трапеций для табличной функции}
VAR I: INTEGER; S: REAL;
BEGIN
S := 0;
FOR I := 1 TO (K-1) DO S := S + Z[I] + Z[I+1];
TRAP_T := H * S * 0.5
END
END.
```

В конец этой библиотеки INTEGRAL включены две функции для интегрирования методами Симпсона и Трапеций табличных функций. Подробнее эти функции будут описаны в разделе 2.3.

2.2. Интегрирование аналитических функций

Если подынтегральная функция задана аналитически, то ее значения можно посчитать в любой точке $X \in [a, b]$ и по квадратурным формулам вычислить значение интеграла. Это значение зависит от N (количества отрезков $[X_i, X_{i+1}]$) на которые разбивается [a, b]. Чем таких отрезков больше, тем меньше погрешность метода численного интегрирования. Однако с ростом N растет количество расчетов, и одновременно возрастает погрешность округления. Эта погрешность вызвана преобразованиями вещественных чисел при операциях деления, извлечения корней, вычисления тригонометрических и показательных функций в условиях ограниченной разрядности оперативной памяти и микропроцессора. Таким образом, с ростом N суммарная погрешность E уменьшается, а затем начинает возрастать. Для каждой квадратурной формулы существует оптимальное значение N и соответствующее ему минимальное значение погрешности. Примеры нахождения E_{min} приведены в (23). В табл. 44 приведены ориентировочные значения нижней границы E, которые можно требовать от методов. При численном интегрировании часто используют метод двойного пересчета. Суть его состоит в том, что интеграл считается несколько раз. Каждый раз число N увеличивается в два раза N := 2 * N. Значения интеграла J_i на i-ом шаге сравнивается с предыдущим значением J_{i-1}. Этот процесс повторяется до тех пор, пока не выполнится условие $|J_i - J_{i-1}| < E$.

Пример 105. Методом двойного пересчета, используя формулу Симпсона, вычислить интеграл $J = \int_a^b \mathbf{x} \cdot \sin(\mathbf{x})d\mathbf{x}$ с погрешностью E. В программе следует использовать библиотеку функций INTEGRAL из раздела 2.1.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
a	A	Нижний предел интеграла.	real
b	B	Верхний предел интеграла.	real
n	NTEK	Число интервалов, на кот. разбивается [A, B].	integer
h	H	Шаг интегрирования функции F(X).	integer
F(X)	F(X)	Подынтегральная функция.	real
J_i	J1	Численное значение интеграла на шаге i.	real
J_{i-1}	J2	Численное значение интеграла на шаге i-1.	real
E	E	Погрешность интегрирования.	real

Структурограмма

Ввести с клавиатуры пределы интегрирования A, B и погрешность E; Установить начальное количество интервалов NTEK = 6 и J1 = 0;
Считать вычисленное значение интеграла за старое: J2 = J1; Вычислить по формуле Симпсона новое значение интеграла J1; Увеличить число отрезков на [A, B] в два раза: NTEK = NTEK * 2;
Делать до тех пор, пока не выполнится условие | J1-J2 | < E;
Вывести на экран монитора значение интеграла J1.

```
PROGRAM PR105;   {Интегрирование аналитически заданных функций}
USES INTEGRAL;   {Наименьшее начальное значение N, удовлетворяющее}
CONST N0 = 6;          {любой формуле интегрирования }
VAR A, B, J1, J2, E: REAL; NTEK: INTEGER;
{$F+}  {Компиляция функций с дальним типом обращения}
FUNCTION F(X: REAL): REAL;       { Подынтегральная функция }
BEGIN   F := X * SIN(X)   END;
{$F-}
BEGIN
WRITELN('ВВЕДИТЕ A, B, E'); READ(A, B, E);
J1 := 0; NTEK := N0;
```

```
REPEAT
J2 := J1; J1 := SIMP(NTEK, A, B, F);
NTEK := 2 * NTEK;        {Увеличение числа отрезков в два раза}
UNTIL ABS(J1-J2) <= E;
WRITELN('Интеграл равен = ', J1:10:7)
END.
```

2.3. Интегрирование табличных функций

Когда известны только табличные значения функции $\{X_i, Y_i / i = \overline{0,n}\}$, то можно лишь оценить численное значение интеграла. Погрешность вычислений, от программиста зависит только в плане выбора метода. Рекомендуется использовать два метода: формулу трапеций, если n - нечетно, и формулу Симпсона для четного n. При равномерной сетке значения X_i становятся не существенными, поскольку в формулах табл. 44 используются только $\{Y_i / i = \overline{0,n}\}$ и h.

Пример 106. Для табличной функции $\{Y_i / i = \overline{0,n}\}$ определенной на равномерной сетке с шагом h вычислить определенный интеграл. В программе использовать библиотеку INTEGRAL.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число отрезков на интервале $[X_0, X_n]$	integer
h	H	Шаг табулирования функции Y(X).	real
Y_i	Y[I]	Значение функции в точке X_i.	real
J	Q	Численное значение интеграла.	real

Структурограмма

Ввод H, N и значений функции Y[i], i = 0..n;	
Анализ N четное	нечетное
Вычисление интеграла формулой Симпсона	Вычисление интеграла формулой трапеций
Вывод найденного значения Q на монитор.	

В отличие от примера 105 в функции численного дифференцирования передаются: имя таблицы Y (массива значений функции), размер таблицы n и шаг h.

```
PROGRAM PR106; {Интегрирование табличной функции}
USES INTEGRAL;
VAR Y: MAS; I, N: INTEGER; H, Q: REAL;
BEGIN
WRITELN('Введите H, N'); READ(H, N);
```

```
WRITELN('Введите массив Y');
FOR I := 0 TO N DO READ(Y[I]);
IF ODD(N)            {Анализ размера таблицы}
THEN Q : = TRAP_T(Y, H, N);   {N - нечетное, формула трапеций для таблицы}
ELSE Q := SIMP_T(Y, H, N)     { N - четное, формула Симпсона для таблицы}
WRITELN('Интеграл равен = ', Q:10:7)
END.
```

2.4 Метод Монте-Карло

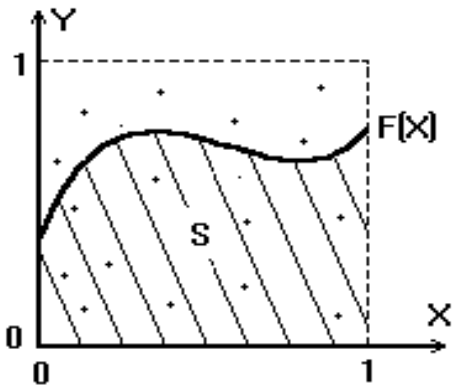


Рис. 1.

Метод Монте-Карло заключается в использовании случайных чисел для моделирования различных объектов, ситуаций, физических явлений и др. Последовательность случайных чисел характеризуется законом распределения. На ЭВМ обычно генерируется равномерно распределенные случайные числа, имеющие равную вероятность появления в диапазоне [A, B]. Случайные числа с другими законами распределения получают с помощью формул преобразования. Наглядным примером использования метода Монте-Карло является численное решение определенных интегралов. Суть метода состоит в следующем. Пусть у нас есть график функции $0 \le F(X) \le 1$ (см. рис.1), заданной на интервале $X \in [0, 1]$. Требуется найти площадь S геометрической фигуры, ограниченной F(X) и осью абсцисс. Для этого нужно равномерно случайным образом разбросать N точек внутри квадрата. Затем посчитать количество точек V, попавших в оцениваемую площадь S. На основании метода Монте-Карло искомая площадь $J \approx V/N$, а погрешность вычислений $E \approx 1/\sqrt{N}$.

Искомую площадь можно трактовать как определенный интеграл:

$$J = \int_0^1 F(X)\,dX \approx \frac{V}{N} \qquad (2\text{-}3)$$

Обобщим задачу. Пусть дана непрерывная функция $Y2 \le F(X) \le Y1$.

Требуется найти: $J = \int_A^B F(X)\,dX$.

На рис.2. показано условное положение функции F(X). Искомый интеграл J равен J = S1 - S2, где S1 суммарная площадь, ограниченная F(X) > 0 и осью X, а S2 суммарная площадь ограниченная F(X) < 0 и осью X. Пусть:

$$Y1 = \begin{cases} Y1, Y1 > 0 \\ 0, \quad Y1 \le 0; \end{cases} \quad Y2 = \begin{cases} Y2, Y2 < 0 \\ 0, \quad Y2 \ge 0. \end{cases} \qquad (2\text{-}4)$$

Тогда нетрудно показать, что численное значение интеграла будет равно:

$J \approx H \cdot (V1 \cdot Y1 + V2 \cdot Y2)/N;\ H = B - A,$

(2-5)

где N - объем выборки, V1 - число точек попавших в область S1, а V2 - число точек попавших в S2.

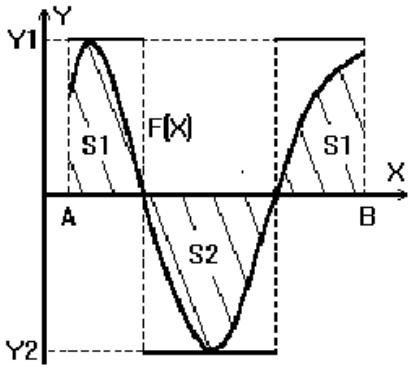


Рис. 2.

С позиций статистического моделирования понятие "бросить точку" на плоскость означает взять два случайных числа $A \le D1 \le B$ и $0 \le D2 \le Y1$, если F(D1)>0, или Y2 < D2 < 0, если F(D1) < 0. Число D1 соответствует значению координаты точки по оси X, а значение D2 соответствует координате точки по оси Y. Принадлежность точки (D1, D2) площади S1 определяется из условия $F(D1) \ge D2 \ge 0$ и площади S2 из условия $F(D1) \le D2 \le 0$.

Пример 107. Вычислить интеграл $J\ p(Z) = \frac{1}{\pi}\int_0^{\pi} COS(Z \cdot SIN(X) - P \cdot X)dX$, определяющий функции Бесселя первого рода порядка P от аргумента Z методом Монте-Карло .

Подынтегральная функция F(X) = COS(Z · SIN(X) - P · X) dX. Очевидно, что $-1 \le F(X) \le 1$, следовательно, из (2-3) можно найти Y1 = 1, Y2 = -1 и выражение (2-4) примет вид:

$J \approx H \cdot (V1 - V2)/N.$ (2-6)

Учитывая из условий задачи, что $B = \pi$, а A = 0 получаем $H = B - A = \pi$. Таким образом, искомый интеграл равен:

$J \approx (V1-V2)/N = V/N,\ V = V1-V2.$ (2-7)

Следует обратить особое внимание на тип переменной N. Несмотря на то, что N по сути целочисленная величина, в программе приходится работать с типом REAL, что вызвано большим объемом выборки и ограничениями целых типов Integer и Word. Применение переменной N: Real, заставляет в программе отказаться от использования арифметического цикла FOR, и применять оператор REPEAT. Величина Y введена для снижения времени работы программы. Аналогично, введена константа B, равная числу π. Эта константа вычисляется один раз перед выполнением программы.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
P	P	Порядок функции Бесселя.	real
Z	Z	Аргумент функции Бесселя.	real
N	N	Объем выборки (число испытаний).	real
V1-V2	V	Число точек попавших в оцениваемую площадь.	real
F(X)	F(X)	Значение подынтегральной функции.	real
B	B	Верхний предел интеграла.	real
D1	X	Случайная величина на оси X в диапазоне 0 .. π	real
D2	SY	Случайная величина на оси Y в диапазоне 0 .. 1.	real
F(X)	Y	Значение подынтегральной функции.	real
R	R	Счетчик числа испытаний.	real

Структурограмма

<table>
<tr><td colspan="3">Инициализировать датчик случайных чисел;
Ввести параметры функции Бесселя P, Z и объем выборки N;
Установить в 0 счетчики V и R.</td></tr>
<tr><td></td><td colspan="2">Увеличить счетчик числа испытаний на 1;
Взять на оси абсцисс в диапазоне 0 .. π новое случайное число X;
Вычислить Y значение функции Бесселя F(X);
Взять на оси ординат в диапазоне 0 .. 1 случайное число SY.</td></tr>
<tr><td></td><td colspan="2">0 < SY <= |Y|
True</td></tr>
<tr><td></td><td colspan="2">Y : 0
больше меньше</td></tr>
<tr><td></td><td>V = V + 1</td><td>V = V - 1</td></tr>
<tr><td colspan="3">Повторять до тех пор, пока не проведем N испытаний.</td></tr>
</table>

Вывести значение интеграла и погрешность расчетов на монитор.

