Лабораторный практикум

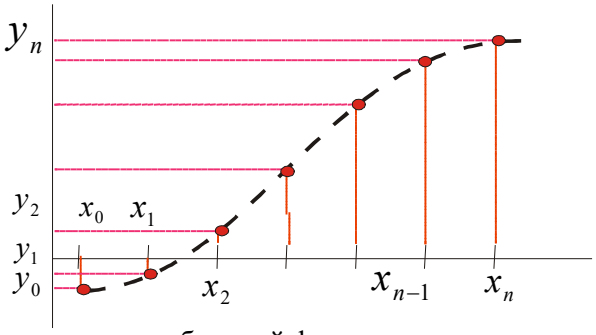
Численные алгоритмы

Численные алгоритмы – решение алгебраических и трансцендентных уравнений, решение систем линейных алгебраических уравнений, решение систем обыкновенных дифференциальных уравнений, решение уравнений в частных производных, оптимизация, обработка числовых данных. Из этого спектра алгоритмов рассмотрим наиболее простые алгоритмы типа «разделяй-и-властвуй». При сохранении общей идеологии в практической реализации алгоритмов существуют заметные различия.

Приближение функций. Понятие приближения функций.

В своей практической деятельности мы часто сталкиваемся с ситуацией, когда те или иные параметры системы или объекта можно определить только в конечном числе точек (узлов). А для проведения численного моделирования необходимо знание значений параметров системы во всей области определения. Наиболее часто исходная функция $y = f(x)$ приближается многочленами.

$$f(x) \approx P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$



X	Y
x_0	y_0
x_1	y_1
x_2	y_2
…	…
…	…
x_n	y_n

Рис.1. Представление табличной функции

В общем случае задачу можно сформулировать следующим образом: Данную функцию $f(x)$ требуется приближённо заменить (аппроксимировать) некоторой функцией $\varphi(x)$, так чтобы отклонение $\varphi(x)$ от $f(x)$ во всей области определения было наименьшим.

Интерполяция – приближённая замена табличной функции аналитической функцией**, значения которой в узлах таблицы совпадают со значениями табличной функции.**

Существуют несколько схем реализующих подобную замену. Они отличаются друг от друга как формой исходных таблиц (равноотстоящие узлы или произвольно расположенные), так и по способу выбора узлов, содержащих информацию, используемую для построения интерполирующей функции. Интерполирующая функция может строиться сразу для всего рассматриваемого интервала аргумента или отдельно для разных частей этого интервала.

Одним из основных понятий, связанных с табличными функциями, является понятие конечной разности.

Пусть $y = f(x)$- заданная функция, а $\Delta x = x_{i+1} - x_i = h = const$ - фиксированная величина приращения аргумента (шаг). Тогда выражение

$$\Delta y \equiv \Delta f(x) = f(x + \Delta x) - f(x)$$

называется первой конечной разностью функции y.

В приведённых формулах используется разностный оператор - Δ.

$\Delta x = x_{i+1} - x_i$ - разность значений переменной x в двух соседних узлах таблицы.

$\Delta y_i = y_{i+1} - y_i$ - разность значений табличной функции в двух соседних узлах таблицы.

Аналогично определяются конечные разности функции y высших порядков.

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$$
$$\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i$$
$$\Delta^4 y_i = \Delta^3 y_{i+1} - \Delta^3 y_i$$
.....
$$\Delta^n y_i = \Delta^{n-1} y_{i+1} - \Delta^{n-1} y_i$$

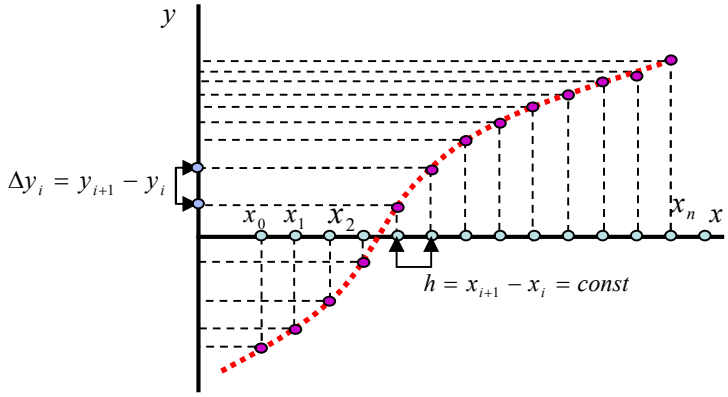


Рис.2. Графическое представление конечных разностей и узлов табличной функции.

Таблица 1. Табличная функция и конечные разности. Горизонтальная таблица.

Аргумент	**Функция**	**1-я к.раз**	**2-я к.раз**	**3-я к.раз**	**4-я к.раз**	**5-я к.раз**
x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0	Δy_0	$\Delta^2 y_0$	$\Delta^3 y_0$	$\Delta^4 y_0$	$\Delta^5 y_0$
x_1	y_1	Δy_1	$\Delta^2 y_1$	$\Delta^3 y_1$	$\Delta^4 y_1$	
x_2	y_2	Δy_2	$\Delta^2 y_2$	$\Delta^3 y_2$		
x_3	y_3	Δy_3	$\Delta^2 y_3$			
x_4	y_4	Δy_4				
x_5	y_5					

Из таблицы 1 видно, что разность более высокого прядка строится на основе разностей предшествующего порядка. Следует обратить внимание на индексацию разностей. Нижний индекс означает, к какому узлу таблицы принадлежит данная разность. Верхний индекс ($\Delta^i y_j$) означает порядок конечной разности (а не возведение в степень *i*)**.** Для таблицы, содержащей *n* узлов можно построить конечные разности до (*n-1*) порядка включительно.

Горизонтальная таблица разностей используется при построении интерполяционного многочлена Ньютона (первая и вторая интерполяционные формулы Ньютона).

Лабораторная работа № 1.

Интерполирование.

Первая интерполяционная формула Ньютона.

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_2(x - x_0)(x - x_1)(x - x_2) + \ldots\ldots \quad (1)$$

где x_i - координаты узлов табличной функции, а a_i - коэффициенты, значения которых определяются исходя из значений табличной функции в соответствующих узлах. В зависимости от количества членов многочлена получаем различные приближения интерполирующей функции к табличной функции. Так первые два члена представляют линейную интерполяцию между двумя узлами (x_0, x_1), первые три члена – квадратичную интерполяцию, первые четыре члена – кубическую интерполяцию, и т.д. Если функция задана в $(n+1)$ узле, то можно построить многочлен n -ой степени.

Рассмотрим процесс вычисления коэффициентов a_i. Полагая в (1) $x = x_0$, получаем $P_n(x_0) = y_0 = a_0$, т.е. $a_0 = y_0$.

Для определения a_1 вычислим конечную разность многочлена в точках x и $x+h$

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + ...$$

$$P_n(x+h) = a_0 + a_1(x + h - x_0) + a_2(x + h - x_0)(x + h - x_1) + \quad (2)$$

$$+ a_3(x + h - x_0)(x + h - x_1)(x + h - x_2) + ...$$

Пологая в (2) $x = x_0$, получаем

$$\Delta P(x_0) = P_n(x+h) - P_n(x) \Rightarrow \Delta P(x_0) = a_1 * h ,$$

$$\Delta P(x_0) = \Delta y_0 \Rightarrow \qquad a_1 = \Delta y_0 / (1!*h) \quad (3)$$

Последовательно, продолжая этот процесс, получим

$$a_j = \Delta^i y_{n-1} / (i!*h^i) \quad (4)$$

Следует отметить, что формулы (2,3) записаны для интерполяции на интервале $[x_0, x_1]$ для «опорного» нулевого узла x_0 . Для i -го узла в формулах нужно произвести замену индексов при переменных x_0 , x_1 , x_2 и т.д. на x_i , x_{i+1} , x_{i+2} и т.д.

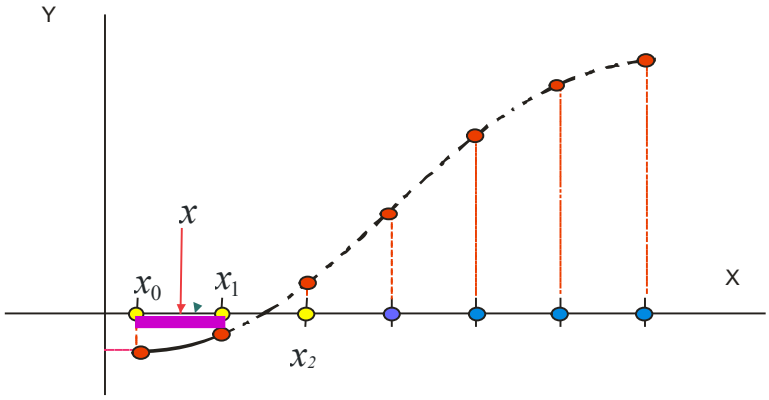


Рис. 3. Выбор узлов для квадратичной интерполяции на участке $[x_0, x_1]$.

Первой интерполяционной формуле Ньютона соответствуют конечные разности, расположенные в горизонтальных рядах Таблицы 1. Интерполирующая функция как бы «прижимается» к левой границе табличной функции. Для построения интерполирующей функции на правой границе табличной функции используется вторая интерполяционная формула Ньютона.

Задание 1.

Используя первую интерполяционную формулу Ньютона построить интерполяционный многочлен второй степени для таблично заданной функции.

Алгоритм решения.

Для тестовой функции $f(x) = \sin(x)$ в интервале $x = [x_0, x_n]$ построить табличную функцию $y(x)$, где $(n+1)$ – количество узлов, отстоящих друг от друга на величину h.

- В тестовых расчётах положить $x_0 = 0, x_n = 1,\ n \le 100$. *Индексы узлов начинаются с «0»*
- Вычислить первые и вторые конечные разности (*создать массивы*)
- Вычислить коэффициенты $a_0,\ a_1,\ a_2$ для каждого интервала (x_{i+1}, x_i) (*создать массивы*) Нижние индексы $a_0,\ a_1,\ a_2$ (см.табл.) относятся к коэффициентам полинома (2), верхние – к номеру соответствующего узла таблицы.

