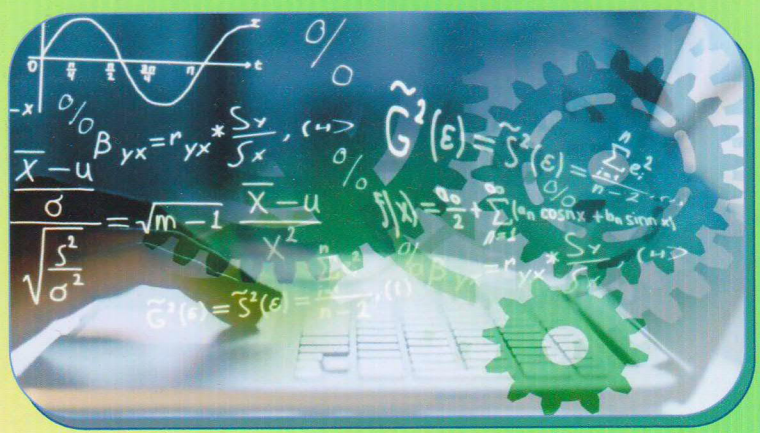
М.Х.Аламинов, Д.П.Бектурсынова,
Э.П.Уразымбетова

ЧИСЛЕННЫЕ МЕТОДЫ

Учебно-методическое пособие



Нукус -2020

МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ ИМЕНИ АЖИНИЯЗА

М.Х. Аламинов, Д.П. Бектурсынова, Э.П. Уразымбетова

ЧИСЛЕННЫЕ МЕТОДЫ

Учебно-методическое пособие

Нукус - 2020

Аламинов М.Х., Бектурсынова Д.П., Уразымбетова Э.П. Численные методы: Учебно-методическое пособие. – Нукус: НГПИ, 2020 г. – 64 с.

В данном учебно-методическом пособии приведены основные сведения по численным методам. Подробно рассмотрены методы решения нелинейных уравнений и систем уравнений. Приведены методика выполнения лабораторных работ по каждому методу, а также соответствующие программы на языке Pascal.

Пособие предназначено для студентов направления образования «Математика и информатика», «Прикладная математика», для специалистов в области вычислительной математики и математического моделирования.

ОТВЕТСТВЕННЫЙ РЕДАКТОР:

А.Абдуллаев – кандидат экономических наук, доцент кафедры «Методика преподавания информатики»

РЕЦЕНЗЕНТЫ:

А.Камалов – доктор физико-математических наук, заведующий кафедрой «Методика преподавания физики и астрономии» Нукусского государственного педагогического института имени Ажинияза

Д.Утебаев – доктор физико-математических наук, заведующий кафедрой «Прикладная математика» Каракалпакского государственного университета имени Бердаха

Учебно-методическое пособие рекомендовано к печати УНМС НГПИ им. Ажинияза от 7 июля 2020 г., протокол № 7.

ПРЕДИСЛОВИЕ

Во многих областях физики, механики, машиностроения численные методы широко используются при разработке и в исследовании математических моделей различных процессов. Первые две главы посвящены численным методам решения нелинейных уравнений и их систем. Представлены вычислительные алгоритмы метода половинного деления отрезка, метод хорд, метод Ньютона и метод простой итерации, приведены соответствующие примеры. Решение систем нелинейных уравнений является более сложной задачей чем решение одного нелинейного уравнения. Поэтому, для решения систем нелинейных уравнений применяется только итерационные методы. Из них часто используемым на практике являются метод простых итераций, метод Ньютона и его различные модификации.

В данном пособии предложены вычислительные алгоритмы выше изложенных методов.

В третьей главе изложены итерационные методы решения систем линейных алгебраических уравнений (СЛАУ). Итерационные методы решения СЛАУ имеет несколько преимуществ. При реализации их на ЭВМ не требуется полностью хранить на памяти данные матриц системы. Простота вычислительных схем и однообразие производимых операций делают эти методы удобными при использовании вычислительной техники.

Также, данное пособие содержит графические иллюстрации, примеры и программы численного решения нелинейных уравнений, линейных и нелинейных систем уравнений. Предложены варианты заданий для лабораторных работ, которые могут быть использованы при проведении практических занятий.

ГЛАВА 1. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ С ОДНИМ НЕИЗВЕСТНЫМ.