```
PROGRAM PR107;          { Вычисление интеграла Бесселя }
CONST B = PI;           { Методом Монте-Карло }
VAR  V, X, P, Z, R, N, Y, SY: REAL;
FUNCTION F(W: REAL): REAL;      {Функция Бесселя}
BEGIN F := COS(P * W - Z * SIN(W)) END;
BEGIN
RANDOMIZE;      {Инициализация датчика случайных чисел}
WRITELN('Укажите параметры функции Бесселя P и Z.'); READLN(P, Z);
WRITELN('Задайте объем выборки N'); READLN(N);
R := 0; V := 0;
REPEAT
R := R + 1; X := B * RANDOM;        {Случайное число в диапазоне  0 .. pi}
Y := F(X); SY := RANDOM;
IF (0 < SY) AND (SY <= ABS(Y))
THEN IF Y>0
        THEN V := V + 1
        ELSE V := V - 1
UNTIL R >= N;
WRITELN('Интеграл Бесселя J = ', V/N:8:6);
WRITELN('Погрешность вычислений: ', 1/SQRT(N):6:4)
END.
```

Замечание. Время выполнения программы большое. Погрешность вычисления примерно соответствует методам левых или правых прямоугольников. Поэтому программа носит чисто демонстрационный характер.

3. МЕТОДЫ ЧИСЛЕННОГО ДИФФЕРЕНЦИРОВАНИЯ

Численное дифференцирование аналитически или таблично заданной функции F(x) заключается в замене F(x) интерполяционным полиномом P(x), производные $P^{(n)}(x) = \frac{d^n P(x)}{dx^n}$ которого можно найти аналитически. Предполагается, что $F^{(n)}(x) \cong P^{(n)}(x)$. В качестве P(x) используют полиномы Ньютона, Гаусса, Стирлинга.

Таблица 3

п/п	Начало	Середина	Конец
X_{i-2}	X_0	X_{-2}	X_{-4}
X_{i-1}	X_1	X_{-1}	X_{-3}
X_i	X_2	X_0	X_{-2}
X_{i+1}	X_3	X_1	X_{-1}
X_{i+2}	X_4	X_2	X_0

Формулы численного дифференцирования найдены для равномерной сетки. В предыдущих разделах для вычисления значения X_i использовалась формула $X_i = X_0 + I \cdot h$, где X_0 крайняя слева, начальная точка равномерной сетки. Отсчет ведется от этой точки вправо I = 0 .. n. Такая сетка называется *начало таблицы*. Если в качестве начальной задана крайняя правая точка *конец таблицы*, то отсчет должен идти влево от этой точки i = -n, ..., -2, -1, 0. Возможно, такое расположение точки X_0, что необходимо учесть узлы лежащие слева и справа от X_0 *середина таблицы*. В этом случае I = ... -3, -2, -1, 0, 1, 2, 3, ... В табл. 3 указано обозначение пяти узлов на оси X используемые в формулах численного дифференцирования. Введем следующие обозначения: значения функций

$Y_{-k} = F(X_{i-k})$, $Y_0 = F(X_0)$, $Y_k = F(X_{i+k})$; значения первой производной $Y'_{-k} \cong F'(X_{i-k})$, $Y'_0 \cong F'(X_0)$, $Y'_k \cong F'(X_{i+k})$; значения n-ой производной $Y^n_{-k} \cong F^{(n)}(X_{i-k})$, $Y^n_0 \cong F^{(n)}(X_0)$, $Y^n_k \cong F^{(n)}(X_{i+k})$.

3.1. Программирование формул численного дифференцирования

В табл. 4 включены получившие наибольшее применение формулы численного дифференцирования аналитически заданных функций. Для расчета производных первой, второй или третьей по этим формулам нужно, во-первых, выбрать

начало, конец или середину таблицы. Во-вторых, учитывая погрешность метода, определить число узлов. В-третьих, подобрать оптимальное значение шага h. Результат вычислений сильно зависит от выбора h.

Формулы численного дифференцирования **Таблица 4**

Число узлов	Положение в таблице	Производные	Погрешность метода
3	Начало	$Y'_0 = (-3 \cdot Y_0 + 4 \cdot Y_1 - Y_2) / 2 \cdot h$ $Y''_0 = (Y_0 - 2 \cdot Y_1 + Y_2) / h^2$	$h^2 \cdot F'''(\xi) / 3$ $-h \cdot F'''(\xi)$
	Середина	$Y'_0 = (Y_1 - Y_{-1}) / 2 \cdot h$ $Y''_0 = (Y_1 - 2 \cdot Y_0 + Y_{-1}) / h^2$	$-h^2 \cdot F'''(\xi) / 6$ $-h^2 \cdot F^4(\xi) / 12$
	Конец	$Y'_0 = (Y_{-2} - 4 \cdot Y_{-1} + 3 \cdot Y_0) / 2 \cdot h$ $Y''_0 = (Y_{-2} - 2 \cdot Y_{-1} + Y_0) / h^2$	$h^2 \cdot F'''(\xi) / 3$ $h \cdot F'''(\xi)$
4	Начало	$Y'_0 = (-11Y_0 + 18Y_1 - 9Y_2 + 2Y_3) / 6h$ $Y''_0 = (2Y_0 - 5Y_1 + 4Y_2 - Y_3) / h^2$	$-h^3 \cdot F^4(\xi) / 4$ $11h^2 \cdot F^4(\xi)/12$
	Конец	$Y'_0 = (-2Y_{-3} + 9Y_{-2} - 18Y_{-1} + 11Y_0)/6h$ $Y''_0 = (-Y_{-3} + 4Y_{-2} - 5Y_{-1} + 2Y_0) / h^2$	$-h^3 \cdot F^4(\xi) / 4$ $11h^2 \cdot F^4(\xi)/12$
5	Середина	$Y'_0 = (-Y_2 + 8Y_1 - 8Y_{-1} + Y_{-2}) / 12h$ $Y''_0 = (-Y_2 + 16Y_1 - 30Y_0 + 16Y_{-1} - Y_{-2}) / 12h^2$ $Y'''_0 = (Y_2 - 2Y_1 + 2Y_{-1} - Y_{-2}) / 2h^3$	$h^4 \cdot F^5(\xi) / 30$ $-h^4 \cdot F^5(\xi)/90$ $-h^2 \cdot F^5(\xi)/4$
7	Середина	$Y'_0 = (Y_3 - 9Y_2 + 45Y_1 - 45Y_{-1} + 9Y_{-2} - Y_{-3}) / 60h$ $Y''_0 = (2Y_3 - 27Y_2 + 270Y_1 - 490Y_0 + 270Y_{-1} - 27Y_{-2} + 2Y_{-3}) / 180h^2$ $Y'''_0 = (-Y_3 + 8Y_2 - 13Y_1 + 13Y_{-1} - 8Y_{-2} + Y_{-3}) / 8h^3$	$-h^6 \cdot F^7(\xi)/140$ $-h^6 \cdot F^8(\xi)/560$ $7h^4 \cdot F^7(\xi)/120$

Пример 108. Построить пользовательскую библиотеку функций DIF для вычисления первой, второй и третьей производных по формулам из табл. 4.

```
UNIT DIF;    {Формулы численного дифференцирования}
INTERFACE
CONST H = 0.001; { Шаг равномерной сетки }
TYPE FUN = FUNCTION(X: REAL): REAL; MAS = ARRAY[0..20] OF REAL;
FUNCTION DIF1_NT3(X: REAL; F: FUN): REAL;
FUNCTION DIF1_ST3(X: REAL; F: FUN): REAL;
FUNCTION DIF1_KT3(X: REAL; F: FUN): REAL;
FUNCTION DIF2_NT3(X: REAL; F: FUN): REAL;
FUNCTION DIF2_ST3(X: REAL; F: FUN): REAL;
```

```
    FUNCTION DIF2_KT3(X: REAL; F: FUN): REAL;
    FUNCTION DIF1_NT4(X: REAL; F: FUN): REAL;
    FUNCTION DIF1_KT4(X: REAL; F: FUN): REAL;
    FUNCTION DIF2_NT4(X: REAL; F: FUN): REAL;
    FUNCTION DIF2_KT4(X: REAL; F: FUN): REAL;
    FUNCTION DIF1_ST5(X: REAL; F: FUN): REAL;
    FUNCTION DIF2_ST5(X: REAL; F: FUN): REAL;
    FUNCTION DIF3_ST5(X: REAL; F: FUN): REAL;
    IMPLEMENTATION
    FUNCTION DIF1_NT3;          {Первая производная, начало таблицы, 3 узла}
    BEGIN  DIF1_NT3 := (-3*F(X) + 4*F(X + H) - F(X + 2*H))/2/H END;
    FUNCTION DIF2_NT3;          {Вторая производная, начало таблицы, 3 узла}
    BEGIN DIF2_NT3 := (F(X) - 2*F(X + H) + F(X + 2*H))/H/H END;
    FUNCTION DIF1_ST3;          {Первая производная, середина таблицы, 3 узла}
    BEGIN DIF1_ST3 := (F(X + H) - F(X - H))/2/H END;
    FUNCTION DIF2_ST3;          {Вторая производная, середина таблицы, 3 узла}
    BEGIN DIF2_ST3 := (F(X - H) - 2*F(X) + F(X + H))/H/H END;
    FUNCTION DIF1_KT3;          {Первая производная, конец таблицы, 3 узла}
    BEGIN DIF1_KT3 := (F(X - 2*H) - 4*F(X - H) + 3*F(X))/2/H END;
    FUNCTION DIF2_KT3;          {Вторая производная, конец таблицы, 3 узла}
    BEGIN DIF2_KT3 := (F(X - 2*H) - 2*F(X - H) + F(X))/H/H END;
    FUNCTION DIF1_NT4;          {Первая производная, начало таблицы, 4 узла}
    BEGIN DIF1_NT4 := (-11*F(X) + 18*F(X + H) - 9*F(X + 2*H) + 2*F(X
          + 3*H))/6/H END;
    FUNCTION DIF2_NT4;          {Вторая производная, начало таблицы, 4 узла}
    BEGIN DIF2_NT4 := (2*F(X) - 5*F(X + H) + 4*F(X + 2*H) - F(X + 3*H))/H/H
END;
    FUNCTION DIF1_KT4;          {Первая производная, конец таблицы, 4 узла}
    BEGIN DIF1_KT4 := (-2*F(X - 3*H) + 9*F(X - 2*H) - 18*F(X - H)
          + 11*F(X))/6/H END;
    FUNCTION DIF2_KT4;          {Вторая производная, конец таблицы, 4 узла}
    BEGIN DIF2_KT4 := (-F(X - 3*H) + 4*F(X - 2*H) - 5*F(X - H) + 2*F(X))/H/H
END;
    FUNCTION DIF1_ST5;          {Первая производная, середина таблицы, 5 узлов}
    BEGIN DIF1_ST5 := (-F(X + 2*H) + 8*F(X + H) - 8*F(X - H)
```