$a_0 = y_i \qquad a_1 = \Delta y_i / (1! * h) \qquad a_2 = \Delta^2 y_i / (2! * h^2)$

x	y	Δy	$\Delta^2 y$	a_0	a_1	a_2
x_0	y_0	Δy_0	$\Delta^2 y_0$	a_0^0	a_1^0	a_2^0
x_1	y_1	Δy_1	$\Delta^2 y_1$	a_0^1	a_1^1	a_2^1
x_2	y_2	Δy_2	$\Delta^2 y_2$	a_0^2	a_1^2	a_2^2
x_3	y_3	Δy_3	$\Delta^2 y_3$	a_0^3	a_1^3	a_2^3
x_4	y_4	Δy_4				
x_5	y_5					

- Сформировать массив значений аргумента $x_{i+1/2}$ в середине каждого интервала(x_{i+1}, x_i). *[Индексация $(i+1/2)$ означает только, что значение $x_{i+1/2}$ берётся в середине интервала (x_{i+1}, x_i), а сам массив имеет целочисленную индексацию]*
- Для этих значений аргумента вычислить значения тестовой функции $f(x) = \sin(x)$ и интерполяционного многочлена $P(x)$.
- Оценить разницу полученных значений $f(x)$ и $P(x)$. Результаты сравнений значений $f(x)$ и $P(x)$представить в виде таблиц
- Провести численный эксперимент для различного числа узлов. Как результат зависит от количества узлов в заданном интервале.

1) $y(x) = \sin^k(\pi x^m)$;
2) $y(x) = \cos^k(\pi x^m)$;
3) $y(x) = \sin^k(\pi x^{1/m})$;
4) $y(x) = \cos^k(\pi x^{1/m})$;
5) $y(x) = \mathrm{tg}^k(\pi x^m / 4)$;
6) $y(x) = \mathrm{tg}^k(\pi x^{1/m} / 4)$;
7) $y(x) = \pm ax^4 \pm bx^3$;
8) $y(x) = (a + bx^m)^{-k}$;
9) $y(x) = (a + bx^m)^{-1/k}$;
10) $y(x) = (a + bx^{1/m})^{-k}$;
11) $y(x) = (a + bx^{1/m})^{-1/k}$;
12) $y(x) = x^k / (a + bx^m)$;
13) $y(x) = x^k / (a + bx)^m$;
14) $y(x) = x^{1/k} / (a + bx)^m$;
15) $y(x) = x^k / (a + bx)^{1/m}$;
16) $y(x) = x^{1/k} / (a + bx)^{1/m}$;
17) $y(x) = (a + x^2)^k / (b + x^4)^m$;
18) $y(x) = (a + x^2)^{1/k} / (b + x^4)^m$;
19) $y(x) = (a + x^2)^k / (b + x^4)^{1/m}$;
20) $y(x) = (a + x^2)^{1/k} / (b + x^4)^{1/m}$;
21) $y(x) = x^k / (a + bx^m)$;
22) $y(x) = x^{1/k} / (a + bx^m)$;
23) $y(x) = x^k / (a + bx^{1/m})$;
24) $y(x) = x^{1/k} / (a + bx^{1/m})$;
25) $y(x) = \ln^k(1 + x^m)$;
26) $y(x) = \ln^{1/k}(1 + x^m)$;

Интерполяционная формула Лагранжа. (Глобальная интерполяция)

Для интерполирования табличной функции с неравноотстоящими узлами используется формула Лагранжа.

На отрезке $[x_0, x_n]$ даны $(n+1)$ различных значений аргумента - $x_0, x_1, x_2, \ldots x_n$ и известны значения функции $f(x)$ в этих узлах - $y_i = f(x_i)$. Нужно построить многочлен $L_n(x)$ степени не выше n, такой, что $L_n(x_i) = y_i$ для $i = 0, 1, 2, \ldots n$

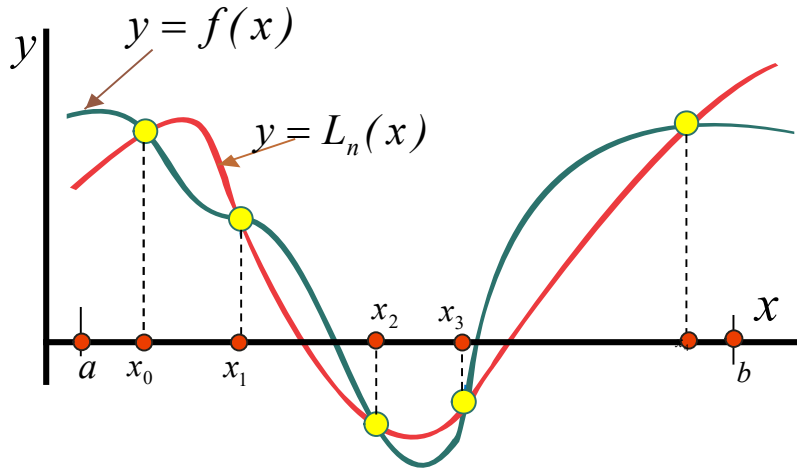


Рис. 4. Табличная функция и многочлен Лагранжа.

Для построения такого многочлена сначала нужно решить частную задачу - построить вспомогательные многочлены $p_i(x)$, такие, что $p_i(x_j)=0$ при $j \neq i$ и $p_i(x_i)=1$, т.е.

$$p_i(x_i) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \qquad (5)$$

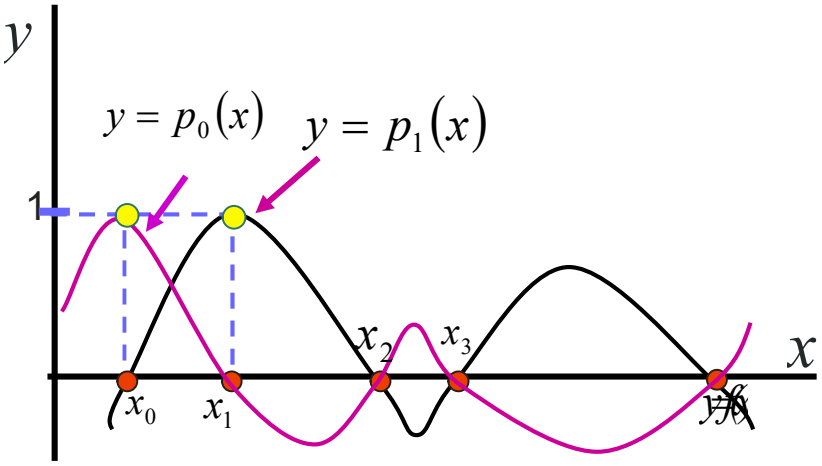


Рис.5. Вспомогательные многочлены.

Так как вспомогательный многочлен $p_i(x)$ обращается в нуль в n узлах (кроме *i-го*), то он может быть записан как

$$p_i(x) = C_i(x-x_0)(x-x_1)(x-x_2)(x-x_3)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n) \qquad (6)$$

где C_i - постоянный коэффициент.

Полагая в (6) $x = x_i$ и учитывая, что $p_i(x_i)=1$, получаем

$$C_i = \frac{1}{(x_i-x_0)(x_i-x_1)(x_i-x_2)(x_i-x_3)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \qquad (7)$$

и, следовательно, многочлен (6) запишется как

$$p_i(x) = \frac{(x-x_0)(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \qquad (8)$$

Чтобы выполнилось исходное требование $L_n(x_i) = y_i$, интерполяционный многочлен должен иметь вид

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \qquad (9)$$

Следует отметить, что интерполяционный многочлен Лагранжа, в отличие от других интерполяционных функций, содержит в явном виде значения y_i, что при решении некоторых задач может оказаться важным фактором.

Задание 2.

Используя интерполяционную формулу Лагранжа построить интерполяционный многочлен для таблично заданной функции.

Алгоритм решения.

Для тестовой функции $f(x) = \sin(x)$ в интервале $x = [x_0, x_n]$ построить табличную функцию $y(x)$, где n – количество узлов, отстоящих друг от друга на величину h. В общем случае расстояние между узлами может быть различным.

- В тестовых расчётах положить $x_0 = 0, x_n = 1,\ n \le 100$.
- Сформировать массив значений аргумента x_i.
- Для этих значений аргумента вычислить значения тестовой функции $f(x) = \sin(x)$.
- Сформировать массив значений аргумента $(x1)_i$ в середине каждого интервала (x_{i+1}, x_i).
- Для этих значений аргумента вычислить значения тестовой функции $f(x) = \sin(x)$.
- Построить интерполяционный многочлен для всего интервала $x = [x_0, x_n]$

1. Построить вспомогательные многочлены (8) для этого использовать алгоритм трёх вложенных циклов.

$$p_i(x) = \frac{(x-x_0)(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

   Для каждого значения x (всего их $k = (n-1)$) из массива значений аргумента $(x1)_i$ построить n вспомогательных многочленов $p_i((x1)_i)$
   Внешний цикл – перебор значения $(x1)_i$ по индексам от 0 до $(n-1)$
2. Внутренний цикл - перебор значения x_i по индексам от 0 до (n) и формирование матрицы вспомогательных многочленов $r(k,i)$ размерностью $(n-1) \times n$
3. Самый внутренний цикл – вычисление произведений в числителе и знаменателе (8) с перебором значений x_j в сомножителях $(x_i - x_j)$.

- Вычислить значения многочлена (9) в серединах интервалов (x_{i+1}, x_i), используя алгоритм двух вложенных циклов.

Студентам: Постарайтесь сначала написать собственный алгоритм.
Если это вызывает сложности, воспользуйтесь приведённым кодом.

```
real x(0:100),y(0:100),x1(0:100),y1(0:100)
   real r(0:100,0:100),p(0:100)

   write(*,*) 'vvesti a b n'
   read(*,*) a,b,n
! ----- шаг таблиц, аргумент, функция ----------------
   h=(b-a)/n
   do i=0,n
     x(i)=h*i
     y(i)=f(x(i))
   enddo
  ! ----- аргумент, функция в серединах интервалов----------------
   do i=0,n-1
     x1(i)=h*i+h/2
     y1(i)=f(x1(i))
   enddo
  !----реализация формулы (9)
!------ L_n(x) = sum_{i=0}^{n} y_i (x-x_0)(x-x_1)(x-x_2)...(x-x_{i-1})(x-x_{i+1})...(x-x_n) / (x_i-x_0)(x_i-x_1)(x_i-x_2)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)   (9)
```

```
      do k=0,n-1    !--цикл по x ≡ x1
        do i=0,n    !--цикл по xi
          u=1
          d=1
          do j=0,n !--цикл по xj в сомножителях: числитель (x − xj) и знаменатель (xi − xj)
            if (i.ne.j) then
              u=u*(x1(k)-x(j))
              d=d*(x(i)-x(j))
            endif
          enddo
          r(k,i)=u/d
        enddo
      enddo
 !-----значение многочлена (9) , вычисление суммы
      do k=0,n-1
        sum=0
        do i=0,n
          sum=sum+y(i)*r(k,i)
        enddo
        p(k)=sum
      enddo
!-----------------------------------------------------
      write(*,*) '    x        y       x1       y1'
      write(*,10) (x(i),y(i),x1(i),y1(i),i=0,n)
      write(*,*)  '       p              y1'
      write(*,11)  (p(j),y1(j),j=0,n-1)
10    format(4(x,f8.5))
11    format(2(x,e14.7))
      end
!------тестовая функция-------------------
      real function f(x)
      real x
       f=sin(x)
      return
      end
```

Лабораторная работа № 2.