Цель работы. Знакомство с некоторыми приближенными методами решения одного нелинейного уравнения и с их численной реализацией на ПК.

Предварительные замечания.

В вычислительной практике часто приходится находить корни нелинейных уравнений вида:

$$F(x)=0, \qquad \text{(I)}$$

где $F(x)$ – некоторая непрерывная функция.

Корнем уравнения называется такое число c, что $F(c)=0$. Корень будет простым, если $F'(c)\neq 0$, и кратным порядка m, если $F^{(k)}(c)=0,\ k=1,2,...,m-1.$ Графически простой корень соответствует точке пересечения графика функции $y=F(x)$ с осью Ox под ненулевым углом, а кратный корень соответствует точке пересечения под нулевым углом.

Нелинейные уравнения можно разделить на две группы – алгебраические и трансцендентные. Алгебраические уравнения содержат только алгебраические функции (целые, рациональные, иррациональные). Так, например, многочлен есть целая алгебраическая функция. Уравнения, которые содержат другие функции (тригонометрические, показательные, логарифмические и т.п.), являются трансцендентными.

Методы решения нелинейных уравнений делятся на точные и итерационные. Точные методы позволяют получить корни рассматриваемого уравнения в результате выполнения конечного числа арифметических действий. Другими словами, эти методы позволяют записать корни в виде некоторого конечного соотношения. Однако большинство нелинейных уравнений нельзя решать так просто. Для их решения используются итерационные (численные или приближенные) методы решения. При их использовании точные значения корней исходного уравнения получаются в результате выполнения бесконечного числа арифметических операций.

При отыскании приближенных значений корней нелинейного уравнения (I) приходится решать две задачи:

1) отделение корней, т.е. отыскание достаточно малых областей, в каждой из которых заключен один и только один корень уравнения;

2) вычисление корней с заданной точностью.

Для отделения корней уравнения (I) можно использовать следующий критерий: если на отрезке $[a,b]$ функция $F(x)$ непрерывна и монотонна, а её значения на концах отрезка имеют разные знаки, то на этом отрезке существует и притом только один корень данного уравнения. Достаточным признаком монотонности функции $F(x)$ на отрезке $[a,b]$ является сохранение знака производной функции. При отделении корней стараются определить отрезок $[a,b]$ как можно меньшей длины.

Определение корней уравнения (I) можно выполнить также графически. Найти корень уравнения (I) – это значит найти абсциссу точки пересечения графика функции $y=F(x)$ с прямой $y=0$ (осью абсцисс). Если построить график функции $y=F(x)$ затруднительно, то уравнение (I) следует представить в эквивалентном виде:

$$F_1(x)=F_2(x).$$

В этом случае строятся графики функций $F_1(x)$ и $F_2(x)$, а потом на оси Ox отмечаются отрезки, локализующие абсциссы точек пересечения этих графиков.

Приближенное значение корня (нулевое или начальное приближение) можно найти из физических соображений, или другими способами. Например, найти два значения x: a и b, в которых функция $F(x)$ будет принимать значения разных знаков, т.е. $F(a)\cdot F(b)<0$. В этом случае между a и b есть по крайней мере одно значение x, для которого $F(x)=0$. В качестве этого значения x приближенно можно взять, например, значения $x_0=x_*=\dfrac{a+b}{2}$.

Итерационные методы состоят в последовательном уточнении начального приближения x_0. Каждый такой шаг называется итерацией. В результате итераций находится последовательность приближенных значений корня $x_0,x_1,x_2,\ldots,x_n,\ldots$ Если при этом с увеличением n значения

x_n приближаются к точному решению заданного уравнения, то говорят, что данный итерационный процесс сходится.

Рассмотрим некоторые численные методы решения трансцендентных уравнений. Эти методы могут использоваться и при решении алгебраических уравнений.

1.1. Метод деления отрезка пополам (метод бисекций).

Метод бисекций является одним из самых простых методов решения нелинейных уравнений вида $F(x)=0$. Главным его достоинством является то, что он всегда сходится. Недостатком этого метода является то, что он не обобщается на системы нелинейных уравнений и не может использоваться для нахождения корней четной кратности.

Алгоритм рассматриваемого метода может быть следующим.

1. Найти начальный интервал неопределенности $[a_0, b_0]$ одним из методов отделения корней, задать малое положительное число ε. Положить $k=0$.