```
      + F(X - 2*H))/12/H END;
    FUNCTION DIF2_ST5;          {Вторая производная, середина таблицы, 5 узлов}
    BEGIN DIF2_ST5 := (-F(X + 2*H) + 16*F(X + H) - 30*F(X) + 16*F(X - H)
      - F(X - 2*H))/12/H/H END;
    FUNCTION DIF3_ST5;          {Третья производная, середина таблицы, 5 узлов}
    BEGIN DIF3_ST5 := (F(X + 2*H) - 2*F(X + H) + 2*F(X - H)
      - F(X - 2*H))/2/H/H/H END
    END.
```

В модуле DIF для всех функций аргументами являются значение X, точки на оси абсцисс, в которой нужно найти производную, и имя дифференцируемой функции F(X).

3.2. Табулирование аналитически заданной функции и ее первых трех производных

Пример 109. Табулировать значения $F(x) = x^3 - 0.5 \cdot x^2 + 2 \cdot x - 1$ и первых трех производных на равномерной сетке x_0, h, n. Известны x_0, x_n и число подинтервалов n. При составлении программы использовать библиотеку DIF из раздела 3.1.

```
    PROGRAM PR109; {Табуляция функции F(x) и ее трех производных}
    USES DIF;
    VAR I, N: INTEGER; T, T0, TN, H: REAL;
    {$F+}
    FUNCTION Z(T: REAL): REAL;
    BEGIN Z := T * T * T - 0.5 * T * T + 2*T - 6   END;
    {$F-}
    BEGIN
    WRITELN('Введите начальную точку X0, конечную XN, число точек N.');
    READLN(T0, TN, N); N := N - 1; H := (TN - T0)/N;
    WRITELN('┌──┬──────────┬──────────────────────────┐');
    WRITELN('│  │ ЗНАЧЕНИЕ │      ПРОИЗВОДНЫЕ:        │');
    WRITELN('│ X │ ФУНКЦИИ ├──────────┬──────────┬──────────┤');
    WRITELN('│  │   F(X)   │ ПЕРВАЯ  │ ВТОРАЯ  │ ТРЕТЬЯ  │');
    WRITELN('├──┼──────────┼──────────┼──────────┼──────────┤');
    FOR I := 0 TO N
    DO BEGIN  T := T0 + I * H;
       WRITELN('│ ', T:4:1,' │', Z(T):8:4 ,'│', DIF1_ST5(T, Z):8:4, '│',
```

```
        DIF2_ST5(T, Z):8:4, '│', DIF3_ST5(T, Z):8:4, '│')
    END;
WRITELN('└──────┴──────────┴──────────┴──────────┴──────────┘');
END.
```

В тексте программы использованы символы псевдографики. Напомним, чтобы набрать символ псевдографики в редакторе Турбо-Паскаль, например символ '┬', необходимо левой рукой нажать на клавишу Alt и держать ее, а правой рукой на дополнительной цифровой клавиатуре набрать цифры 194. После чего следует отпустить клавишу Alt . Если все выполнено правильно, то в том месте, где располагался курсор, появится символ '┬'.

3.3. Интерполирование производных таблично заданных функций

Если на равномерной сетке [a, b] задана табличная функция, то можно для x ∈ [a, b] оценить значение первой производной с помощью формул из табл. 5. Эти формулы найдены для таблицы содержащей 3, 4 и 5 узлов путем дифференцирования интерполяционного полинома.

Формулы численного дифференцирования **Таблица 5**

Число узлов	Интерполяционный полином первой производной
3	$Y'(x_0+qh)=((q-0.5)Y_{-1}+2qY_0+(q+0.5)Y_1)/h$
4	$Y'(x_0+qh)=-(3q^2-6q+2)Y_{-1}/6h+(3q^2-4q-1)Y_0/2h+$ $+(3q^2-2q-2)Y_1/2h+(2q^2-1)Y_2/6h$
5	$Y'(x_0+qh)=(2q^3-3q^2-q+1)Y_{-2}/12h+(4q^3-3q^2-8q+4)\times$ $Y_{-1}/6h+(2q^3-5q)Y_0/2h-(4q^3+3q^2-8q+4)Y_1/6h+$ $(2q^3+3q^2-q-1)Y_2/12h$

В этих формулах $h=\frac{(b-a)}{n}$, $q=\frac{(x-a)}{h}$, $x=a+q\cdot h$.

Пример 110. Построить алгоритм и программу вычисления первой производной для табличной функции $\{x_i, y_i; x_i = x+i\cdot h; i=\overline{0,n}\}$. В расчетах следует использовать интерполяционный полином для трех узлов из табл. 5.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
X_0	X0	Начало интервала [a, b].	real
h	H	Шаг интерполирования (для равномерной сетки).	real
i	I	Номер интервала [X_{i-1}, X_{i+1}].	integer

n	N	Число узлов в таблице.	integer
F_i	Y[I]	Заданные значения функции F(Xi), I = 0 .. N-1.	real
X	X	Значение аргумента внутри интервала [a, b]	real
Xc	XC	Центральный узел интервала $[X_{i-1}, X_{i+1}]$.	real
F'(X)	YP	Искомое значение производной функции F(X).	real
q	Q	Безразмерная переменная в формуле Ньютона.	real

Алгоритм начнем с ввода параметров равномерной сетки X_0, h, n и значений функции $\{Y_i / i = \overline{0,n}\}$. Следующий шаг состоит в задании X, для которого следует вычислить F'(X). Если условие $X \in [X_0, Xn]$ выполнено, то можно вычислять производную по формулам из табл. 5. Легко заметить, что все три формулы заданы для середины таблицы, поэтому требуется вычислить положение центрального узла Xc. В формулах он обозначен за X_0. Номер центрального узла можно найти из формулы $I =](X - X_0)/h[$. Если окажется, что I = 0, то за центральный узел нужно брать I = 1. Если I = n, то за центральный следует взять узел I = n -1. (Это справедливо только для трех узлов). Найдя значение I можно рассчитать $Xc = X_0 + I \cdot h$ и найти величину q = (X - Xc) / h. Теперь, зная q и I, можно легко воспользоваться формулой для расчета производной.

Структурограмма

<table>
<tr><td colspan="2">Ввод X0, H, N; Присвоить N значение N-1;
Ввести Y[i], где i = 0 .. N;
Ввод значения X.</td></tr>
<tr><td colspan="2">Анализ: $X \in [X_0, Xn]$
False — True</td></tr>
<tr><td>Вывод сообщения:
'X - вне интервала интерполяции'</td><td>Нахождение номера I интервала $[X_{i-1}, X_{i+1}]$;
Вычисление XC и Q;
Расчет значения производной YP в точке X;
Вывод значения функции F'(X) на монитор.</td></tr>
</table>

```
PROGRAM PR110;        {Нахождение производной табличной функции}
CONST M = 10;         {Максимальный размер таблицы}
VAR Y: ARRAY[1..M] OF REAL; I, N: INTEGER; X, Q, H, X0, XC, YP: REAL;
BEGIN
WRITELN('Введите размер таблицы, N'); READLN(N); N := N-1;
WRITELN('Введите значения X0 и H'); READLN(X0,H);
```