Численное дифференцирование.

Для численного определения значений производных можно использовать представление их через конечные разности

$$y' \approx \Delta y / \Delta x \quad (10)$$

Это представление является *аппроксимацией производной с помощью конечных разностей.*

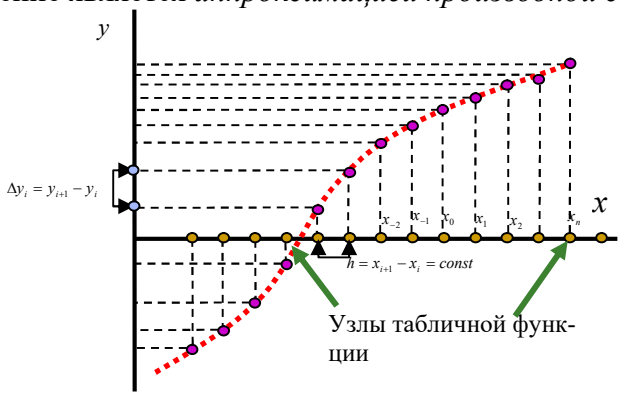


Рис.6. Графическое представление табличной функции.

Для табличной функции возможны представления производной с помощью левых разностей, правых разностей и центральных разностей.

Рассмотрим табличную функцию $f(x_i)$, определённую на сетке с постоянным шагом $\Delta x = h$ (Рис.6), первая конечная разность $\Delta y_i = y_i - y_{i-1}$. Определим первую производную как

$$y_i' = \frac{(y_i - y_{i-1})}{h} \quad (10)$$

Таким образом, производную в точке x_i выразили через разность значений функции в этой точке y_i и в точке слева от нее y_{i-1}, т. е. через левые разности. Аналогично можно поступить, выразив производную с помощью правых разностей

$$y_i' = \frac{(y_{i+1} - y_i)}{h} \quad (11)$$

или с помощью центральных разностей

$$y_i' = \frac{(y_{i+1} - y_{i-1})}{2h} \quad (12)$$

По такому же принципу можно определить производные более высокого порядка:

$$y_i'' = \frac{(y_{i+1}' - y_{i-1}')}{h} = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \quad (13)$$

Такие же соотношения можно получить с помощью ряда Тейлора

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2!} f''(x)(\Delta x)^2 + \frac{1}{3!} f'''(x)(\Delta x)^3 + \ldots\ldots \quad (14)$$

Пусть функция $f(x)$ записана в виде таблицы

$$f(x_i) = y_i \qquad (i = 1,2,3,\ldots n)$$

При $x = x_1$, $\Delta x = -h$ ряд Тейлора с точностью до членов порядка h^2 запишется как

$$y_0 = y_1 - y_1' h + O(h^2) \quad (15)$$

Отсюда производная в точке $x = x_1$ определяется как

$$y_1' = \frac{y_1 - y_0}{h} + O(h) \quad (16)$$

Это выражение совпадает с ранее полученным выражением аппроксимации первой левосторонней производной первого порядка $(r = 1)$. Также можно получить выражение для правосторонней производной.

Рассмотрим выражения для первой и второй производных второго порядка погрешности.

$$\begin{aligned} y_2 &= y_1 + y_1' h + \frac{1}{2!} y_1'' h^2 + \frac{1}{3!} y_1''' h^3 + O(h^4) \\ y_0 &= y_1 - y_1' h + \frac{1}{2!} y_1'' h^2 - \frac{1}{3!} y_1''' h^3 + O(h^4) \end{aligned} \quad (17)$$

Вычитая из первого уравнения второе, получим

$$y_1' = \frac{y_2 - y_0}{2h} + O(h^2) \quad (18)$$

что совпадает с аппроксимацией центральными разностями. Погрешность имеет второй порядок.

Складывая уравнения, оценим погрешность аппроксимации производной второго порядка

$$y_1'' = \frac{y_2 - 2y_1 + y_0}{h^2} + O(h^2) \quad (19)$$

Данное выражение также совпадает с аппроксимацией центральными разностями.

Для численного определения производных можно использовать различные интерполяционные полиномы с различным количеством членов. На практике более удобным оказывается использование полиномов Лагранжа, т.к. в них в явном виде присутствуют значения табличной функции в узлах. В зависимости от количества используемых узлов получаем выражения для производных функции $f(x)$ с различной погрешностью вычислений.

Рассмотренный источник погрешностей – погрешность аппроксимации, которая определяется величиной остаточного члена. При уменьшении шага таблиц h эта погрешность, как правило, уменьшается.

Другие погрешности – неточность значений функции $y = f(x)$ в узлах и погрешность округления вычислений. Эти погрешности возрастают при уменьшении шага h.

Полином Лагранжа имеет вид (9)

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)(x-x_1)(x-x_2)....(x-x_{i-1})(x-x_{i+1})....(x-x_n)}{(x_i-x_0)(x_i-x_1)(x_i-x_2)....(x_i-x_{i-1})(x_i-x_{i+1})....(x_i-x_n)}$$

Рассмотрим случай трёх узлов интерполирования ($n = 2$).
Расписав многочлен как сумму трёх (соответствующих) слагаемых и приведя их к общему знаменателю, для таблиц с постоянным шагом получим следующие выражения для многочлена и первой производной

$$L(x) = \frac{1}{2h^2}\left[(x-x_1)(x-x_2)y_0 - 2(x-x_0)(x-x_2)y_1 + (x-x_0)(x-x_1)y_2\right]$$
$$L'(x) = \frac{1}{2h^2}\left[(2x-x_1-x_2)y_0 - 2(2x-x_0-x_2)y_1 + (2x-x_0-x_1)y_2\right] \quad (20)$$

где $y'''(x_*)$ - значение производной третьего порядка в некоторой внутренней точке $x_* \in [x_0, x_n]$

Запишем значения производных в точках x_0, x_1, x_2

$$y_0' = \frac{1}{2h}(-3y_0 + 4y_1 - y_2) + \frac{h^2}{3}y'''(x_*)$$
$$y_1' = \frac{1}{2h}(y_2 - y_0) - \frac{h^2}{6}y'''(x_*) \quad (21)$$
$$y_2' = \frac{1}{2h}(y_0 - 4y_1 + 3y_2) + \frac{h^2}{3}y'''(x_*)$$

Подобным образом можно вычислить производные для любого количества узлов аппроксимации. При четных значениях n наиболее простые выражения получаются для центральных точек. Вариант с чётным значением n - *аппроксимация с помощью центральных разностей.* Он более удобен и при вычислении вторых производных. Для четырёх узлов производные

$$y_0' = \frac{1}{6h}(-11y_0 + 18y_1 - 9y_2 + 2y_3)$$
$$y_1' = \frac{1}{6h}(-2y_0 - 3y_1 + 6y_2 - y_3)$$
$$y_2' = \frac{1}{6h}(y_0 - 6y_1 + 3y_2 + 2y_3) \quad (22)$$
$$y_3' = \frac{1}{6h}(-2y_0 + 9y_1 - 18y_2 + 11y_3)$$

Для пяти узлов производные

$$y_0' = \frac{1}{12h}\left(-25y_0 + 48y_1 - 36y_2 + 16y_3 - 3y_4\right)$$
$$y_1' = \frac{1}{12h}\left(-3y_0 - 10y_1 + 18y_2 - 6y_3 + y_4\right)$$
$$y_2' = \frac{1}{12h}\left(y_0 - 8y_1 + 8y_3 - 2y_4\right) \qquad (23)$$
$$y_3' = \frac{1}{12h}\left(-y_0 + 6y_1 - 18y_2 + 10y_3 + 3y_4\right)$$
$$y_4' = \frac{1}{12h}\left(3y_0 - 16y_1 + 36y_2 - 48y_3 + 25y_4\right)$$

Аналогичным образом можно получить выражения для вторых производных, дважды продифференцировав соответствующий многочлен Лагранжа.

Для трёх узлов

$$y_0'' = (y_0 - 2y_1 + y_2)/h^2$$
$$y_1'' = (y_0 - 2y_1 + y_2)/h^2$$
$$y_2'' = (y_0 - 2y_1 + y_2)/h^2$$

Для четырёх узлов

$$y_0'' = (2y_0 - 5y_1 + 4y_2 - y_3)/h^2$$
$$y_1'' = (y_0 - 2y_1 + y_2)/h^2$$
$$y_2'' = (y_1 - 2y_2 + y_3)/h^2$$
$$y_3'' = (-y_0 + 4y_1 - 5y_2 - 2y_3)/h^2$$

Для пяти узлов

$$y_0'' = (35y_0 - 104y_1 + 114y_2 - 56y_3 + 11y_4)/12h^2$$
$$y_1'' = (11y_0 - 20y_1 + 6y_2 + 4y_3 - y_4)/12h^2$$
$$y_2'' = (-y_0 + 16y_1 - 30y_2 + 16y_3 - y_4)/12h^2$$
$$y_3'' = (-y_0 + 4y_1 + 6y_2 - 20y_3 + 11y_4)/12h^2$$
$$y_4'' = (11y_0 - 56y_1 + 114y_2 - 104y_3 + 35y_4)/12h^2$$

Задание 1.

Используя интерполяционную формулу Лагранжа построить интерполяционный многочлен для таблично заданной функции.

Алгоритм решения.

Для тестовой функции $f(x) = \sin(x)$ в интервале $[x_0, x_n]$ построить табличную функцию $y(x)$, где $(n+1)$ – количество узлов, отстоящих друг от друга на величину h. В тестовых расчётах положить $x_0 = 0, x_n = 1,\ n = 10$.

- Сформировать массив значений аргумента x_i. Индекс i от 0 до n.
- Сформировать массив значений функции $y(x_i)$.
- Используя интерполяционную формулу Лагранжа для таблиц с постоянным шагом записать выражения производных в схемах с четырьмя $(n = 3)$ и пятью $(n = 4)$ узлами. Вычислить значения производных в этих узлах. Сравнить значения производных в одноименных узлах между собой и тестовой функцией.
- Для узла с индексом «1» вычислить значения левосторонней, центральной и правосторонней производных и сравнить их со значениями, полученными из многочлена Лагранжа.
- Провести численный эксперимент с варьированием величины шага h.
- Вычислить значения вторых производных в схемах с четырьмя и пятью узлами.
- Результаты работы представить в соответствующей форме, удобной для анализа.