2. Найти середину текущего интервала неопределенности:

$$c_k = \frac{a_k + b_k}{2}.$$

3. Если $F(a_k)\cdot F(c_k)<0$, то положить $a_{k+1}=a_k, b_{k+1}=c_k$, а если $F(c_k)\cdot F(b_k)<0$, то принять $a_{k+1}=c_k, b_{k+1}=b_k$. В результате находится текущий интервал неопределенности $[a_{k+1}, b_{k+1}]$.

4. Если $b_{k+1}-a_{k+1}\le\varepsilon$, то процесс завершить: $x_*\in[a_{k+1}, b_{k+1}]$. Приближенное значение корня можно найти по формуле

$$x_* \cong \frac{a_{k+1}+b_{k+1}}{2}.$$

Если $b_{k+1}-a_{k+1}>\varepsilon$, положить $k=k+1$ и перейти к п.2.

Итерационный процесс продолжаем до тех пор, пока значение функции $F(x)$ после n-ой итерации не станет меньше по модулю, чем некоторое $\varepsilon>0$, т.е. пока $|F(x)|<\varepsilon$, где $\varepsilon-$очень маленькое положительное число (точность, с которой надо решить уравнение (I)). Можно закончить счет и тогда, когда длина очередного отрезка $[a,b]$ станет меньше ε.

Надо отметить, что метод бисекций имеет линейную, но безусловную сходимость, и его погрешность за каждую итерацию уменьшается в два раза: $|b_1 - a_1| = \frac{|b_0 - a_0|}{2}; |b_2 - a_2| = \frac{|b_1 - a_1|}{2} = \frac{|b_0 - a_0|}{2^2}, ...,$ $|b_k - a_k| = |b_0 - a_0| \cdot 2^{-k}$. Из последнего соотношения можно оценить число k для достижения заданной точности ε:

$$|b_0 - a_0| \cdot 2^{-k} \le \varepsilon; \quad k \ge \log_2 \frac{(b_0 - a_0)}{\varepsilon}.$$

Отсюда видно, что, например, для достижения точности $\varepsilon \approx 10^{-3}$ при $(b_0 - a_0) \cong 1$ необходимо выполнить примерно десять итераций.

К достоинствам метода следует отнести то, что он позволяет найти простой корень уравнения $x_* \in [a_0, b_0]$ для любых непрерывных функций $F(x)$ при любых значениях a_0, b_0, таких, что $F(a_0) \cdot F(b_0) < 0$. Недостатки метода – он не обобщается на системы нелинейных уравнений и не может использоваться для нахождения корней четной кратности.

Дадим геометрическую интерпретацию метода бисекций и приведем его блок-схему.