```
WRITELN('Введите значения функции, F(XI)');
FOR I := 0 TO N DO READ(Y[I]);
WRITELN('Укажите X, для которого требуется найти производную');
READLN(X);
IF (X0 <= X) AND (X<=X0 + N * H)
THEN BEGIN
 I := TRUNC((X - X0)/H);          {Нахождение номера центрального узла}
 IF I = 0 THEN I := 1;
 IF I = N THEN I := N-1;
 XC := X0 + I * H;                {Значение центрального узла}
 Q := (X - XC)/H; YP := 0;
 YP := ((Q - 0.5)*Y[I-1] - 2*Q*Y[I] + (Q + 0.5)*Y[I + 1])/H;
 WRITELN('Значение производной F(X) = ', YP:7:4)
 END
ELSE WRITELN('X - Вне интервала интерполяции!');
END.
```

4. ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

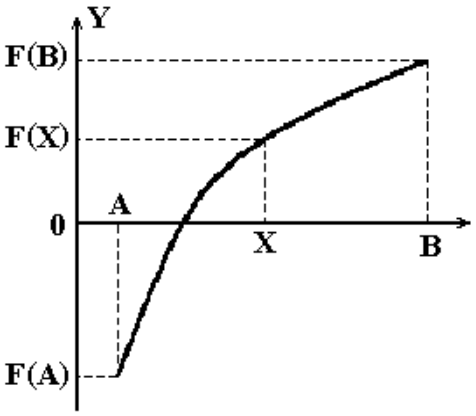
В общем плане задача формулируется следующим образом: необходимо найти значения X_0 X_1, ..., X_k, при которых $F(X_i) = 0$.

В данном пособии рассматриваются методы отыскания действительных (вещественных) корней уравнения. Поставленная задача, как правило, решается в два этапа. Первый из них связан с отделением корней. Второй - заключается в уточнении корня в интервале изоляции с заданной абсолютной погрешностью E. Для удобства изложения материала мы сначала рассмотрим методы уточнения, а затем способы отделения корней.

4.1. Уточнение корня

Интервалом изоляции корня называется промежуток [A, B], содержащий единственный действительный корень уравнения F(X). Задача уточнения корня формулируется следующим образом: для уравнения F(X) = 0, в интервале изоляции [A, B] следует уточнить корень $X^* \in$ [A, B] с погрешностью, не превышающей E. Алгоритмически эта задача состоит из четырех фрагментов. Во-первых, нужно найти X_0 первое приближение к корню. Во-вторых, в соответствии с выбранным методом уточнения корня и первым приближением, построить такую последовательность $\{X_0, X_1, X_2,.., X_{i-1}, X_i,...\}$, чтобы предел X_i был равен корню X^*, при i стремящемся к бесконечности. В-третьих, сформулировать условие окончания вычисления X_i, учитывая погрешность E, и закончить процесс приближения, когда это условие станет истинным. В-четвертых, используя последнее приближение вычислить искомое значение корня.

Метод дихотомии



Пусть в уравнении F(X) = 0 на интервале изоляции [A, B] требуется уточнить корень X^* с погрешностью E методом дихотомии (половинного деления).

Первоначальное приближение: X = (B + A)/2. Правило построение последующих приближений: уменьшить интервал изоляции в два раза путем переноса левой границы из A в X, если корень в правой половине $F(X) \cdot F(B) < 0$, и переноса правой границы из B в X, если корень в левой половине $F(X) \cdot F(A) < 0$. Вычислить новое приближение путем деления найденного интервала пополам. Условие окончания: корень находится внутри интервала, который уменьшается в два раза с каждым последующим приближением. Когда выполнится условие $|B - A| / 2 < E$ остановим процесс приближения к корню. За искомое значение корня примем середину последнего интервала.

Таблица имен

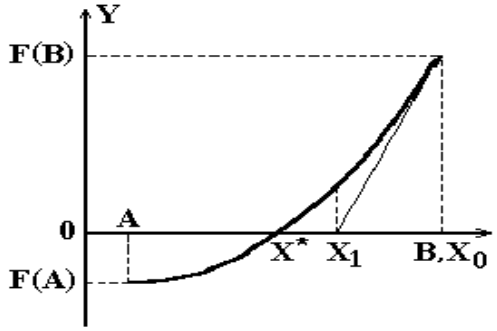
Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
X_i	X	Текущее приближение корня.	real
A	A	Левая граница интервала изоляции корня.	real
B	B	Правая граница интервала изоляции корня.	real
E	E	Абсолютная погрешность вычисления корня.	real
F(X)	Z(X)	Значение исследуемой функции в точке X.	real
X	DIX	Искомое значение корня.	real

Структурограмма

<table>
<tr><td colspan="3">Поделить отрезок [A, B] пополам: X = (A + B)/2; {первое приближение}</td></tr>
<tr><td colspan="3">Выяснить в какой половине корень?</td></tr>
<tr><td>True
False</td><td>$Z(X) \cdot Z(A) < 0$</td><td></td></tr>
<tr><td>(в левой)</td><td>(в правой)</td><td></td></tr>
<tr><td colspan="2">Сделать X правой границей B = X</td><td>Сделать X левой границей A = X</td></tr>
<tr><td colspan="3">Делать до тех пор, пока не выполнится $|B - A| / 2 < E$</td></tr>
<tr><td colspan="3">Вывод найденного значения корня (A + B) / 2</td></tr>
</table>

Использование этого метода ограничивается случаем, когда функция пересекает ось X на [A, B]. Если корнем является точка касания оси X функцией F(X), например $F(X) = X^2$, то предлагаемый алгоритм не найдет очевидного корня X = 0.

Метод касательных



В основе этого метода лежит теорема, сформулированная Ньютоном в 1685 году. Согласно этой теореме, если $F(A) \cdot F(B) < 0$, причем F'(X) и F"(X) отличны от нуля, и сохраняют знаки на [A, B], то, исходя из начального приближения $X_0 \in$ [A, B], удовлетворяющего неравенству $F(X_0) \cdot F''(X_0) > 0$, единственный корень X^* с любой степенью точности E можно вычислить методом касательных.

Воспользуемся алгоритмом метода в формульно-словесном виде. Как следует из теоремы Ньютона, за начальное приближение X_0 берут A, если $F(A) \cdot F''(A) > 0$ или B, если $F(B) \cdot F''(B) > 0$. В точке начального приближения строиться касательная к функции F(X) которая пересекается с осью X в точке следующего приближения X_1 и т. д. Этот итерационный процесс описывается формулой, найденной из уравнения касательной:

$$X_{i+1} = X_i - F(X_i) / F'(X_i). \qquad (4\text{-}1)$$

Последовательность $\{X_i\}$ быстро приближается к корню (рис. 34). Условием окончания приближения может быть событие $|F(X_{i+1})| < E$, или $|X_{i+1} - X_i| < E$, или $F(X_{i+1} - E) \cdot F(X_{i+1} + E) < 0$. За значение искомого корня X^* принимают последнее приближение X_{i+1}

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
X_i	X0	Приближение корня, рассчитанное на шаге i.	real
X_{i+1}	X1	Новое приближение корня на шаге i+1.	real
A	A	Левая граница интервала изоляции корня.	real
B	B	Правая граница интервала изоляции корня.	real
E	E	Погрешность вычисления корня.	real
F(X)	Z(X)	Значение исследуемой функции в точке X.	real
F'(X)	Z1(X)	Значение первой производной F(X)	real
F"(X)	Z2(X)	Значение второй производной F(X)	real
X^*	KAS	Искомое значение корня.	real

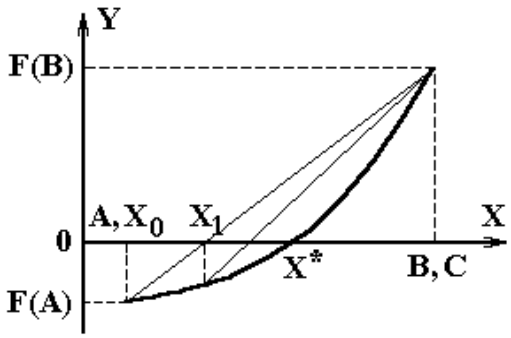
Структурограмма

Z(A) · Z2(A) > 0	
True	False
Первое приближение корня: X1 = A;	Первое приближение корня: X1 = B;
Считать найденное приближения корня старым: X0 = X1; Найти новое приближение корня: X1 = X0 - Z(X0) / Z1(X0);	
Делать до тех пор, пока не выполнится |X1-X0| < E;	
Вывод найденного значения корня X1.	

Недостатком этого метода, по сравнению с методом дихотомии, является необходимость в аналитическом виде задавать первую F'(X) и вторую F''(X) производные. Если F''(X) на интервале изоляции [A, B] меняет знак на противоположный, например, для функции F(X) = Sin(X), то алгоритм зацикливается, или очередное приближение начинает удаляться от корня.

Достоинствами этого метода является самая быстрая сходимость и возможность уточнения корня при касании гладкой функции оси X, например $F(X) = X^2$.

Метод хорд



Этот метод является улучшением метода дихотомии. Вместо деления отрезка пополам предлагается его делить в пропорции F(A) : F(B).

Алгоритм метода в формульно-словесном виде похож на метод Ньютона. Новым приближением в этом методе считается пересечение хорды с осью X. Алгоритм начинается с нахождения первого приближения. Из двух точек A и B выбирается та, в которой произведение функции на ее вторую производную больше 0. Эта точка обозначается за C. Противоположная точка принимается за первое приближение.

Все хорды, меняя угол наклона к оси X, будут проходить через точку (C, F(C)). Приближения считаются на основании рекуррентной формулы:

$X_{i+1} = X_i - F(X_i) \cdot (A - X_i) / (F(A) - F(X_i)); i = 0, 1, 2, ...$

(4-2)

Вычисления продолжаются до тех пор, пока не выполнится условие $|X_{i+1} - X_i| < E$. За точное значение корня принимается последнее приближение.

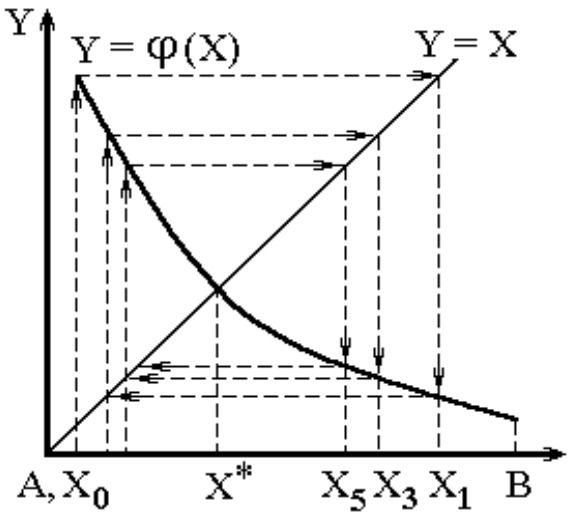
Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
X_i	X0	Приближение корня, рассчитанное на шаге i.	real
X_{i+1}	X1	Новое приближение корня на шаге i + 1.	real
A	A	Левая граница интервала изоляции корня.	real
B	B	Правая граница интервала изоляции корня.	real
E	E	Погрешность вычисления корня.	real
F(X)	Z(X)	Значение исследуемой функции в точке X.	real
F"(X)	Z2(X)	Значение второй производной F(X).	real
X^*	KAS	Искомое значение корня.	real
C	C	Точка вращения хорды.	real

Структурограмма

$Z(A) \cdot Z2(A) > 0$	
True	False
Первое приближение корня: X1 = B; C = A;	Первое приближение корня: X1 = A; C = B;
Считать ранее найденное приближения корня старым: X0 = X1; Найти новое приближение корня: X1 = X0 - Z(X0)*(C -X0)/(Z(C) -Z(X0));	
Делать до тех пор, пока не выполнится |X1-X0| < E;	
Вывод найденного значения корня X1.	

Метод итераций



Пусть [A, B] интервал изоляции корня F(X) = 0. Решение этого уравнения методом итераций предполагает его преобразование к виду X = φ (X), где φ (X) = X + L · F(X).

Обязательное условие для сходимости метода итераций $|\varphi'(X)| < 1$. Выполнение этого условия достигается выбором коэффициента $L = 1/|F'(A)|$, если $|F'(A)| \geq |F'(B)|$ и

$L = 1/|F'(B)|$, если $|F'(B)| > |F'(A)|$. Графическая интерпретация метода итераций для условия $-1 < \varphi'(X) < 0$ представлена на рис. 36. За начальное приближение можно взять любую точку $X_0 \in [A, B]$.

Мы возьмем $X_0 = (A + B) / 2$. Последующие приближения считаются по рекуррентной формуле:

$$X_{i+1} = X_i + L \cdot F(X_i).$$

(4-3)

Условие завершения процесса приближения $|X_{i+1} - X_i| < E$. За искомое решение X^* принимают последнее приближение X_{i+1}.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
X_i	X0	Приближение корня, рассчитанное на шаге i.	real
X_{i+1}	X1	Новое приближение корня на шаге i + 1.	real
A	A	Левая граница интервала изоляции корня.	real
B	B	Правая граница интервала изоляции корня.	real
E	E	Погрешность вычисления корня.	real
F(X)	Z(X)	Значение исследуемой функции в точке X.	real
F'(X)	Z1(X)	Значение первой производной F(X).	real
X^*	ITER	Искомое значение корня.	real
L	AL	Коэффициент приводящий F(X) к функции φ(X).	real

Структурограмма метода итераций.

<table>
<tr><td colspan="3">Принять за первое приближение корня X1 = (A + B)/ 2;</td></tr>
<tr><td colspan="3">ABS(Z1(A)) >= ABS(Z1(B))
True False</td></tr>
<tr><td colspan="2">Коэффициент AL = -1/ Z1(A);</td><td>Коэффициент AL = -1/ Z1(B);</td></tr>
<tr><td></td><td colspan="2">Считать ранее найденное приближения корня старым: X0 = X1;
Найти новое приближение корня: X1 = X0 + AL * Z(X0);</td></tr>
<tr><td></td><td colspan="2">Делать до тех пор, пока не выполнится |X1-X0| < E;</td></tr>
<tr><td colspan="3">Вывод найденного значения корня X1.</td></tr>
</table>

4.2. Программирование методов уточнения корней

Пример 111. Создать пользовательскую библиотеку ROOT в которую включить функции уточнения корней методами дихотомии, касательных, хорд и итераций. Все функции имеют в качестве входных параметров A, B левую и правую границы интервала изоляции корня, E погрешность вычисления и имя функции, корень которой нужно найти. Выходным возвращаемым параметром является численное значение корня.