==== вариант =======

=0-й= =1-й= =2-й= =3-й= =4-й=
==== тест_1 первые производные
0.1000000E+01 0.9950042E+00 0.9800666E+00 0.9553365E+00 0.9210610E+00
==== 4 узла
0.1000030E+01 0.9949925E+00 0.9800798E+00 0.9552920E+00
==== 5 узлов
0.9999807E+00 0.9950089E+00 0.9800633E+00 0.9553415E+00 0.9210409E+00
==== 2_л = 2_ц = 2 _п =
0.9983342E+00
0.9933466E+00
0.9883591E+00
==== тест _2 вторые производные
0.0000000E+00 -0.9983342E-01 -0.1986693E+00 -0.2955202E+00 -0.3894183E+00
==== 4 узла
-0.1000613E-02 -0.9975135E-01 -0.1985021E+00 -0.2972528E+00
==== 5 узлов
0.8106232E-03 -0.9991601E-01 -0.1986668E+00 -0.2954416E+00 -0.3902406E+00

Лабораторная работа № 3.

Численное интегрирование.

Численное вычисление однократного интеграла называется *механической квадратурой,* двойного – *механической кубатурой*. Соответствующие формулы называются *квадратурными* и *кубатурными* формулами.

Основным инструментом численного интегрирования является представление подынтегральной функции интерполирующим полиномом. Такая аппроксимация позволяет приближённо заменить определённый интеграл конечной суммой

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i y_i \qquad (2)$$

где y_i - значения функции в узлах интерполяции, a_i - числовые коэффициенты. Это выражение называется квадратурной формулой, а правая часть – квадратурной суммой. В зависимости от способа вычисления этой суммы получаются различные численные методы - методы прямоугольников, трапеций, парабол, сплайнов и др.

Квадратурную сумму можно вычислить как

$$\sum_{i=0}^{n} a_i y_i = \sum_{i=1}^{n} \sigma_i \qquad (3)$$

где $\sigma_i = f(\xi_i)\Delta x_i$ - приближённое значение площади элементарной криволинейной трапеции, соответствующей элементарному отрезку $[x_{i-1}, x_i]$.

В качестве точек ξ_i можно выбрать левые точки $\xi_i = x_{i-1}$ или правые точки $\xi_i = x_i$ элементарного интервала $\Delta x_i = x_i - x_{i-1}$. Обозначая $y_i = f(x_i)$ и $\Delta x_i = h_i$ получаем формулы метода прямоугольников для левых и правых сумм

$$\int_a^b f(x)dx \approx h_1 y_0 + h_2 y_1 + ... + h_n y_{n-1} \qquad (4)$$

$$\int_a^b f(x)dx \approx h_1 y_1 + h_2 y_2 + ... + h_n y_n \qquad (5)$$

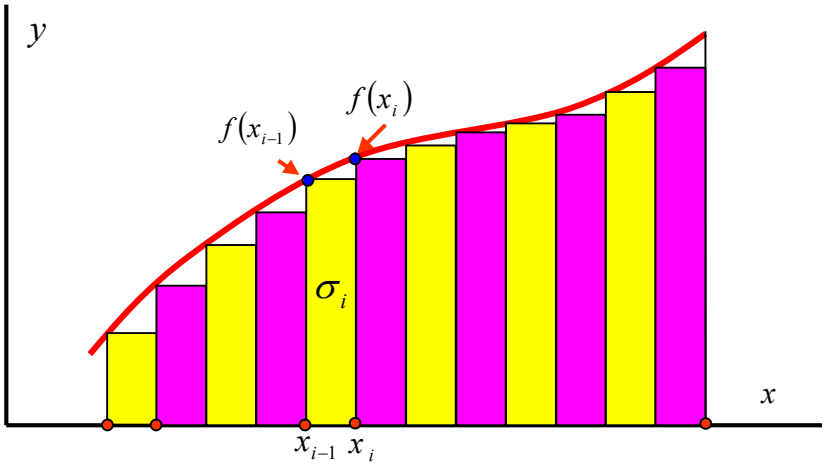


Рис.1. Левые суммы.

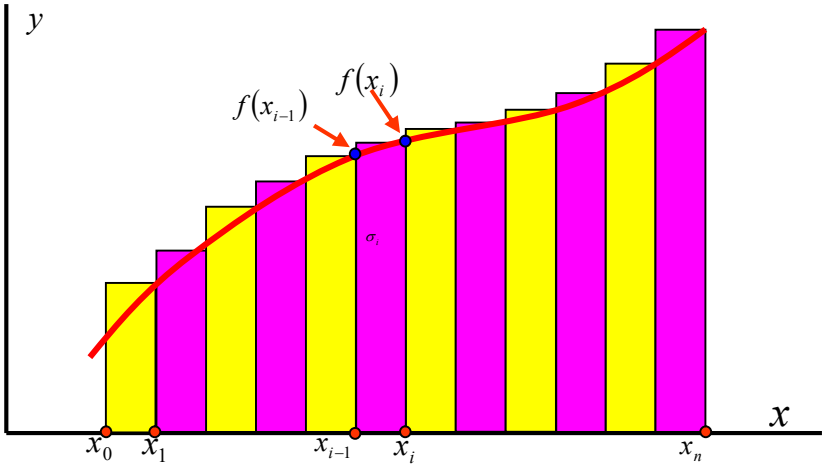


Рис.2. Правые суммы.

Более точным приближение метода прямоугольников является приближение с использованием средних значений функции в элементарных прямоугольниках (в полуцелых узлах).

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_{i-1/2})h_i \qquad (6)$$

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

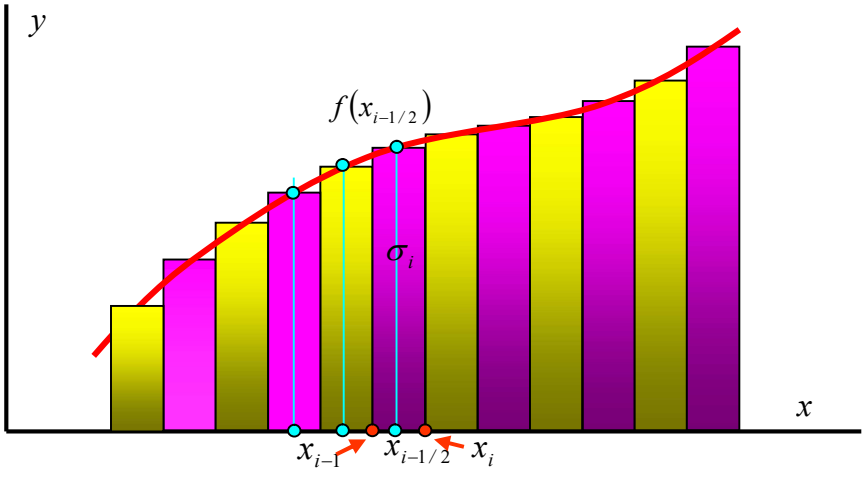


Рис.2. Центральные суммы.

Следующим методом, использующим линейную интерполяцию, является метод трапеций. График функции $f(x)$ представляется в виде ломанной линии, соединяющей точки (x_i, y_i). Площадь всей фигуры приближённо складывается их суммы площадей прямолинейных трапеций (Рис.3). Площадь каждой трапеции определяется как

$$\sigma_i = \frac{y_{i-1} + y_i}{2} h_i \qquad i = 1,2,...,n \quad (7)$$

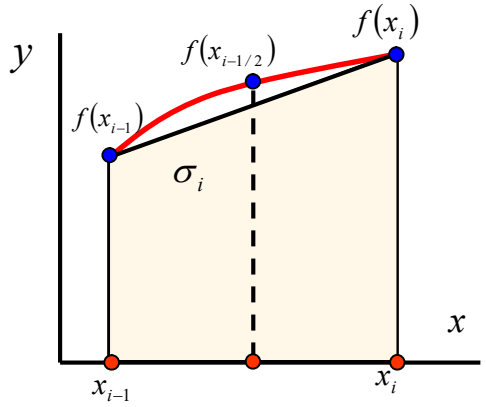


Рис.3. Метод трапеций. Фрагмент – малый интервал.

Таким образом, полная площадь под ломанной кривой (т.е. приближённое значение интеграла запишется как)

$$\int_a^b f(x)dx \approx \frac{1}{2}\sum_{i=1}^{n}(y_{i-1} + y_i)h_i \qquad (8)$$

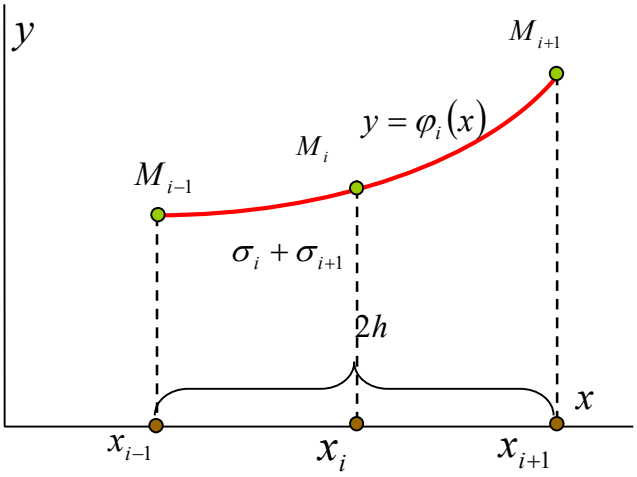
$x_{i-1/2}$

Для таблицы с равноотстоящими узлами $h_i = h = const \quad (i = 1,2,...,n)$ формулы прямоугольников и трапеций принимают соответственно вид

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

$$\int f(x)dx \approx h\left(\sum \frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right) \qquad (9)$$

Повышение точности вычислений достигается за счёт повышения степени интерполяционных многочленов – квадратичной интерполяции (метод Симпсона) и интерполирование с помощью сплайнов.



Рис

Метод Симпсона.

Пусть функция $f(x)$ определена на интервале $[a,b]$, который разобьём на чётное число n равных частей с шагом h. На каждом отрезке $[x_0, x_2], [x_2, x_4], ..., [x_{i-1}, x_{i+1}], ... [x_{n-1}, x_n]$ подынтегральную функцию заменим многочленом второй степени

$$f(x) \approx \varphi_i(x) = a_i x^2 + b_i x + c_i \quad \text{при } x_{i-1} \le x \le x_{i+1} \quad (10)$$

Коэффициенты a_i, b_i, c_i можно вычислить из условия прохождения многочлена через узлы табличной функции.