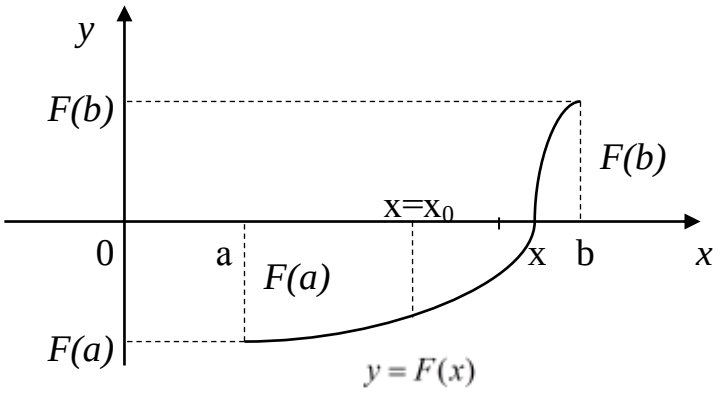
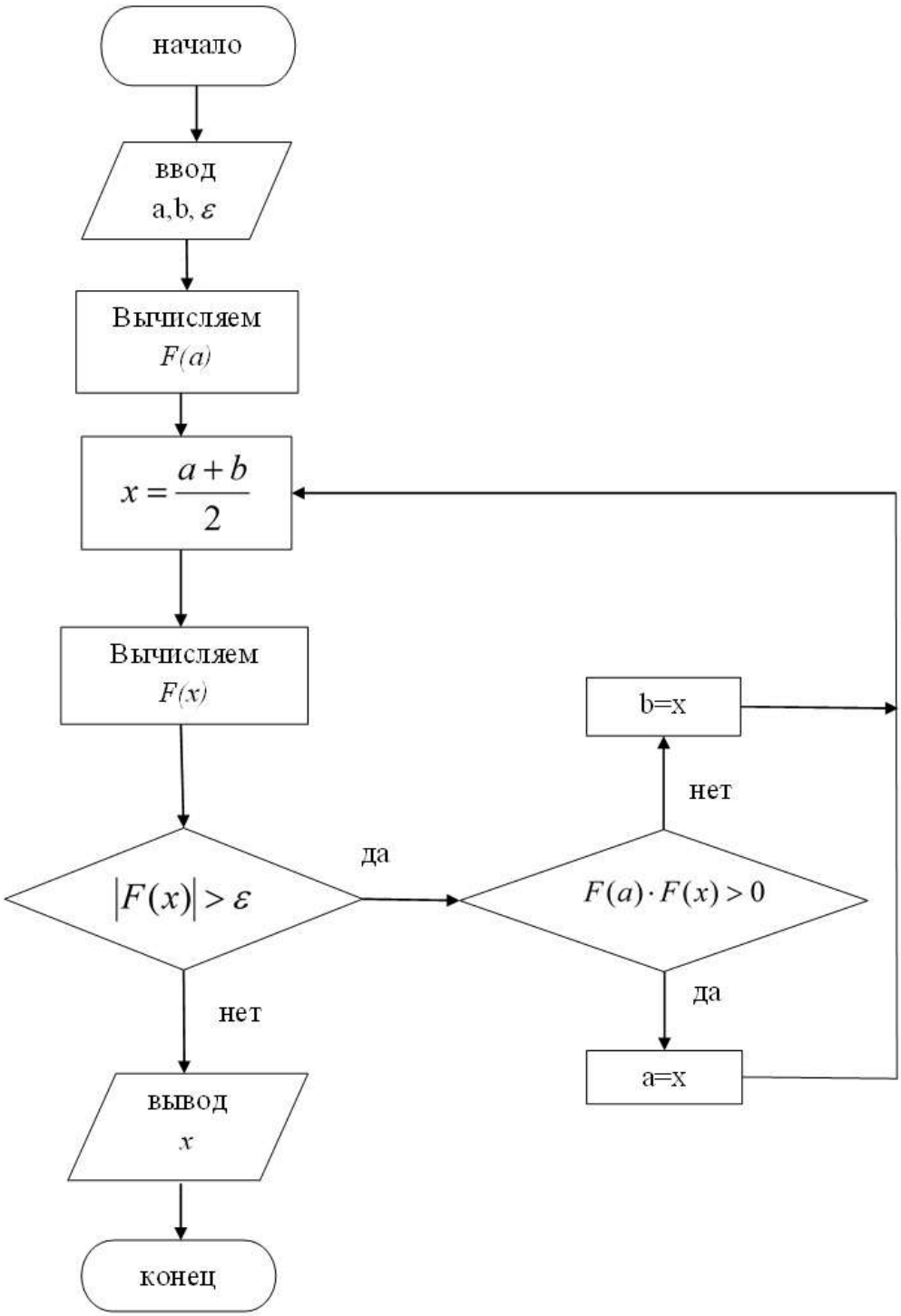


Рис. 1.

В данной блок-схеме сужение отрезка $[a, b]$ происходит путем замены границы a или b на текущее значение корня x. При этом значение $F(a)$ вычисляется один раз, т.к. нам нужен лишь знак функции $F(x)$ на левой границе, а он в процессе итераций не меняется.



Блок-схема метода бисекций

Пример. Найти корень уравнения $\sin 2x - \ln x = 0$ методом бисекций с точностью до 10^{-4}.

Решение. Для графического отделения корней перепишем заданное уравнение в виде $\sin 2x = \ln x$. Обозначив $y_1 = \sin 2x$, $y_2 = \ln x$, построим графики этих функции:

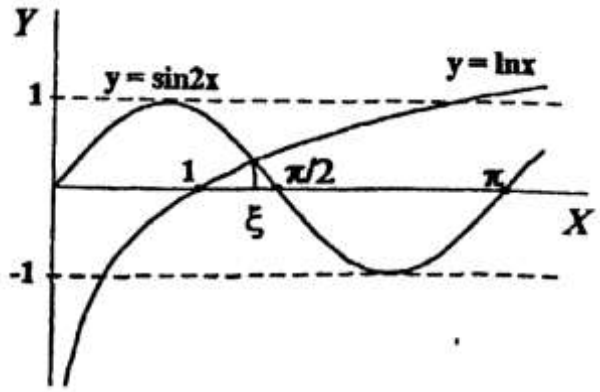


Рис. 2.