```
UNIT ROOT;        {Библиотека функций уточнения корней нелинейных}
INTERFACE         {уравнений}
TYPE  FUN = FUNCTION(X: REAL): REAL;
FUNCTION  DIX(A, B, E: REAL; Z: FUN): REAL;
FUNCTION  KAS(A, B, E: REAL; Z, Z1, Z2: FUN): REAL;
FUNCTION  XORD(A, B, E: REAL; Z, Z2: FUN): REAL;
FUNCTION  ITER(A, B, E: REAL; Z, Z1: FUN): REAL;
IMPLEMENTATION
FUNCTION  DIX;    {Уточнение корня методом дихотомии}
VAR  X: REAL;
BEGIN
REPEAT
X := (A + B)/2;       {Деление отрезка по палам }
IF  Z(X) * Z(A) <= 0
THEN  B := X          {Корень находится в левой половине отрезка}
ELSE  A := X          {Корень - в правой половине}
UNTIL (B - A)/2 < E;
DIX := (A + B)/2      {Приближение корня с погрешностью E}
END;
FUNCTION  KAS; {Уточнение корня методом касательных}
VAR X0, X1: REAL;
```

```
BEGIN
IF Z2(B)*Z(B) >= 0
THEN  X1 := B              {Начальное приближение корня}
ELSE  X1 := A;
REPEAT
X0 := X1;                  {Предыдущее приближение}
X1 := X0 -Z(X0)/Z1(X0)  {Новое приближение}
UNTIL  ABS(X1-X0) < E;
KAS := X1            {Окончательное приближение корня}
END;
FUNCTION  XORD;       {Уточнение корня методом хорд}
VAR X0, X1, C: REAL;
BEGIN
IF Z2(B)*Z(B) >= 0         {Начальное приближение корня}
THEN  BEGIN C := B; X1 := A  END
ELSE  BEGIN C := A; X1 := B  END;
REPEAT
X0 := X1;                  {Предыдущее приближение}
X1 := X0 - Z(X0)*(C - X0)/(Z(C) - Z(X0))    {Новое приближение}
UNTIL ABS(X1 - X0) < E;
XORD := X1          {Окончательное приближение корня}
END;
FUNCTION   ITER;        {Уточнение корня методом итераций}
VAR X0, X1, AL: REAL;
BEGIN
X1 := (A + B)/2;           {Начальное приближение корня}
IF  ABS(Z1(A)) >= ABS(Z1(B))
THEN  AL := -1/Z1(A)
ELSE  AL := -1/Z1(B);
REPEAT
X0 := X1;                  {Предыдущее приближение}
X1 := X0 + AL * Z(X0)    {Новое приближение}
UNTIL  ABS(X1 - X0) < E;
ITER := X1                 {Окончательное приближение корня}
END;
END.
```

4.3. Отделение и уточнение корней

Процесс отделения корней связан с определением интервалов изоляции корней. Напомним, интервалом изоляции корня R_j называется промежуток $[X_i, X_{i+1}]$, содержащий единственный действительный корень уравнения F(X). Иногда отделение корней осуществляется на основании физического смысла задачи или из анализа ее упрощенной модели. Когда нет информации о расположении корней, можно воспользоваться табличным способом, при котором для выделения интервалов изоляции корней отрезок $[X_0, X_n]$ разбивают на N частей $[X_i, X_{i+1}]$,

где: $X_i = X_0 + I \cdot dX$, $X_{i+1} = X_i + dX$, $I = \overline{0, n-1}$.

Каждый интервал $[X_i, X_{i+1}]$ исследуется на наличие корня с помощью критерия

$F(X_i) \cdot F(X_{i+1}) < 0$.

Когда количество корней и интервалы изоляции определены, переходят ко второму этапу решения, уточняя корни одним из выбранных методов.

Пример 112. Табличным способом найти интервалы изоляции корней в уравнении

$X^3 - 1.25 \cdot X^2 - 4.4944 \cdot X + 5.918 = 0$

в интересующей нас области изменения неизвестной $X \in [X_0, Xn]$ и уточнить их методом касательных. Для уточнения корней использовать библиотеку функций ROOT.

Решение этой задачи следует начать с объявления функции F(X), которая представляет собой левую часть нелинейного уравнения:

$$F(X) = X^3 - 1.25 \cdot X^2 - 4.4944 \cdot X + 5.918. \quad (4\text{-}4)$$

Поскольку в заданном методе уточнения корней нужны первая и вторая производные, дифференцируя F(X), найдем эти функции:

$$F'(X) = 3 \cdot X^2 - 2.5 \cdot X - 4.4944, \quad (4\text{-}5)$$

$$F''(X) = 6 \cdot X - 2.5. \quad (4\text{-}6)$$

Исходными данными для алгоритма изоляции корней табличным способом являются: отрезок $[X_0, Xn]$ и количество интервалов n. По формуле $dX = (X_n - X_0)/n$ можно вычислить шаг табуляции F(X). Далее нужно осуществить перебор всех интервалов $\{ [X_i, X_{i+1}] / I = \overline{0, n-1} \}$ и для каждого из них проверить условие наличия корня:

$$F(X_i) \cdot F(X_{i+1}) < 0. \quad (4\text{-}7)$$

Для тех интервалов, где это условие истинно, то есть функция пересекает ось X, значения X_i и X_{i+1} записываются в таблицу M отдельной строкой. Количество строк таблицы M, равное числу K и есть искомое количество корней в уравнении $F(X) = 0$. Когда все интервалы проанализированы и число корней $K > 0$, можно результаты вывести на монитор.

В первую очередь следует вывести общее число найденных корней на отрезке [X_0, Xn]. Затем, построить таблицу, содержащую четыре графы: порядковый номер корня, левую и правую границы интервала изоляции корня, уточненное методом касательных значение корня.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
X_i	XI	Начало интервала [X_i, X_{i+1}].	real
X_{i+1}	XI1	Конец интервала [X_i, X_{i+1}].	real
X_0	X0	Начало отрезка [X_0, X_n].	real
X_n	XN	Конец отрезка [X_0, X_n].	real
dX	DX	Шаг табулирования функции F(X).	real
i	I	Номер интервала [X_i, X_{i+1}].	integer
n	N	Число интервалов [X_i, X_{i+1}].	integer
E	E	Погрешность вычисления корня.	real
F(X)	F(X)	Значение исследуемой функции в точке X.	real
F'(X)	F1(X)	Значение первой производной F(X).	real
F"(X)	F2(X)	Значение второй производной F(X).	real
k	K	Количество изолированных корней.	integer
-	M[J, 1]	Левая граница интервала изоляции корня j.	real
-	M[J, 2]	Правая граница интервала изоляции корня j.	real

Структурограмма

	Ввести значения X0, XN, N и E; Обнулить счетчик числа корней K = 0; Вычислить dX = (XN - X0)/N.
	Для I от 0 до N с шагом 1 делать:
	Нахождение границ i-го отрезка: XI := X + I * dX; XI1 := XI + dX;
	F(XI) * F(XI1) <= 0 True Найден корень: K := K + 1; Изоляция интервала: M[K,1]:= XI; M[K,2]:=XI1
	Вывод числа корней K.
	Для J от 1 до K с шагом 1 сделать:
	Вывод номера корня J, границ интервала изоляции M[J, 1], M[J, 2] и уточненного значения корня.

```
PROGRAM  PR112;        {Программа изоляции корней и их уточнения}
{методом касательных }
USES  ROOT;            {Пользовательская библиотека, содержащая }
{функции уточнения корней нелинейных уравнений}
VAR    M: ARRAY [1..10,1..2] OF REAL; N, I, K, J: INTEGER;
X0, DX, XI, XI1, XN, E: REAL;
{$F+}
FUNCTION  F(X: REAL): REAL;      {Исследуемая функция формула 4-4}
BEGIN  F := X*(X*(X - 1.25) - 4.4944) + 5.918  END;
FUNCTION  F1(X: REAL): REAL;     {Первая производная формула 4-5}