Рассмотрим интерполяционный многочлен второй степени Лагранжа, проходящий через точки $M_{i-1}(x_{i-1}, y_{i-1}), M_i(x_i, y_i), M_{i+1}(x_{i+1}, y_{i+1})$

$$\varphi_i(x) = \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} y_{i-1} + \\ + \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_i)(x_i-x_{i+1})} y_i + \\ + \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} y_{i+1} \quad (11)$$

Учитывая равенства $x_{i+1} - x_i = x_i - x_{i=1} = h$ можно записать сумму площадей σ_i и σ_{i+1} как определенный интеграл

$$\sigma_i + \sigma_{i+1} = \int_{x_{i-1}}^{x_{i+1}} \varphi_i(x) =$$

$$= \frac{1}{2h^2} \int_{x_{i-1}}^{x_{i+1}} [(x-x_i)(x-x_{i+1})y_{i-1} - 2(x-x_{i-1})(x-x_{i+1})y_i + (x-x_{i-1})(x-x_i)y_{i+1}]dx = \quad (12)$$

$$= \frac{h}{3}(y_{i-1} + 4y_i + y_{i+1})$$

Проводя вычисления на всех отрезках $[x_0, x_2], [x_2, x_4], \ldots, [x_{i-1}, x_{i+1}], \ldots [x_{n-1}, x_n]$, получаем формулу Симпсона

$$\int_a^b f(x) \approx \frac{h}{3}[y_0 + 4(y_1 + y_3 + \ldots + y_{n-1}) + 2(y_2 + y_4 + \ldots + y_{n-2}) + y_n] \quad (13)$$

Задание 1.

Используя формулы левых, правых и центральных сумм вычислить значения интегралов для таблично заданной функции. Произвести вычисления интегралов методами трапеции и Симпсона.

Алгоритм решения.

Для тестовой функции $f(x) = \sin(x)$ в интервале $[x_0, x_n]$ построить табличную функцию $y(x)$, где $(n+1)$ – количество узлов, отстоящих друг от друга на величину h. В тестовых расчётах положить $x_0 = 0, x_n = 1,\ n = 10$.

- Сформировать массив значений аргумента x_i. Индекс i от 0 до n.
- Сформировать массив значений функции $y(x_i)$.
- Провести численные расчеты по всем методам и сравнить результаты.
- Провести численный эксперимент с варьированием величины шага h.

Лабораторная работа № 4.

Поиск корня алгебраического и трансцендентного уравнения.

Задание 1. Деление отрезка пополам.

Функция $f(x)$ задана на отрезке $[a, b]$ и имеет только один корень ξ. Сначала проверяется условие – если $f((a+b)/2) = 0$, то поиск завершён, иначе организуется итерационный процесс, на каждой итерации которого интервал поиска корня функции уменьшается вдвое. Критерием выбора той или иной половины рассматриваемого интервала для дальнейшего поиска является условие, что непрерывная функция $f(x)$ принимает значения разных знаков на концах подынтервалов либо $[a,\ a + (a+b)/2]$, либо $[a + (a+b)/2,\ b]$.

Алгоритм решения:

Полагая, что $b > a$ и ζ и δ - малые значения.

Шаг 1: Вычисляем значения функции $f(x)$ на концах интервала - $f(a)$, $f(b)$. Проверяем условие наличия корня на интервале $[a, b]$ - если корень существует, то переходим к шагу 2, иначе – выход.

Шаг 2: Определяем середину интервала $x_{cp} = (a+b)/2$ и вычисляем в этой точке значение функции $f(x_{cp})$. Если $f(x_{cp}) \neq 0$, то переходим к шагу 3, иначе – выход.

Шаг 3: Вычисляем произведение – $f(a)f(x_{cp})$

Если $f(a)f(x_{cp}) < 0$, то присваиваем $b = x_{cp}$, иначе присваиваем $a = x_{cp}$ и $f(a) = f(x_{cp})$

Шаг 4: Признак завершения счета. Если $abs(b-a) < \delta$, то перейти **к шагу 5**, иначе перейти **к шагу 2**.

Шаг 5: Вычисляем $\xi = (a+b)/2$ и $f(\xi)$. Проверяем - если $f(\xi) < \zeta$, вычисления закончены, иначе уменьшаем значение δ и переходим к **шагу 2.**

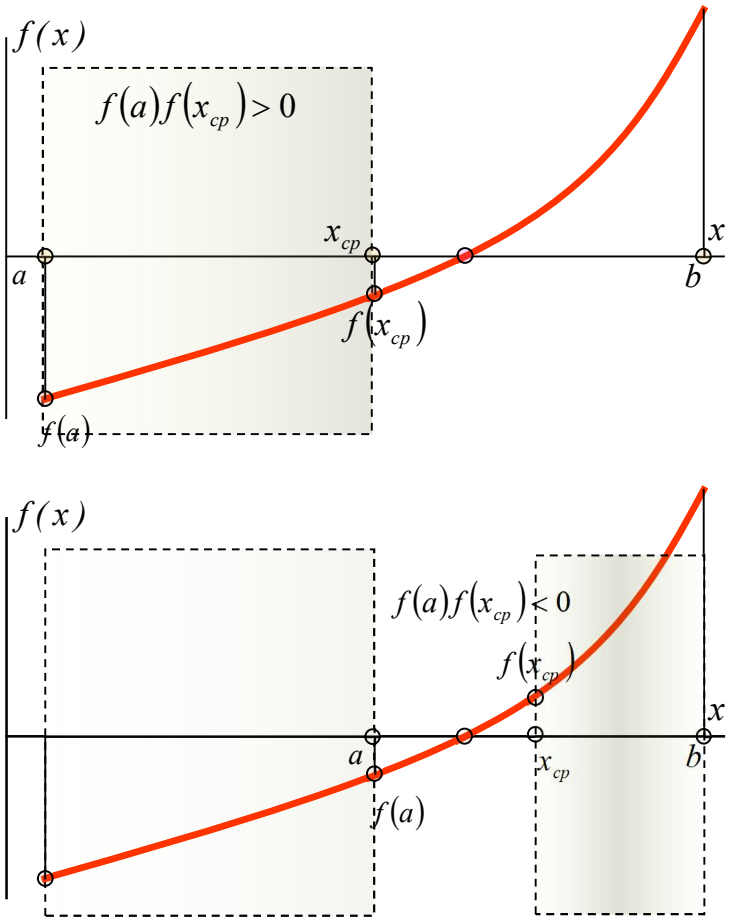


Рис. 21. Алгоритм «Деление отрезка пополам». Затенённые области исключаются из последующих итераций, границы поиска либо a, либо b перемещаются на место x_{cp} в зависимости от знака произведения $f(a)f(x_{cp})$.

Задание 2. Метод хорд.

Более быстрый способ нахождения корня ξ уравнения $f(x) = 0$ на отрезке $[a, b]$ заключается в замене криволинейной функции $f(x)$ прямолинейным отрезком $A - B$ - хордой, соединяющей точки $A[a, f(a)]$ и $B[b, f(b)]$ (Рис.22). Уравнение прямой, проходящей через две точки $A[a, f(a)]$ и $B[b, f(b)]$ записывается как

$$\frac{x-a}{b-a} = \frac{f(x)-f(a)}{f(b)-f(a)} \quad (1)$$

Для доказательства сходимости процесса поиска корня, считаем, что он отделён и вторая производная $f''(x) > 0$ сохраняет знак на отрезке $[a, b]$. Кривая функции $f(x)$ выпукла вниз и, следовательно расположена ниже своей хорды $A - B$ (Рис. 22).

Отсюда, полагая $f(x) = 0$, получаем первое приближение корня

$$x_0 = a - \frac{f(a)}{f(b) - f(a)}(b - a) \qquad (2)$$

Затем вычисляем значение функции $f(x)$ в точке x_0 - $f(x_0)$ и определяем точку $A_1[x_0, f(x_0)]$. Через точки $A_1[x_0, f(x_0)]$ и $B[b, f(b)]$ проводим новую хорду $A_1 - B$.

Получаем второе приближение корня

$$x_1 = x_0 - \frac{f(x_0)}{f(b) - f(x_0)}(b - x_0) \qquad (3)$$

Далее повторяется процесс построения новых хорд и определения последующих приближений значений корня. Обобщая (3) можно записать

$$x_{i+1} = x_i - \frac{f(x_i)}{f(b) - f(x_i)}(b - x_i) \qquad (4)$$

Таким образом, строится монотонно возрастающая последовательность

$$a < x_0 < x_1 < \ldots < x_n < \xi < b$$

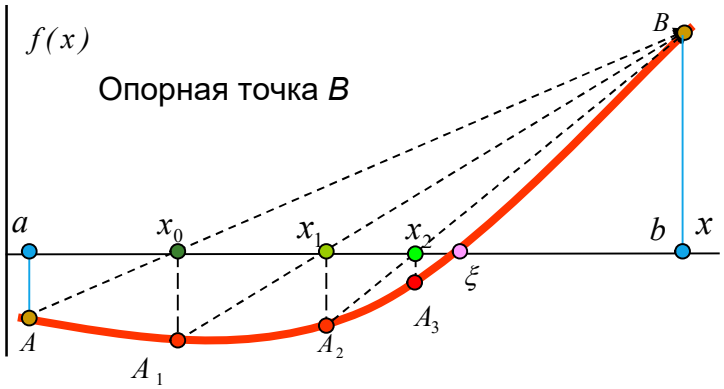


Рис.22. Поиск корня уравнения $f(x) = 0$. Алгоритм: метод хорд.
Опорная точка B.

Приближенное значение корня уравнения $f(x) = 0$ определяем как

$$\overline{\xi} = \lim_{n \to \infty} x_n \qquad (5)$$

Алгоритм решения:

Полагая, что $b > a$ и ζ и δ - малые значения.

Шаг 1: Вычисляем значения функции $f(x)$ на концах интервала - $f(a)$, $f(b)$. Проверяем условие наличия корня на интервале $[a, b]$ - если корень существует, то переходим к шагу 2, иначе – выход.

Шаг 2: Вычисляем значения второй производной $f''(x)$ - $f''(a)$, $f''(b)$. Выбираем опорную точку.

Шаг 3: Используя уравнение хорды (3.2) и формулу (3.5) определяем приближённое значение корня x_0 и запоминаем его в ячейке x_{cm}.

Шаг 4: Используя уравнение хорды (1) и формулу (4) определяем приближённое значение корня x_0.

Шаг 5: Вычисляем $abs(x_{cm} - x_0)$.

Если $abs(x_{cm} - x_0) < \delta$ и $f(x_0) < \zeta$, то выход, иначе уменьшаем значение δ и переходим к **шагу 3.**

Шаг 6: Выход.

Для тестирования программы найдите корень уравнения $y = (x-1)^2 - 2$ на интервале [1, 4]. Результат - $x_0 = 2.414$.

Варианты функций

Аргумент *x* определен в диапазоне *[0÷1]*

1. $f(x) = -cos(\pi x/2) + 2x$
2. $f(x) = sin(\pi x/2) - (1-x)$
3. $f(x) = sh(x) - cos(\pi x)$
4. $f(x) = 1 - x - tg(\pi x/4)$
5. $f(x) = e^{-x} - sin(\pi x/2)$
6. $f(x) = (1-x)^2 - 2x$
7. $f(x) = arcsin(x) - (1-x)^2$
8. $f(x) = ln((1+x) - (1-x)$

Лабораторная работа № 5.