Из графика следует, что уравнение имеет корень, принадлежащий отрезку $[1;1,5]$. Вычислим для проверки с помощью калькулятора значения функции $\sin 2x - \ln x = 0$ на концах отрезка $[1;1,5]$:

$$F(1) = 0,909298; \quad F(1,5) = -0,264344.$$

Из графика видно, что на отрезке $[1;1,5]$ имеется единственный корень исходного уравнения. Рассмотренный приём позволяет сузить отрезок, полученный графически. Так, в нашем примере имеем: $F(1,3) = 0,253137 > 0,$ так что отрезком отделения корней можно считать $[1,3;1,5]$. При этом $F(1,3) \cdot F(1,5) < 0.$

Чтобы уточнить корень методом бисекций разделим отрезок $[a_0, b_0] = [1,3;1,5]$ пополам. Получим точку

$$x_0 = \frac{a_0 + b_0}{2} = \frac{1,3+1,5}{2} = 1,4\,.$$

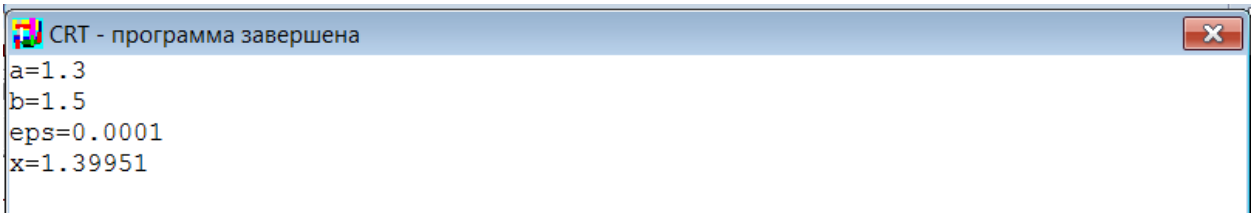
Вычислим значение функции в этой точке: $F(1,4) = -0,001484.$ Отсюда, $F(1,3) \cdot F(1,4) < 0.$ Тогда, имеем $a_1 = a_0, b_1 = c_0,$ т.е. следующий отрезок, на котором функция меняет знак, есть $[a_1, b_1] = [1,3;1,4]$. Делим отрезок пополам, при этом $c_1 = (a_1 + b_1)/2 = 1,35,$ и следующий отрезок на котором функция меняет знак, есть $[a_2, b_2] = [1,35;1,4]$. Делим отрезок $[a_2, b_2] = [1,35;1,4]$ пополам, при этом $c_2 = 1,375$ и получаем следующий отрезок $[a_3, b_3] = [1,375;1,4]$ и т.д.

Для отрезка $[a_{10}, b_{10}] = [1,399414;1,399609]$ выполнено условие $(b_n - a_n)/2 < \varepsilon$, поэтому можно принять $x_{10} = (a_{10} + b_{10})/2 = 1,399512$ за приближение к корню с заданной точностью ε.

Программа решения задачи на языке Pascal

```
program  MDOP;    { Метод деления отрезка пополам }
uses crt;
var a,b,eps,x,fa,fc,c:real;
function f(x:real):real;
begin
f:=sin(2*x)-ln(x)           { вид функции f(x) }
end;
begin clrscr;
write('a='); read(a);          {ввод границ отрезка}
write('b='); read(b);          {ввод границ отрезка}
write('eps=');    read(eps);       {ввод точности}
fa:=f(a);
while abs(b-a)>eps do
begin
c:=(a+b)/2;       {деления отрезка пополам}
fc:=f(c);
if fa*fc<=0       then     b:=c
else begin        a:=c;        fa:=fc   end;
end;
writeln('x=',c:5:5);       {окончательный результат}
end.
```

Результат работы программы



1.2. Метод хорд (метод секущих).

Пусть найден отрезок $[a,b]$, где уравнение $F(x)=0$ имеет корень. Для определенности будем считать, что $F(a)>0$, а $F(b)<0$. В данном методе процесс итераций состоит в том, что в качестве приближений к корню уравнения (I) принимаются значения $x_0, x_1, x_2, \ldots, x_n, \ldots$ точек пересечения хорды с осью абсцисс.