BEGIN  F1 := X*(3*X - 2.5) - 4.4944  END;
FUNCTION  F2(X: REAL): REAL;     {Вторая производная формула 4-6}
BEGIN  F2 := 6*X - 2.5   END;
{$F-}
BEGIN
WRITELN('Введите значение X0, XN, N, E');
READLN(X0, XN, N, E); DX := (XN - X0)/N;
K := 0;                  {Число корней}
FOR I := 0 TO N-1  {Табуляция функции}
DO BEGIN
     XI := X0 + I * DX; XI1 := XI + DX;
     IF F(XI) * F(XI1) <= 0   {Критерий наличия корня в [XI, XI1] формула 11-
6}
     THEN BEGIN
          K := K + 1; M[K,1] := XI; M[K, 2] := XI1
          END
     END;
WRITELN('Найдено корней: ', K:2);
WRITELN('Интервалы изоляции и уточненные значения корней: ');
FOR J := 1 TO K
DO WRITELN(J:2, M[J, 1]:12:6, M[J, 2]:12:6,
    KAS(M[J, 1], M[J, 2], E, F, F1, F2):14:9);
END.
```

Замечание. Табличный способ отделения корней основан на критерии (4-7) и находит корни только в том случае, если F(X) пересекает ось абсцисс.

5. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Одной из основных задач линейной алгебры является решение системы линейных алгебраических уравнений (СЛАУ) вида:

$$\begin{cases} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + \ldots + A_{1n}x_n = A_{1,n+1} \\ A_{21}x_1 + A_{22}x_2 + A_{23}x_3 + \ldots + A_{2n}x_n = A_{1,n+1} \\ \ldots \\ A_{n1}x_1 + A_{n2}x_2 + A_{n3}x_3 + \ldots + A_{nn}x_n = A_{n,n+1} \end{cases} \quad (5\text{-}1)$$

где A_{ij} - заданные элементы расширенной матрицы системы уравнений $i = \overline{1,n}$, $j = \overline{1,n+1}$. Решением СЛАУ называется процесс вычисления величин $\{X_i / i = \overline{1,n}\}$. Система совместна тогда и только тогда, когда есть такая совокупность значений $\{X_i / i = \overline{1,n}\}$, что выполняются все равенства (12-1). Методы решения делятся на две группы: прямые (точные) и итерационные. Прямые методы являются универсальными и применяются для систем сравнительно невысокого порядка $n \leq 200$. В этих методах используются операторы арифметического цикла и заранее известно количество арифметических операций (меньше чем n^3). Итерационные методы используют для СЛАУ высоких порядков со слабо заполненными матрицами.

5.1. Метод Гаусса

Метод Гаусса, или метод последовательного исключения неизвестных, основан на приведении матрицы коэффициентов A_{ij} из исходной системы (5-1) к треугольному виду:

$$\begin{cases} x_1 + A'_{12}x_2 + A'_{13}x_3 + \ldots + A'_{1n}x_n = A_{1,n+1}; \\ x_2 + A'_{23}x_3 + \ldots + A'_{2n}x_n = A'_{2,n+1}; \\ \ldots \\ x_n = A'_{n,n+1}. \end{cases} \quad (5\text{-}2)$$

Процесс получения системы (12-2) называется прямым ходом алгоритма Гаусса. На втором этапе систему уравнений (12-2) преобразуют к виду:

$$\begin{cases} x_n = A'_{n,n+1}; \\ x_{n-1} = A_{n-1,n+1} - A'_{n-1,n}x_n; \\ \ldots \\ x_1 = A'_{1,n+1} - A'_{12}x_2 - A'_{13}x_3 - \ldots - A'_{1n}x_n. \end{cases} \quad (5\text{-}3)$$

После чего вычисляются величины X_n, X_{n-1}, X_{n-2}, .., X_1. Этот процесс называют обратным ходом алгоритма Гаусса.

Прямой ход начинается с выбора главного элемента и заключается в анализе коэффициентов A_{11}, A_{21}, .., A_{n1}. Требуется найти номер уравнения K, имеющего наибольший коэффициент при X_1:

$$\max_i |A_{i1}|, \; i = \overline{1,n}. \quad (5\text{-}4)$$

После чего уравнения с номерами 1 и K меняются местами. Цель этой перестановки следующая: во-первых, найти коэффициент A_{i1} при X_1 отличный от нуля; во-вторых, повысить точность расчетов потому, что следующее действие состоит в делении всех A_{1j} на величину A_{11}:

$$A'_{1j} = A_{1j} / A_{11},\ j = \overline{1, n+1}. \qquad (5\text{-}5)$$

После этих преобразований в (5-2) найдено первое уравнение. Теперь с коэффициентами уравнений 2, 3, .., n нужно сделать линейные преобразования такие, чтобы $A'_{i1} = 0$, $i = \overline{2, n}$, то есть исключить неизвестную X_1 из этих уравнений. Формулы такого пересчета имеют вид:

$$A'_{ij} = A_{ij} - A_{i1} \cdot A'_{1j},\ i = \overline{2, n},\ j = \overline{1, n+1}. \qquad (5\text{-}6)$$

Далее, для оставшихся уравнений $i = \overline{2, n}$ повторяются вычисления (5-4, 5-5, 5-6). То есть для A'_{i2}, $i = \overline{2, n}$ находится наибольший по модулю элемент, имеющий номер k. Далее, второе и k-ое уравнения переставляются местами. Все коэффициенты второго уравнения делятся на A'_{22} и в уравнениях с номерами $i = \overline{3, n}$ пересчитываются коэффициенты для исключения переменных X_2 из этих уравнений.

Этот процесс повторяется до тех пор, пока уравнение с номером n не примет вид:

$$X_n = A'_{n,\,n+1}.$$

В обратном ходе алгоритма Гаусса последовательно просчитываются формулы (12-3) которые для программирования удобно записать в виде:

$$X_n = A_{n,\,n+1};\ X_s = A_{s,\,n+1} - \sum_{j=s+1}^{n} A'_{sj} \cdot X_j \qquad (5\text{-}7)$$

Пример 113. Разработать алгоритм и составить программу решения СЛАУ в общем виде (12-1) методом Гаусса. Считать, что число неизвестных не более 20 ($n \le 20$).

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число неизвестных, число уравнений.	integer
m	M	Число коэффициентов с учетом свободного члена (M = N + 1).	integer
A	A[I, J]	Расширенная матрица коэффициентов, I = 1 .. N, J = 1.. M.	real
X_j	X[j]	Искомое решение системы уравнений, J = 1 .. N.	real
j	J	Номер неизвестной.	integer
i	I	Номер уравнения.	integer
k	K	Номер уравнения с максимальным коэффициентом $\max_l \lvert A_{lj} \rvert$, l = i .. N.	integer
$\max_l \lvert A_{lj} \rvert$	AMAX	Численное значение коэффициента $\lvert A_{ki} \rvert$.	real

-	R	Рабочая ячейка, в том числе используется для обмена местами A_{ii} и A_{ki}	real

Примечание:

- операторы с 1 по 9 соответствуют Прямому ходу алгоритма Гаусса;
- операторы 10 - 12 соответствуют Обратному ходу алгоритма Гаусса;
- 1 ввод с клавиатуры n - числа неизвестных и матрицы A – коэффициентов СЛАУ (5-1);
- 2-3 в столбце **I** поиск наибольшего по модулю элемента в строках с **I** по **N**;
- 4-5 перестановка местами уравнений k и **I**;
- 6-7 в уравнении **I** коэффициенты разделить на A_{ii} при X_i;
- 8-9 исключение X_i из уравнений с i+1 до **N;**
- 10 нахождение **Xn**;
- 11-12 нахождение X_{n-1}, X_{n-2}, .. , X_1 по формуле (5-7).

Структурограмма

```
1   Ввод N  и A[i, j], для i = 1.. n, j = 1.. n + 1; M = N+1.
    Для I от 1 до N с шагом 1 делать:
      AMAX = ABS(A[I, I]); K = I;
2     Для L от I + 1 до N с шагом 1 делать:
        ABS(A[L, I]) < AMAX
                                                    True
3       AMAX = ABS(A[L, I]; K = L;
4     K = I
                                                    True
        Для J от I до M с шагом 1 делать:
5       R = A[I, J]; A[I, J] = A[K, J]; A[K, J] = R;
      R = A[I, I];
      R = 0
                                                    True
      Система несовместна или вырождена: вывод сообщения, конец программы.
6     Для J от I до M с шагом 1 делать:
7       A[I, J] = A[I, J] / R;
8     Для L от I + 1 до N с шагом 1 делать:
        R = A[L, I];
          Для J от I + 1 до N с шагом 1 делать:
9           A[L, J] = A[L, J] - A[I, J] * R;
10  X[N]=A[N, M];
11  Для I от N-1 до 1 с шагом -1 делать:
      X[I] = A[I, M];
        Для J от I + 1 до N с шагом 1 делать:
12      X[I] = X[I] - A[I, J] * X[J];
    Вывод X[i] на экран.
```

```
PROGRAM PR113;                { Решение системы линейных алгебраических
  уравнений методом Гаусса }
CONST G = 20;                 { Максимальное число неизвестных }
VAR   X: ARRAY [1 .. G] OF REAL; A: ARRAY [1 .. G, 1 .. G + 1] OF REAL;
I, N, M, K, L, J: INTEGER; AMAX, R: REAL;
BEGIN
WRITELN('Введите число неизвестных'); READLN(N); M := N + 1;
WRITELN ('Введите коэффициенты A[I, J]');
FOR I := 1 TO N DO FOR J := 1 TO M DO READ(A[I, J]);
{ Прямой ход алгоритма Гаусса }
FOR I := 1 TO N
DO BEGIN   { Выбрать из уравнений с номером I, I+1, ... , N то в котором
коэффициент при X I имеет наибольшее по модулю значение
( точнее - найти номер K этого уравнения ) }
AMAX := ABS(A[I, I]); K := I;
FOR L := I + 1 TO N
DO IF ABS(A[L, I]) > AMAX THEN BEGIN AMAX := ABS(A[L, I]); K := L
END;
{Если K <> I, то поменять местами I-ое и K-е уравнения}
IF K <> I
THEN FOR J := I TO M
        DO BEGIN R := A[I, J]; A[I, J] := A[K, J]; A[K, J] := R END;
R := A[I, I];
IF R = 0
THEN BEGIN WRITELN('Система несовместна или вырождена.'); EXIT
END;
{ Разделить I-ое уравнение на его коэффициент при XI }
FOR J := I TO M DO A[I, J] := A[I, J] / R;
 { Исключить XI из уравнений начиная с I + 1 по N }
FOR L := I + 1 TO N
DO  BEGIN R := A[L, I];
      FOR J := I + 1 TO M
      DO A[L, J] := A[L, J] - A[I, J]*R END;
      END;
{Конец прямого хода алгоритма Гаусса. Начало обратного хода.
Вычисление XN}
X[N] := A[N, M];
```