Решение СЛАУ

Способы решения систем линейных алгебраических уравнений можно разделить на две группы:

1. Точные методы – представляют собой конечные алгоритмы для вычисления корней системы (правило Крамера, метод Гаусса, метод главных элементов, метод квадратных корней и др.)
2. Итерационные методы – построенные на сходящихся бесконечных процессах, обеспечивающих заданную точность вычисления корней системы (метод итераций, метод Зейделя, метод релаксации и др.)

Полученные результаты в обоих случаях являются приближёнными из-за округления численных результатов и погрешностей методов.

Рассмотрим систему n линейных уравнений с n неизвестными

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ &a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ &..... \\ &a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{aligned} \qquad (24)$$

Введем обозначения:

- матрица коэффициентов $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$
- вектор-столбец правых частей системы и вектор-столбец неизвестных

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Система (24) в матричном виде запишется как

$$Ax = b \quad (25)$$

Необходимым и достаточным условием существования единственного решения системы является условие неравенства нулю детерминанта матрицы A.

$$D = Det\ A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \neq 0 \qquad (26)$$

В противном случае система называется *вырожденной* и может иметь либо множество решений, либо не иметь ни одного решения.
При $D \approx 0$ система является *плохо обусловленной*. Это условие является необходимым, но недостаточным для определения плохо обусловленной системы.

Прямые методы – расчёт ведётся по конечным соотношениям (формулам) для определения неизвестных. Число операций заранее известно. Достоинством прямых методов является их относительная простота и универсальность. Недостатком является необходимость хранить в памяти компьютера сразу всю матрицу, что при большой размерности системы приводит к неэффективности вычислений, особенно, если матрица содержит много нулевых элементов – в этом случае производятся ненужные операции с нулями. Другим недостатком является накопление погрешности результатов, т.к. последующие операции используют данные, полученные на предшествующих операциях (это особенно опасно для плохо обусловленных матриц).

Итерационные методы – методы последовательных приближений. Для работы процедуры необходимо задать начальное приближение, т.е. задать произвольное значение вектору неизвестных. Это стартовое значение позволяет провести один цикл вычислений (итерацию). Затем вычисления повторяются до достижения нужной точности результатов. Алгоритмы итерационных методов, как правило, более сложные, чем алгоритмы прямых методов. Их достоинством является отсутствие необходимости хранить в памяти машины целиком все матрицы, достаточно хранить несколько векторов с n компонентами. Погрешности вычислений не накапливаются, т.к. на текущей итерации используются результаты только предшествующей итерации, а не всех предшествующих вычислений. Более того, случайный сбой вычислений означает, что расчёт начинается с нового произвольного начального приближения.

Прямые методы решения СЛАУ.
Метод Гаусса

Этот метод основан на приведении матрицы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. С помощью первого уравнения исключается x_1 из всех последующих уравнений. Преобразованная система содержит $(n-1)$ уравнение относительно $x_2, x_3 \ldots x_n$ неизвестных. Процесс повторяется - с помощью первого уравнения преобразованной системы исключается x_2 из всех последующих уравнений этой системы. Процесс продолжается до получения одного уравнения относительно неизвестного x_n. Таким образом, из исходной системы и преобразованных систем уравнений формируется матрица треугольного вида. Этот процесс называется *прямым ходом метода Гаусса.*

Например, для системы из трёх уравнений

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \qquad (27)$$

Формирование матриц в процессе прямого хода можно отобразить следующим образом
Разделив первое уравнение системы (27) на a_{11}, получим уравнение

$$x_1 + \frac{a_{12}}{a_{11}}x_2 + \frac{a_{13}}{a_{11}}x_3 = \frac{b_1}{a_{11}} \qquad (28)$$

которое умножим на a_{21}

$$a_{21}x_1 + a_{21}\frac{a_{12}}{a_{11}}x_2 + a_{21}\frac{a_{13}}{a_{11}}x_3 = a_{21}\frac{b_1}{a_{11}} \qquad (29)$$

Вычтем из второго уравнения (27) уравнение (29), получим

$$\left(a_{22} - a_{21}\frac{a_{12}}{a_{11}}\right)x_2 + \left(a_{23} - a_{21}\frac{a_{13}}{a_{11}}\right)x_3 = b_2 - a_{21}\frac{b_1}{a_{11}} \qquad (30)$$

Умножим уравнение (28) на a_{31} и вычтем это уравнение из третьего уравнения (27)

$$\left(a_{32} - a_{31}\frac{a_{12}}{a_{11}}\right)x_2 + \left(a_{33} - a_{31}\frac{a_{13}}{a_{11}}\right)x_3 = b_3 - a_{31}\frac{b_1}{a_{11}} \quad (31)$$

В результате этих действий пришли к системе из двух уравнений, не содержащих переменную x_1

$$\begin{aligned} a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 &= b_2^{(1)} \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 &= b_3^{(1)} \end{aligned} \quad (32)$$

где $a_{22}^{(1)} = \left(a_{22} - a_{21}\frac{a_{12}}{a_{11}}\right)$, $a_{23}^{(1)} = \left(a_{23} - a_{21}\frac{a_{13}}{a_{11}}\right)$, (33)

$$a_{32}^{(1)} = \left(a_{32} - a_{31}\frac{a_{12}}{a_{11}}\right), \quad a_{33}^{(1)} = \left(a_{33} - a_{31}\frac{a_{13}}{a_{11}}\right),$$

Проведём подобное преобразование для системы из двух уравнений (32) и получим одно уравнение с одним неизвестным x_3.

$$a_{33}^{(2)}x_3 = b_3^{(2)} \quad (34)$$

На этом шаге процедура прямого хода заканчивается, треугольная матрица сформирована.

$$\begin{matrix} a_{11} & a_{12} & a_{13} & b_1 \\ & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ & & a_{33}^{(2)} & b_3^{(2)} \end{matrix} \quad (35)$$

В процессе формирования треугольной матрицы (исключение неизвестных) производится деления на коэффициенты $a_{11}, a_{22}^{(1)}$ и т.д. Поэтому они должны быть отличными от нуля. В случае необходимости нужно переставить уравнения системы местами, чтобы выполнить это требование. На этом принципе исключения переменных основаны несколько алгоритмов.

Приведённый ниже алгоритм не требует хранения промежуточных данных (пересчёта коэффициентов промежуточных этапов).В общем виде вычисление коэффициентов можно записать как

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}a_{kj}^{(k-1)} \quad (36)$$

$$b_i^{(k)} = b_i^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}b_k^{(k-1)}$$

На следующем этапе производится вычисление неизвестных, начиная с x_n

В результате обратного хода

$$x_k = \frac{1}{a_{kk}^{(k-1)}}\left(b_k^{(k-1)} - \sum_{j=k+1}^{n} a_{kj}^{(k-1)}x_j\right) \quad (37)$$

Последовательно вычисляются все неизвестные. Этот алгоритм часто называют схемой единственного деления, а элемент $a_{kk}^{(k-1)}$ - ведущим элементом.

Задание .

Используя зависимости (36) и (37) найти решение тестовой системы уравнений

$$\begin{aligned} 1.000x_1 + 0.001x_2 + 0.001x_3 + 0.001x_4 + 0.001x_5 &= 1.00 \\ 0.001x_1 + 2.000x_2 + 0.001x_3 + 0.001x_4 + 0.001x_5 &= 2.00 \\ 0.001x_1 + 0.001x_2 + 3.000x_3 + 0.001x_4 + 0.001x_5 &= 3.00 \\ 0.001x_1 + 0.001x_2 + 0.001x_3 + 4.000x_4 + 0.001x_5 &= 4.00 \\ 0.001x_1 + 0.001x_2 + 0.001x_3 + 0.001x_4 + 5.000x_5 &= 5.00 \end{aligned}$$

Алгоритм решения. (псевдокод).

Для $k = 1, 2, \dots, n-1$ (*цикл последовательного исключения переменных*)

  Для $i = k+1, \dots, n$

    $t_{ik} := a_{ik}/a_{kk}$

    $b_i := b_i - t_{ik} b_k$

    Для $j = k+1, \dots, n$

      $a_{ij} := a_{ij} - t_{ik} a_{kj}$

$x_n = b_n/a_{nn}$

Для $k = n-1, \dots, 2, 1$

$$x_k = \left(b_k - \sum_{j=k+1}^{n} a_{kj} x_j \right) / a_{kk}$$

Студентам: Постарайтесь сначала написать собственный алгоритм.
Если это вызывает сложности, воспользуйтесь приведённым кодом.

Код программы

```
      real a(5,5),t(5,5),b(5),x(5)
      write(*,*) 'vvesti n'
      read(*,*) n
      do i=1,n
        do j=1,n
          if (i.eq.j)  then
            a(i,j)=i
          else
            a(i,j)=0.1
          endif
        enddo
        b(i)=i
      enddo
      write(*,*) 'исходная матрица a(n*n)  и вектор b(n)'
      write(*,10)  ((a(i,j),j=1,5),b(i),i=1,5)
      do k=1,n-1
        do i=k+1,n
          t(i,k)=a(i,k)/a(k,k)
          b(i)=b(i)-t(i,k)*b(k)
          do j=k+1,n
            a(i,j)=a(i,j)-t(i,k)*a(k,j)
          enddo
        enddo
c     промежуточная печать на k-ой итерации
      write(*,*) ' матрица a(n*n)  и вектор b(n)  k=',k
      write(*,10)  ((a(i,j),j=1,5),b(i),i=1,5)
      enddo
c     прямой ход
       write(*,*) 'результирующая матрица и вектор'
      write(*,10)  ((a(i,j),j=1,5),b(i),i=1,5)
c     обратный ход
      x(n)=b(n)/a(n,n)
      sum=0
      do k=n-1,1,-1
        do j=k+1,n
          sum=sum+a(k,j)*x(j)
        enddo
        x(k)=(b(k)-sum)/a(k,k)
      enddo
```

```
      write(*,*) 'результат'
      write(*,10)  (x(k),k=1,n)
10     format (5e12.5,4x,e12.5)
      end
```