Сначала запишем уравнение хорды AB, как прямой, проходящей через две точки $A(a,F(a))$ и $B(b,F(b))$:

$$\frac{y-F(a)}{F(b)-F(a)}=\frac{x-a}{b-a}.$$

Полагая $x=x^{(1)}$ и $y=0$, получаем

$$x^{(1)}=a-\frac{F(a)}{F(b)-F(a)}\cdot(b-a).$$

Предположим, что вторая производная $F''(x)$ сохраняет постоянный знак, и рассмотрим два случая: $F(a)>0,\ F''(x)>0$ (рис.3,а) и $F(a)<0,\ F''(x)>0$ (рис.3,б). Случай $F''(x)<0$ сводится к рассматриваемому, если уравнение записать в форме: $-F(x)=0.$

Первому случаю (см. рис. 3, а) соответствует формула

$$\begin{aligned} &x^{(0)}=b,\\ &x^{(k+1)}=x^{(k)}-\frac{F(x^{(k)})}{F(x^{(k)})-F(a)}\cdot\left(x^{(k)}-a\right),\quad k=0,1,\ldots, \end{aligned} \tag{2.1}$$

а второму случаю (см. рис. 3,6) соответствует формула

$$\begin{aligned} &x^{(0)}=a,\\ &x^{(k+1)}=x^{(k)}-\frac{F(x^{(k)})}{F(b)-F(x^{(k)})}\cdot\left(b-x^{(k)}\right),\quad k=0,1,\ldots, \end{aligned} \tag{2.2}$$

В первом случае остается неподвижным конец a, а во втором случае - конец b.

Для выявления неподвижного конца используется условие $F''(x)\cdot F(t)>0,$ где $t=a$ или $t=b.$ Если неподвижен конец a, применяется формула (2.1), а если конец b, - формула (2.2).

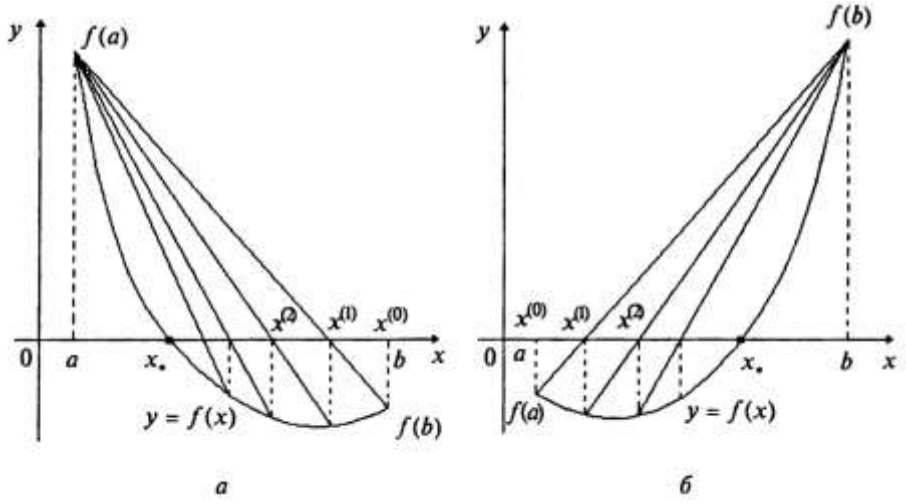


Рис. 3.

Блок-схема метода хорд аналогична блок-схеме метода бисекций с той лишь разницей, что в четвертом блоке нужно вместо формулы $x = \frac{a+b}{2}$ записать формулу (2.1) или (2.2). Кроме того, в блок-схему необходимо ввести операторы, вычисляющие значения $F(x)$ на границах новых отрезков.

Метод хорд в ряде случаев дает более быструю сходимость, чем метод деления отрезка пополам. Метод хорд, так же как и метод бисекций всегда сходится.

Пример. Уравнение $\sin 2x - \ln x = 0$ имеет единственный корень на отрезке $[1,3;1,5]$. Решим это уравнение с точностью до 10^{-4} методом хорд.