```
{ Вычислить X[N-1], X[N-2], ... , X[1] }
FOR I := N -1 DOWNTO 1
DO BEGIN X[I] := A[I, M];
FOR J := I + 1 TO N
DO X[I] := X[I] - A[I, J]*X[J] END;
{ Конец обратного хода. Вывод результатов X1, X2 ... , XN }
FOR J := 1 TO N DO WRITELN(' X[', J:1,'] = ', X[J]:11:8)
END.
```

5.2. Метод Жордана

Жордан ввел в рассмотрение систему линейных алгебраических уравнений следующего вида:

$$\begin{cases} Y_1 = A_{11}(-X_1) + A_{12}(-X_2) + A_{13}(-X_3) + \ldots + A_{1n}(-X_n) + A_{1,n+1}; \\ Y_2 = A_{21}(-X_1) + A_{22}(-X_2) + A_{23}(-X_3) + \ldots + A_{2n}(-X_n) + A_{2,n+1}; \\ \quad \ldots \\ Y_n = A_{n1}(-X_1) + A_{n2}(-X_2) + A_{n3}(-X_3) + \ldots + A_{nn}(-X_n) + A_{n,n+1}; \end{cases} \tag{5-8}$$

где $\{X_i / i = 0 .. n\}$ свободные переменные, а $\{Y_i / i = 0 .. n\}$ зависимые переменные. При $\{Y_i / i = 0 .. n\}$ система (5-8) превращается в систему (5-1). Основная идея метода Жордана состоит в линейном преобразовании системы (5-8) с целью поменять местами переменные X_i и Y_i, чтобы (5-8) привести к следующему виду:

$$\begin{cases} X_1 = A^*_{11}(-Y_1) + A^*_{12}(-Y_2) + A^*_{13}(-Y_3) + \ldots + A^*_{1n}(-Y_n) + A^*_{1,n+1}; \\ X_2 = A^*_{21}(-Y_1) + A^*_{22}(-Y_2) + A^*_{23}(-Y_3) + \ldots + A^*_{2n}(-Y_n) + A^*_{2,n+1}; \\ \quad \ldots \\ X_n = A^*_{n1}(-Y_1) + A^*_{n2}(-Y_2) + A^*_{n3}(-Y_3) + \ldots + A^*_{nn}(-Y_n) + A^*_{n,n+1}; \end{cases} \tag{5-9}$$

Теперь в системе (5-9), положив $\mathbf{Y_i} = 0$ для $i = \overline{1,n}$, можно найти значения $X_i = A_{i,\,n+1}$, $i = \overline{1,n}$, которые и являются решением исходной системы (5-1).

Переход от системы (5-8) к системе (5-9) осуществляется по шагам и на каждом шаге сводится к замене свободной переменной X_S на зависимую переменную Y_R. Правила обмена свободных и зависимых переменных можно сформулировать следующим образом:

Пусть есть матрица коэффициентов, и элемент A_{RS} является разрешающим (он стоит на пересечении X_S и строки Y_R).

	$-X_1$	…	$-X_s$	…	$-X_j$	…	$-X_n$	B_i
Y_1	A_{11}	…	$A_{1\,s}$	…	$A_{1\,j}$	…	$A_{1\,n}$	$A_{1,\,n+1}$
	⋮		⋮		⋮		⋮	⋮
Y_r	A_{r1}	…	$A_{r\,s}$	…	$A_{r\,j}$	…	$A_{r\,n}$	$A_{r,\,n+1}$
	⋮		⋮		⋮		⋮	⋮
Y_i	A_{i1}	…	$A_{i\,s}$	…	$A_{i\,j}$	…	$A_{i\,n}$	$A_{i,\,n+1}$
	⋮		⋮		⋮		⋮	⋮
Y_n	A_{n1}	…	$A_{n\,s}$	…	$A_{n\,j}$	…	$A_{n\,n}$	$A_{n,\,n+1}$

2. Разрешающий элемент становится величиной обратной $A_{RS} = \frac{1}{A_{RS}}$.

3. Элементы разрешающей строки делятся на A_{rs}, включая свободный член:

$$A_{rj} = \frac{A_{rj}}{A_{rs}}, \quad j = 1,2,\ldots, s-1, s+1,\ldots, n+1.$$

4. Элементы разрешающего столбца делятся на A_{rs} с обратным знаком:

$$A_{is} = \left(\frac{-A_{is}}{A_{rs}}\right), \quad i = 1,2,\ldots, r-1, r+1,\ldots n.$$

5. Остальные элементы вычисляются по правилу прямоугольника:

$$A_{ij} = A_{ij} - A_{is}\frac{A_{rj}}{A_{rs}}; \quad i = 1,2,\ldots, r-1, r+1,\ldots, n; \quad j = 1,2,\ldots, s-1, s+1,\ldots, n+1.$$

Пример 114. Разработать алгоритм и составить программу решения СЛАУ в общем виде (5-1) методом Жордана. Считать, что число неизвестных не более 20.

Для программирования метода Жордана нужно сформулировать порядок выбора разрешающего элемента. Можно использовать следующее правило: поскольку все свободные переменные необходимо заменить на зависимые, то свободные переменные будем брать по порядку с номера 1 до номера n. Для неизвестной X_i нужно найти разрешающий элемент A_{ki}, причем искать нужно среди оставшихся уравнений, в которых зависимые переменные - Y_i. Чтобы разделить уравнения на две группы, будем поступать следующим образом: по аналогии с

методом Гаусса за разрешающий берется элемент $\max_{l} |A_{lj}|$, l = i, i + 1, … , n. Пусть номер такого элемента k.

Уравнения i и k меняются местами. Далее в уравнении i делается перестановка местами переменных X_i и Y_i и пересчитываются коэффициенты матрицы A. Использование главного (наибольшего по модулю) элемента приводит к тому, что, во-первых, разрешающие элементы всегда располагаются на главной диагонали матрицы A. Все уравнения с зависимыми переменными Y_i находятся ниже разрешающего элемента $A_{i\,i}$. Во-вторых, точность расчетов повышается за счет выбора в качестве разрешающего - наибольшего по модулю $A_{l\,i}$. Если наибольший по модулю элемент равен 0, значит система несовместна ($A_{i\,i} = 0$ и $A_{i,\,n+1} \neq 0$), или имеет бесконечное множество решений ($A_{i\,i} = 0$ и $A_{i,\,n+1} = 0$). В-третьих, столбцы соответствующие свободным переменным, уже переведенным в зависимые, находятся левее разрешающего элемента. Коэффициенты матрицы **A**, для этих столбцов можно не пересчитывать, поскольку они в дальнейших расчетах влияния на величины $\{X_i / i = \overline{1,n}\}$ не оказывают.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число неизвестных, число уравнений.	integer
m	M	Число коэффициентов с учетом свободного члена (M = N + 1).	integer
$A_{i\,j}$	A[I, J]	Расширенная матрица коэффициентов, I = 1 .. N, J = 1.. M.	real
X_j	X[j]	Искомое решение системы уравнений, J = 1.. N.	real
k	K	Номер уравнения с максимальным значением, $\max_{l} \|A_{lj}\|$ l = i .. N.	integer
$\max_{l} \|A_{lj}\|$	AMAX	Численное значение коэффициента $\|A_{ki}\|$.	real
-	R	Рабочая ячейка, в том числе используется для обмена местами A_{ii} и A_{ki}	real

Структурограмма

	Ввод N и A[i, j], для i = 1 .. n, j = 1 .. n + 1; M = N + 1;
	Для I от 1 до N с шагом 1 делать:
	AMAX = ABS(A[I, I]); K = I;
1	Для L от I + 1 до N с шагом 1 делать:
	ABS(A[L, I]) < AMAX
	True
2	AMAX = ABS(A[L, I]; K = L;
	K = I
	True
	Для J от I до M с шагом 1 делать:
3	R = A[I, J]; A[I, J] = A[K, J]; A[K, J] = R;
4	R = A[I, I];
	R = 0
	True
	Система несовместна или вырождена: вывод сообщения, конец программы.
5	Для L от 1 до N с шагом 1 делать:
	L ≠ I
	True
	Для J от I + 1 до M с шагом 1 делать:
6	A[L, J = A[L, J] - A[I, J] ∗ A[L, I] / R;
7	Для J от I + 1 до N с шагом 1 делать:
8	A[I, J] = A[I, J] / R;
	Вывод X[i] на экран.

Примечание:

- операторы с 1 по 2 соответствуют поиску в столбце **I** наибольшего по модулю

элемента в строках с **I** по **N**;

- 2-3 перестановка местами уравнений **K** и **I;**
- 4-5 пересчет коэффициентов **A[L, J]** для **L ≠ I;**

в уравнении I коэффициенты разделить на Aii.

```
PROGRAM PR114;        { Решение системы линейных алгебраических
  уравнений методом жордановых исключений }
```

```
CONST G = 20;              {МАКСИМАЛЬНОЕ ЧИСЛО НЕИЗВЕСТНЫХ}
VAR A:       ARRAY [1 .. G, 1 .. G + 1] OF REAL;
I, N, M, K, L, J: INTEGER; AMAX, R: REAL;
BEGIN
WRITELN('Введите число неизвестных'); READLN(N); M := N + 1;
WRITELN ('Введите коэффициенты A[I, J]');
FOR I := 1 TO N DO FOR J := 1 TO M DO READ(A[I, J]);
FOR I := 1 TO N    { Исключение XI из независимых переменных }
DO BEG         { Выбрать из уравнений с номером I, I + 1,..., N , то в котором коэффициент при XI имеет наибольшее по модулю значение ( точнее - найти номер K этого  уравнения ) }
      AMAX := ABS(A[I, I]); K := I;
      FOR L := I + 1 TO N
      DO IF ABS(A[L, I]) > AMAX
          THEN BEGIN  AMAX := ABS(A[L, I]); K := L  END;
{ Если K ≠ I, то поменять местами I-ое и K-ое уравнения}
      IF K <> I
      THEN FOR J := I TO M
            DO BEGIN  R := A[I, J]; A[I, J] := A[K, J]; A[K, J] := R  END;
      R := A[I, I];
      IF R = 0
      THEN BEGIN WRITELN('Система несовместна или вырождена.');
            EXIT END;
      FOR L := 1 TO N      {Пересчет коэффициентов A[L, J]}
      DO  IF L <> I
          THEN FOR J := I + 1 TO M
                DO A[L, J] := A[L, J] - A[I, J]*A[L, I]/R;
{ Разделить I - ое  уравнение на его коэффициент при Xi }
      FOR J := I + 1 TO M DO A[I, J] := A[I, J] / R;
      END;
FOR J := 1 TO N DO WRITELN(' X[', J:1, ']= ', A[J, M]:11:8)
END.
```

Часто решение системы линейных алгебраических уравнений является этапом решения более сложной математической задачи. В этом случае удобно иметь готовую подпрограмму решения СЛАУ.

Пример 115. Используя программу по примеру 114 написать подпрограмму решения СЛАУ (5-1) методом Жордана и оформить ее в виде процедуры SLAY. Входными параметрами должны быть: N - число неизвестных и A - матрица ко-

эффициентов, имеющая в разделе TYPE описание MAT = ARRAY [1 .. G, 1 .. G + 1] OF REAL, где G - максимальное число неизвестных. Возвращаемые значения X должны иметь тип VEKT = ARRAY[1 .. G] OF REAL.