Результаты расчёта

```
Ввести n
5
        'исходная матрица a(n*n)  и вектор b(n)'
0.10000E+01  0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00     0.10000E+01
 0.10000E+00 0.20000E+01 0.10000E+00 0.10000E+00 0.10000E+00     0.20000E+01
 0.10000E+00 0.10000E+00 0.30000E+01 0.10000E+00 0.10000E+00     0.30000E+01
 0.10000E+00 0.10000E+00 0.10000E+00 0.40000E+01 0.10000E+00     0.40000E+01
 0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00 0.50000E+01     0.50000E+01

матрица a(n*n)  и вектор b(n)  k=1
 0.10000E+01 0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00   0.10000E+01
 0.10000E+00 0.19900E+01 0.90000E-01  0.90000E-01 0.90000E-01    0.19000E+01
 0.10000E+00 0.90000E-01 0.29900E+01  0.90000E-01 0.90000E-01    0.29000E+01
 0.10000E+00 0.90000E-01 0.90000E-01  0.39900E+01 0.90000E-01    0.39000E+01
 0.10000E+00 0.90000E-01 0.90000E-01  0.90000E-01 0.49900E+01    0.49000E+01

матрица a(n*n)  и вектор b(n)  k=2
 0.10000E+01 0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00    0.10000E+01
 0.10000E+00 0.19900E+01 0.90000E-01  0.90000E-01  0.90000E-01    0.19000E+01
 0.10000E+00 0.90000E-01 0.29859E+01  0.85930E-01  0.85930E-01    0.28141E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.39859E+01  0.85930E-01    0.38141E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.85930E-01  0.49859E+01    0.48141E+01

матрица a(n*n)  и вектор b(n)  k=3
 0.10000E+01 0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00    0.10000E+01
 0.10000E+00 0.19900E+01 0.90000E-01  0.90000E-01  0.90000E-01    0.19000E+01
 0.10000E+00 0.90000E-01 0.29859E+01  0.85930E-01  0.85930E-01    0.28141E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.39835E+01  0.83457E-01    0.37331E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.83457E-01  0.49835E+01    0.47331E+01

матрица a(n*n)  и вектор b(n)  k=4
 0.10000E+01 0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00    0.10000E+01
 0.10000E+00 0.19900E+01 0.90000E-01  0.90000E-01 0.90000E-01    0.19000E+01
 0.10000E+00 0.90000E-01 0.29859E+01  0.85930E-01 0.85930E-01    0.28141E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.39835E+01 0.83457E-01    0.37331E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.83457E-01 0.49817E+01    0.46549E+01

результирующая матрица и вектор
 0.10000E+01 0.10000E+00 0.10000E+00 0.10000E+00 0.10000E+00    0.10000E+01
 0.10000E+00 0.19900E+01 0.90000E-01  0.90000E-01 0.90000E-01    0.19000E+01
 0.10000E+00 0.90000E-01 0.29859E+01  0.85930E-01 0.85930E-01    0.28141E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.39835E+01 0.83457E-01    0.37331E+01
 0.10000E+00 0.90000E-01 0.85930E-01  0.83457E-01 0.49817E+01    0.46549E+01
результат
 0.17575E+00   0.71283E+00   0.86303E+00   0.91757E+00   0.93439E+00
```

Комментарии к результатам расчёта по данному алгоритму

- Алгоритм не требует дополнительной памяти, т.к. не сохраняет матрицы промежуточных преобразований матриц и векторов (используется область исходной матрицы и вектора).
- На очередной итерации преобразуются элементы матриц, выделенные зеленым цветом. Строки, полученные на предшествующей итерации, не изменяют своих значений.
- В результирующей матрице для вычисления значений x_i используются только элементы верхней диагональной части (чёрный цвет). Элементы нижней части (красный цвет) – это «отходы производства».

Итерационные методы.
Метод простой итерации.

При большом числе уравнений метод Гаусса может оказаться достаточно сложным. В этом случае более удобно пользоваться не точными, а приближёнными численными методами. Систему уравнений $Ax = b$ представим в развёрнутом виде

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ &a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ &\quad ..... \\ &a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{aligned} \qquad A = \begin{bmatrix} a_{11}a_{12}\cdots a_{1n} \\ a_{21}a_{22}\cdots a_{2n} \\ \cdots\cdots\cdots\cdots \\ a_{n1}a_{nn}\cdots a_{nn} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (38)$$

Полагая $a_{ii} \neq 0 \qquad (i = 1,2,...,n)$ решим первое уравнение относительно x_1, второе — относительно x_2 и т. д. Получаем эквивалентную систему

$$\begin{aligned} x_1 &= \beta_1 + \alpha_{12}x_2 + \alpha_{13}x_3 ... + \alpha_{1n}x_n \\ x_2 &= \beta_2 + \alpha_{21}x_1 + \alpha_{23}x_3 ... + \alpha_{2n}x_n \\ &\quad ..... \\ x_n &= \beta_n + \alpha_{n1}x_1 + \alpha_{n2}x_2 ... + \alpha_{n,n-1}x_{n-1} \end{aligned} \qquad (39)$$

где $\beta_i = \dfrac{b_i}{a_{ii}}, \quad \alpha_{ij} = -\dfrac{a_{ij}}{a_{ii}} \qquad i \neq j \qquad (40)$

и

$\alpha_{ij} = 0$ при $i = j, \quad (i, j = 1,2,..n)$

Введя матрицы

$$\alpha = \begin{bmatrix} \alpha_{11}\alpha_{12}\cdots\alpha_{1n} \\ \alpha_{21}\alpha_{22}\cdots\alpha_{2n} \\ \cdots\cdots\cdots\cdots \\ \alpha_{n1}\alpha_{nn}\cdots\alpha_{nn} \end{bmatrix} \quad \text{и} \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \qquad (41)$$

и систему уравнений можно записать в матричном виде

$$x^{(0)} = \beta + \alpha x \qquad (42)$$

Эта система решается методом последовательных приближений. За нулевое приближение принимается, например, столбец свободных членов $x^{(0)} = \beta$.

Затем строится первое приближение

$$x^{(1)} = \beta + \alpha x^{(0)} \qquad (43)$$

потом — второе приближение

$$x^{(2)} = \beta + \alpha x^{(1)} \qquad (44)$$

и т. д. Общий алгоритм решения

$$x^{(k+1)} = \beta + \alpha x^{(k)} \qquad (45)$$

В результате получаем последовательность $x^{(0)}, x^{(1)}, ..., x^{(k)} ...$, предел которой (если он существует)

$$x = \lim_{k \to \infty} x^{(k)}$$

является решением системы $\lim_{k \to \infty} x^{(k+1)} = \beta + \lim_{k \to \infty} x^{(k)}$

Формулы приближений можно представить в виде

$$x_i^{(0)} = \beta_i$$

$$x_i^{(k+1)} = \beta_i + \sum_{j=1}^{n} \alpha_{ij} x_j^{(k)} \qquad (46)$$

$$\left(\alpha_{ii} = 0; i = 1,..., n; k = 0,1,2,...,\right)$$

Задание .

Систему уравнений из предшествующего задания решить итерационным методом и сравнить результаты.

Алгоритм.

- Сформировать матрицы $\boldsymbol{A}$ и b.
- Сформировать матрицы $\boldsymbol{\alpha}$ и β.
- За нулевое приближение принять столбец свободных членов $x^{(0)} = \beta$.
- В цикле организовать процесс (46). Завершение процесса – значения вектора x на двух соседних итерациях отличаются на малую величину ε (точность вычислений)

Лабораторная работа № 6.

Нахождения минимума функции.

Задание 1.

Алгоритм реализуется в задачах оптимизации тремя способами – методом дихотомии, методом деления отрезка пополам и золотым сечением. Различие заключается в быстродействии, которое в данном случае определяется количеством обращений к вычислению минимизируемой функции F_{min}.

Сравнить методы по быстродействию – количеству обращений к вычислению функции. Произвести вычисления при различных значениях точности вычисления. В последующих заданиях значения варьируемых параметров (границы интервала поиска минимума функции и точность вычисления) задавать с клавиатуры, используя операторы ввода данных.

Дихотомия

Наиболее простой алгоритм, в котором на каждой итерации интервал поиска минимума функции *сокращается ровно вдвое*. Вычисление значения минимизируемой функции на каждой итерации производится дважды. Суть алгоритма заключается в следующем – на текущей итерации определяется x_{cp} середина отрезка *[a,b]* и вычисляются значения двух близко лежащих точек $x_1 = x_{cp} - \varepsilon$ и $x_2 = x_{cp} + \varepsilon$, где ε наперёд заданное малое значение аргумента функции (Рис.7). Далее сравниваются значения функции в точках x_1 и x_2.

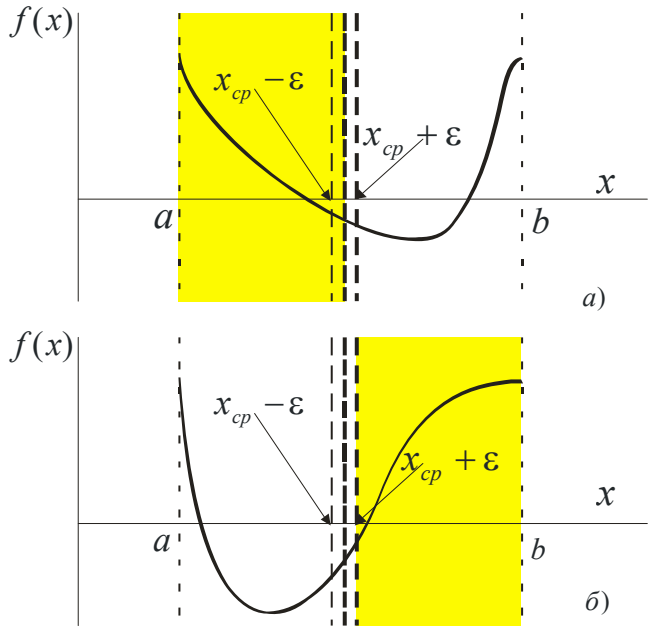


Рис.7. Сокращение интервала поиска методом дихотомии

Пусть функция $f(x)$ задана на отрезке $[a,b]$ и имеет только один минимум. Параметры ε, δ – малые значения.
Шаг1: Определить середину интервала $x_{cp}=(a+b)/2$, выделить две точки на оси аргументов $x_1 = x_{cp} - \varepsilon$ и $x_2 = x_{cp} + \varepsilon$.
Шаг2: Вычислить значения функции $f(x_1)$ и $f(x_2)$.
Шаг3: Сравнить значения функции $f(x_1)$ и $f(x_2)$.
Если $f(x_1) < f(x_2)$, то присваиваем $b = x_{cp}$, иначе $a = x_{cp}$.
Шаг4: Признак завершения счета. Если $abs(b-a) < \delta$, то перейти к шагу 5, иначе перейти к шагу 1.
Шаг5: Вычислить $x_{min}=(a+b)/2$ и $f(x_{min})$

Для тестирования программы найдите минимум параболы $y = (x-1)^2 + 3$ на интервале [-1, 3]. Результат - $x_{min} = 1$.

Варианты функций

1. $f(x) = -x^3 + 3x^2 + 9x + 10$ $\quad x \in [-3,3]$
2. $f(x) = (2x+1)^2 + (x-4)$ $\quad x \in [-2,4]$
3. $f(x) = (10-x)^2$ $\quad x \in [-10,20]$
4. $f(x) = x^3 - 12x + 3$ $\quad x \in [-2,5]$
5. $f(x) = (10x^3 + 3x^2 + x + 5)^2$ $\quad x \in [-2,2]$
6. $f(x) = 3x^4 + (x-1)^2$ $\quad x \in [-2,2]$
7. $f(x) = 4x * sin(x)$ $\quad x \in [-\pi/4, \pi/2]$
8. $f(x) = 2(x-3)^2 + e^{x/2}$ $\quad x \in [0,4]$

Деление отрезка пополам.