```
PROCEDURE  SLAY(N: INTEGER; A: MAT; VAR X: VEKT);
VAR  I, M, K, L, J: INTEGER; AMAX, R: REAL;
BEGIN
M := N + 1;
FOR I := 1 TO N
DO BEGIN
    AMAX := ABS(A[I, I]); K := I;
    FOR L := I + 1 TO N
    DO IF ABS(A[L, I]) > AMAX
    THEN BEGIN AMAX := ABS(A[L, I]); K := L END;
    IF K <> I
    THEN FOR J := I TO M
    DO BEGIN R := A[I, J]; A[I, J] := A[K, J]; A[K, J] := R END;
    R := A[I, I];
    IF R = 0
    THEN BEGIN WRITELN('Система несовместна или вырождена.');
            EXIT END;
    FOR L := 1 TO N
    DO  IF L <> I THEN FOR J := I + 1 TO M
    DO A[L, J] := A[L, J] - A[I, J]*A[L, I]/R;
    FOR J := I + 1 TO M DO A[I, J]:=A[I, J]/R;
    END;
FOR J := 1 TO N DO X[J] := A[J, M]
END.
```

5.3. Итерационный метод Зейделя

Для решения СЛАУ методом Зейделя систему (5-1) нужно преобразовать к виду:

$$\begin{cases} X_1 = (A_{1,n+1} - A_{12}X_2 - A_{13}X_3 - \ldots - A_{1n}X_n)/A_{11}; \\ X_2 = (A_{2,n+1} - A_{21}X_1 - A_{22}X_2 - \ldots - A_{2n}X_n)/A_{22}; \\ \qquad\qquad \cdots \\ X_n = (A_{n,n+1} - A_{n1}X_1 - A_{n2}X_2 - \ldots - A_{nn-1}X_{n-1})/A_{nn}. \end{cases} \qquad (5\text{-}10)$$

Задавая начальное приближение, например $\{ X_i^0 = A_{i,n+1} / i = \overline{1,n} \}$ и, подставляя эти значения в (5-10), можно найти следующее приближение:

$$\begin{cases} X_1^1 = (A_{1,n+1} - A_{12}X_2^0 - A_{13}X_3^0 - \ldots - A_{1n}X_n^0)/A_{11}; \\ X_2^1 = (A_{2,n+1} - A_{21}X_1^1 - A_{23}X_3^0 - \ldots - A_{2n}X_n^0)/A_{22}; \\ \quad \cdots \\ X_n^1 = (A_{n,n+1} - A_{n1}X_1^1 - A_{n2}X_2^1 - \ldots - A_{n,n-1}X_{n-1}^1)/A_{nn}. \end{cases} \quad (5\text{-}11)$$

Вычисляя значения $\{ x_i^1 / i = \overline{1,n} \}$, и, подставляя их в (12-11) на место $\{ x_i^0 / i = \overline{1,n} \}$, можно вычислить новое приближения $\{ x_i^2 / i = \overline{1,n} \}$ и т.д.

В общем виде для шага K систему можно представить:

$$\begin{cases} X_1^k = (A_{1,n+1} - A_{12}X_2^{k-1} - A_{13}X_3^{k-1} - \ldots - A_{1n}X_n^{k-1})/A_{11}; \\ X_2^k = (A_{2,n+1} - A_{21}X_1^k - A_{23}X_3^{k-1} - \ldots - A_{2n}X_n^{k-1})/A_{22}; \\ \quad \cdots \\ X_n^k = (A_{n,n+1} - A_{n1}X_1^k - A_{n2}X_2^k - \ldots - A_{n,n-1}X_{n-1}^k)/A_{nn}. \end{cases} \quad (5\text{-}12)$$

Если система совместна и выполнены условия сходимости, то используя (5-12), можно найти решение (5-1). В процессе вычислений возможны следующие три ситуации. Во-первых, процесс сходиться, то есть за конечное число шагов M мы можем получить решение с заданной погрешностью. Во-вторых, процесс зацикливается, когда количество итераций велико, а требования к точности решения не выполнены. В-третьих, итерационный процесс расходиться и решить (5-1) с помощью (5-12) не удается.

Условие достаточности сходимости итерационного процесса можно записать в виде:

$$|A_{ii}| \geq \frac{1}{2} \sum_{j=1}^{n} |A_{ij}|. \quad (5\text{-}13)$$

То есть модуль диагонального элемента матрицы A_{ii} (5-1) должен быть не меньше суммы модулей остальных коэффициентов при неизвестных. Хотя бы для одного уравнения неравенство (5-13) должно быть строгим, чтобы избежать зацикливания. Условие сходимости обеспечивается путем перестановки уравнений и неизвестных, а также линейных преобразований для (5-1).

Пример 116. Написать подпрограмму проверки критерия сходимости (5-13) и оформить ее в виде функции PRS. Входными параметрами должны быть: **A** - матрица коэффициентов и N - число неизвестных. Возвращаемое значение должно иметь тип BOOLEAN. Событие True должно соответствовать сходимости СЛАУ (5-1), а False - нарушению условия сходимости (5-13).

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число неизвестных, число уравнений.	integer
i	I	Номер уравнения.	integer
j	J	Номер переменной.	integer
$A_{i\,j}$	A[I, J]	Матрица коэффициентов, I = 1 .. N, J = 1 .. N.	real
-	T	Счетчик числа равенств в (12-13)	integer
-	S	Рабочая ячейка, используемая для накопления сумм.	real

```
FUNCTION PRS(A: MAT; N: INTEGER): BOOLEAN;
VAR   I, J, T: INTEGER; S: REAL;
BEGIN
PRS := TRUE; T := 0; FOR I := 1 TO N        { Перебор неравенств (12-13) }
DO BEGIN S := 0;
     FOR J := 1 TO N
     DO BEGIN S := S + ABS(A[I, J]);
          IF ABS(A[I, I]) < S/2 THEN PRS := FALSE;
          IF ABS(A[I, I]) = S/2 THEN T := T + 1
          END;
     END;
IF T = N THEN PRS := FALSE {Нет ни одного строгого неравенства}
END;
```

В качестве условий завершения вычислений (5-12) предложено несколько критериев:

1. По абсолютной погрешности: $\max\limits_{i}\left|X_i^k - X_i^{k-1}\right| < A$; $i = \overline{1, n}$.

Недостаток этого критерия состоит в том, что не учитываются масштабы осей:

2. По относительной погрешности: $\max\limits_{i}\left|\dfrac{X_i^k - X_i^{k-1}}{X_i}\right| < E$; $i = \overline{1, n}$.

Недостаток этого критерия состоит в том, что в начале осей он плохо работает в связи с необходимостью деления на 0.

Критерий Гаусса: $\min\limits_{i}(\min(\dfrac{X_i^{k-1}}{X_i^k}, \dfrac{X_i^k}{X_i^{k-1}})) > D$, где D = 1 - E, $i = \overline{1, n}$.

Этот критерий, по сути, является оценкой относительной погрешности, но переносит область расчетов из 0 в 1, где у ЭВМ наибольшая точность вычислений.

Пример 117. Разработать алгоритм и программу решения СЛАУ (5-1) методом Зейделя. Для завершения расчетов использовать критерий Гаусса.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число неизвестных, число уравнений.	integer
m	M	Максимальное число итераций.	integer
$A_{i\,j}$	A[I, J]	Расширенная матрица коэффициентов, I = 1 .. N, J = 1 .. N + 1.	real
X_i	X[I]	Искомое решение системы уравнений, I = 1 .. N.	real
-	T	Число несущественных приращений.	integer
i	I	Номер уточняемой переменной.	integer
k	K	Номер итерации K = 1 .. M.	integer
D	D	Допустимое отношение двух последовательных приближений.	real
x_i^{k-1}	XS	Предыдущее значение x_i на K-1 шаге.	real
-	S	Рабочая ячейка, используемая для накопления сумм.	real

Структурограмма

```
PROGRAM PR117;     { Решение системы линейных алгебраических уравнений
        методом Зейделя }
CONST G = 20;              { Максимальное число неизвестных }
VAR X: ARRAY [1 .. G] OF REAL; A: ARRAY [1 .. G, 1 .. G + 1] OF REAL;
T, N, M, K, I, J: INTEGER; S, XS, E, D: REAL;
BEGIN
WRITELN('Введите число неизвестных'); READLN(N);
WRITELN ('Введите коэффициенты матрицы A');
FOR I := 1 TO N DO FOR J := 1 TO N + 1 DO READ(A[I, J]);
FOR J := 1 TO N
DO X[J] := 0;    {Присвоение начальных значений неизвестным X[J] }
WRITELN('Укажите число итераций -> M и погрешность -> E');
READLN(M, E); D := 1.0 - E;
FOR K := 1 TO M            { Итерации от 1 до M }
DO BEGIN
    T := 0;             { Количество несущественных приращений в итерации }
    FOR I := 1 TO N        { Оценка неизвестной XI на шаге K }
    DO BEGIN
        XS := X[I]; S := 0;
        FOR J := 1 TO N DO S := S + A[I, J] * X[J];
        X[I] := (-S + A[I, I]*X[I] + A[I, N + 1])/A[I, I];
        IF ABS(X[I]) > ABS(XS)       {Анализ приращения по XI}
        THEN BEGIN IF D <= ABS(XS/X[I]) THEN T := T + 1 END
        ELSE  IF D <= ABS(X[I]/XS) THEN T := T + 1
        END;
    IF T = N THEN BREAK     { Приращения всех XI несущественны }
    END;
IF K >= M THEN WRITELN('Сходимость процесса очень медленная');
WRITELN('Номер итерации -> ', K);
FOR I := 1 TO N              {Вывод текущих значений XI}
DO WRITELN(' X[', I:2,'] = ', X[I]:11:8)
END.
```

Замечание. Итерационные методы обладают свойством самоисправления ошибок.

5.4. Оценка погрешности решения СЛАУ

Погрешность вычислений $\{X_i / i = \overline{1,n}\}$ определяется точностью выполнения на ЭВМ арифметических операций. Количество таких операций зависит от числа уравнений и, например, в методе Гаусса примерно равняется $2n^3/3$. Оценить погрешность найденного решения можно путем проверки невязок:

$$R_i = \left| A_{i,n+1} - \sum_{j=1}^{n} A_{ij} \cdot X_j \right|, \qquad (5\text{-}14)$$

где величина R_i представляет собой разность между правой и левой частями уравнения с номером i из (5-1).

Пример 118. Разработать процедуру ERR оценки невязки решения СЛАУ. На входе процедуры даны матрица коэффициентов **A** и вектор решения **X**, на выходе получить максимальную невязку R и номер уравнения K соответствующего этой невязке.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число уравнений.	integer
$A_{i\,j}$	A[I, J]	Расширенная матрица коэффициентов, I=1 .. N, J=1 .. N+1.	real
X_i	X[I]	Оцениваемое решение системы уравнений.	real
i	I	Номер уравнения по порядку.	integer
k	K	Номер уравнения с максимальной невязкой.	integer
$\max_i \lvert R_i \rvert$	R	Максимальное значение невязки, I =1.. n.	real
-	S	Рабочая ячейка, используемая для накопления сумм.	real

Структурограмма

Невязку R и номер уравнения K положить равными 0;

Для **I** от 1 до **N** с шагом 1 делать:

- Положить S = A[i, n+1];
- Для **J** от 1 до **N** с шагом 1 делать:
  - S = S - A[I, J] ∗ X[J]
- R < ABS(S)
  - True
  - {Найденная невязка больше прежней} R = ABS(S); K = I.

Для использования этой процедуры в основной программе в разделе TYPE нужно объявить матрицу MAT = ARRAY [1 .. N, 1 .. N] OF REAL; и вектор VECT = ARRAY[1 .. N] OF REAL;

```
PROCEDURE  ERR(A: MAT; X: VECT; N: INTEGER; VAR R: REAL;
   VAR K: INTEGER);
VAR  I: INTEGER; S: REAL;
BEGIN
R := 0; K := 0;
FOR I := 1 TO N   { Последовательный просмотр уравнений}
DO BEGIN
     S := A[I, N + 1];
     FOR J := 1 TO N DO S := S - A[I, J] * X[J];
     IF  R < ABS(S) THEN BEGIN R := ABS(S); K := I END
     END
END;
```

Если все невязки равны нулю, то найдено точное решение. Этому событию соответствует K = 0. Если K > 0, то решение приближенное и значение K соответствует номеру уравнения с наибольшей невязкой R.

Оглавление

1. ПРИБЛИЖЕНИЕ ФУНКЦИЙ 3
1.1. Линейная интерполяция 3
1.2. Интерполирование полиномом Лагранжа 5
1.3. Выбор и нахождение параметров эмпирической формулы 7
2. ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ 13
2.1. Программирование интерполяционно-квадратурных формул 13
2.2. Интегрирование аналитических функций 17
2.3. Интегрирование табличных функций 19
2.4 Метод Монте-Карло 20
3. МЕТОДЫ ЧИСЛЕННОГО ДИФФЕРЕНЦИРОВАНИЯ 24
3.1. Программирование формул численного дифференцирования 24
3.2. Табулирование аналитически заданной функции и ее первых трех производных 27
3.3. Интерполирование производных таблично заданных функций 28
4. ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ 31
4.1. Уточнение корня 31
4.2. Программирование методов уточнения корней 37
4.3. Отделение и уточнение корней 39
5. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ 42
5.1. Метод Гаусса 42
5.2. Метод Жордана 46
5.3. Итерационный метод Зейделя 51
5.4. Оценка погрешности решения СЛАУ 55