Метод, в котором на каждой итерации интервал поиска минимума функции *сокращается вдвое*. Он основан на выборе трёх пробных точек, расположенных равномерно на интервале поиска минимума целевой функции. На текущей итерации определяется x_{cp} середина отрезка $[a,b]$ и значения двух симметрично расположенных точек $x_1 = x_{cp} - abs(b-a)/4$ и $x_2 = x_{cp} + abs(b-a)/4$. Далее сравниваются значения функции в точках x_{cp}, x_1, и x_2. Можно определить, какую часть интервала поиска можно исключить на следующей итерации.

На Рис.8. приведены возможные положения расположения минимума целевой функции, которые могут возникнуть в процессе сокращения интервала поиска. Заштрихованные области исключаются из дальнейшего рассмотрения.
При $f(x_1) > f(x_{cp})$ – правую половину (Рис.8а.), при $f(x_1) < f(x_{cp})$ - левую половину (Рис.8б.) и при $f(x_2) > f(x_{cp})$ - левую и правую четверти (Рис.8в.). Соответствующим образом переносятся границы интервала

поиска $a \rightarrow x_{cp}$ (Рис.8а.) или $b \rightarrow x_{cp}$ (Рис.8б.) или $a \rightarrow x_1$ и $b \rightarrow x_2$ (Рис.8в.) и формируется очередной интервал $[a,b]$.

Поиск завершается, когда интервал поиска $[a,b]$ сокращается до заданной малой величины δ.

На первом шаге поиска производится три обращения к вычислению целевой функции в точках x_{cp}, x_1, и x_2 на последующих шагах поиска – два обращения (в точках x_1, и x_2) Общее количество обращений к вычислению целевой функции определяется как $(2*N+1)$, где N – число шагов поиска.

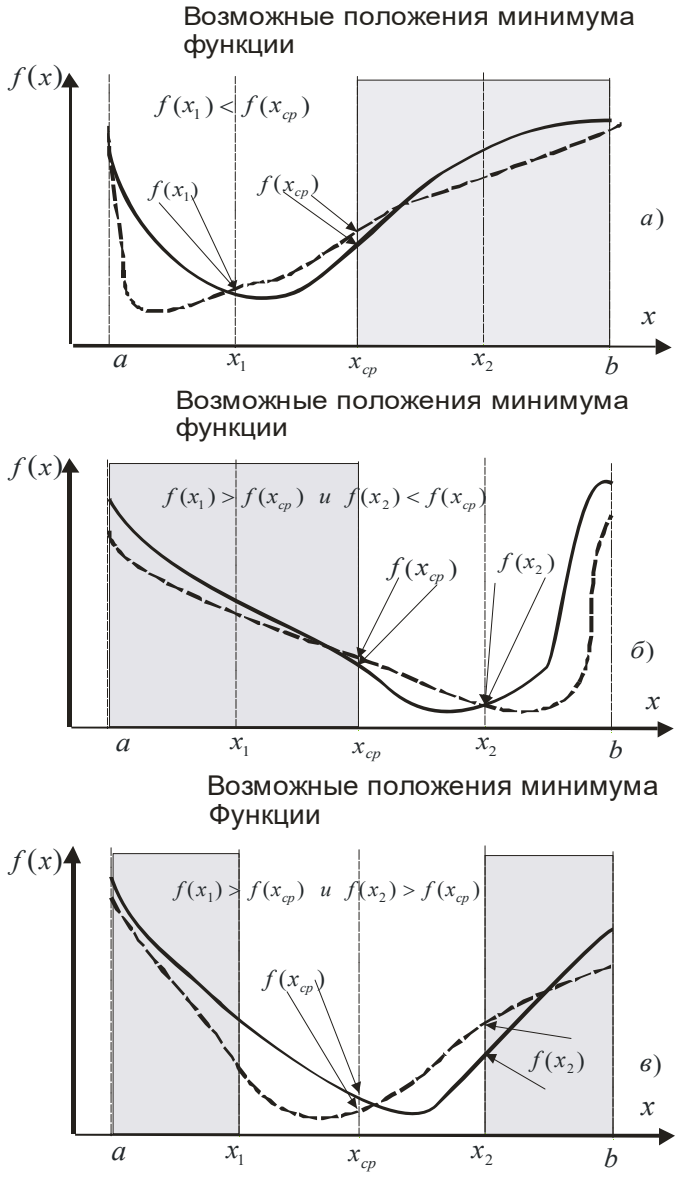


Рис.8. Возможные положения минимумов целевой функции.

Алгоритм поиска.

Пусть функция $f(x)$ задана на отрезке $[a,b]$ имеет только один минимум. δ – малое значения.

Шаг1: Определить середину интервала $x_{cp}=(a+b)/2$. Вычислить в этой точке значение функции $fx_{cp}= f(x_{cp})$.

Шаг2: Определить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(b-a)/4$.

Шаг3: Вычислить значения функции $fx_1= f(x_1), fx_2 =f(x_2)$.

Шаг4: Сравнить значения функции fx_1 и fx_{cp}.

Если $fx_1 < fx_{cp}$, то присвоить $b= x_{cp}$, $x_{cp}=x_1$, $fx_{cp}= fx_1$, вычислить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(a)/4$,

перейти к шагу 6. Иначе перейти к шагу 5.

Шаг5: Сравнить значения функции fx_2 и fx_{cp}.

Если $fx_2 < fx_{cp}$, то присвоить $a= x_{cp}$, $x_{cp}=x_2$, $fx_{cp}= fx_2$,

вычислить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(b-a)/4$, перейти к шагу 6. Иначе присвоить $a= x_1$, $b= x_2$, вычислить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(b-a)/4$

Шаг6: Завершения счета. Если $abs(b-a)< \delta$, то перейти к шагу 7, иначе перейти к шагу 3.

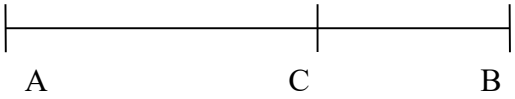
Шаг7: Вычислить $x_{min}=(a+b)/2$ и $f(x_{min})$

Золотое сечение.

Из рассмотрения выше приведенных методов можно сделать выводы позволяющие повысить эффективность поиска:

1. Если количество пробных точек равно двум, то их следует размещать на одинаковых расстояниях от середины интервала.
2. В соответствии с общей минимаксной стратегией пробные точки должны размещаться по симметричной схеме таким образом, чтобы отношение длины исключаемого подынтервала к величине интервала поиска оставалось постоянным.
3. На каждой итерации поиска должна вычисляться только одна пробная точка и значение целевой функции в этой точке.

Геометрическая интерпретация принципа метода золотого сечения заключается в следующем. Дана прямая АСВ



Отношение отрезков прямой определяется соотношением AC/AB=CB/AC, т.е. отношение большего отрезка к целому равно отношению меньшего отрезка к большему отрезку. Если принять целый отрезок AB=1, то $AC = (-1+\sqrt{5})/2$. Приближённо AC≈0.62, а CB≈0.38.

Количество обращений к вычислению целевой функции определяется как (N+1), где N – число шагов поиска.

Пусть функция *f(x)* задана на отрезке [*a,b*] и имеет только один минимум. Параметр δ – малое значение.

Шаг1: Задать коэффициенты *λ2*=0.62, *λ1*=0.38, интервал *[a,b]* и длину отрезка *l=abs(b-a)*.

Шаг2: Выделить две точки *v=a+l* λ1* и *w=a+l* λ2* .

Шаг3: Если abs(*b-a)* < *δ* , то перейти к шагу 7,
иначе перейти к шагу 4.

Шаг4: Вычислить в этих точках значения функции *fv=f(v)* и *fw=f(w).*

Шаг5: Сравнить значения функции *fw* и *fv.*

Если *fw* < *fv*, то присвоить *a=v, v=w, fv =fw* и вычислить *w=a+abs(b-a)* λ2*, *f(w).* Перейти к шагу 3.
иначе перейти к шагу 6.

Шаг6: Присвоить *b=w, w=v, fw =fv* и вычислить *v=a+abs(b-a)* λ1*, *f(v).* Перейти к шагу 3.

Шаг7: Вычислить $x_{min}=(a+b)/2$ и $f(x_{min})$.

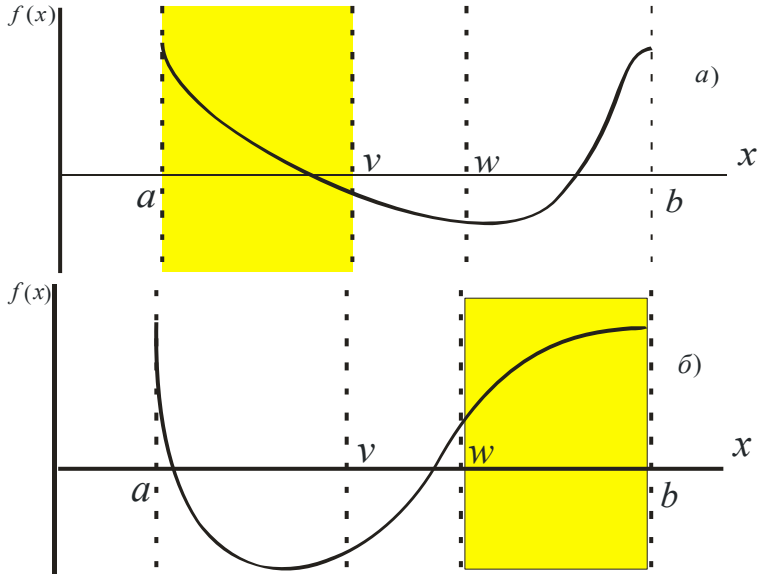


Рис.9. Возможные положения минимума целевой функции.

Пусть длина исходного интервала поиска L_1, а конечного интервала - L_n. Эффективность метода можно оценить по отношению L_n/L_1. В методах дихотомии и деления отрезка пополам длина конечного интервала составляет $L_1(0{,}5)^{N/2}$, а в методе золотого сечения - $L_1(0{,}618)^{N-1}$. При заданной точности вы-

числений ε количество обращений к вычислению целевой функции в двух первых методах составляет $N = 2\ln(\varepsilon)/\ln(0{,}5)$, а для метода золотого сечения $N = 1 + \ln(\varepsilon)/\ln(0{,}618)$.

Задание.

- Реализовать приведённые алгоритмы и проверить для тестовой функции.
- Провести расчёт для одного из приведённых вариантов функций.
- Сравнить приведённые методы поиска минимума функции одной переменной по эффективности - количеству обращений к вычислению самой функции при одинаковой точности расчётов.