Решение. Чтобы уточнить корень методом хорд, определим знаки функций $F(x) = \sin 2x - \ln x$ на концах промежутка $[1,3;1,5]$ и знак второй производной в этом промежутке:

$$F(1,3) = 0,253137; \qquad F(1,5) = -0,264345;$$

$$F'(x) = 2\cos 2x - \frac{1}{x};$$

$$F''(x) = -4\sin 2x + \frac{1}{x^2} < 0 \quad \text{при} \quad x \in [1,3;1,5].$$

$F''(x) \cdot F(b) > 0$ и, следовательно, имеем второй случай. Положим $x^{(0)} = a$, $k = 0$. Тогда по формуле (2.2) получаем

$$x^{(1)} = x^{(0)} - \frac{F(x^{(0)})}{F(b) - F(x^{(0)})} \cdot \left(b - x^{(0)}\right) = 1,3 - \frac{0,253137}{-0,264345 - 0,253137} \cdot (1,5 - 1,3) = 1,397834.$$

Так как, $\left|x^{(1)} - x^{(0)}\right| = 0,097834 > \varepsilon = 10^{-4}$, продолжаем процесс до выполнения условия $\left|x^{(k+1)} - x^{(k)}\right| < \varepsilon$. При $k = 2$, это условие выполняется, т.е. $\left|x^{(3)} - x^{(2)}\right| = |1,399429 - 1,39941| = 0,000019 < \varepsilon = 0,0001$, и, следовательно, корень уравнения $x_* \cong 1,399429$.

Программа решения задачи на языке Pascal

```
program hord;
uses crt;
```

```
label 1,2;
var a,b,eps,x:real;
 function f(x:real):real;
begin
 f:= sin(2*x)-ln(x);
end;
begin clrscr;
    writeln(' Metod xord ');
write('a=');      read(a);
 write('b=');          read(b);
    write('eps=');         read(eps);
2:   x:=b;
    x:=b-f(b)*(b-a)/(f(b)-f(a));
if abs(x-b)<eps       then          goto 1
else begin        b:=x;        goto 2       end;
1:   writeln('x=',x:8:5);
end.
```

Результаты работы программы

1.3. Метод Ньютона (метод касательных).

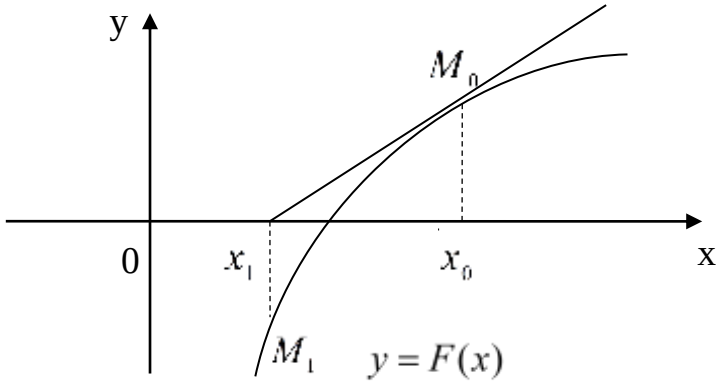
Метод Ньютона (метод касательных) является одним из наиболее популярных численных методов. Он быстро сходится (имеет квадратичную сходимость). Однако этот метод эффективен при весьма жестких ограничениях на характер функции $F(x)$:

1) существование второй производной функции $F(x)$ на множестве $G = \{a \le x \le b\}$;

2) удовлетворение первой производной условию $F'(x) \ne 0$ для всех $x \in G$;

3) знакопостоянство $F'(x), F''(x)$ для всех $x \in G$.

Поэтому его желательно использовать совместно с другими методами, например методом половинного деления, чтобы достигнуть диапазона $\tilde{a} \le x \le \tilde{b}$, где указанные условия начинают выполняться.



Этот метод в отличие от метода хорд на k-ой итерации вместо построения хорды требует построить касательную к кривой $y = F(x)$ при $x = x_k$, при этом за следующее приближение x_{k+1} принимается точка пересечения этой касательной с осью Ох. Пользуясь этим методом не обязательно знать отрезок $[a,b]$, где содержится корень уравнения $F(x) = 0$, а достаточно лишь найти некоторое начальное приближение к корню $x = x_0$.

Запишем уравнение касательной к кривой $y = F(x)$ в точке $M_0(x_0, F(x_0))$

$$y - F(x_0) = F'(x_0)(x - x_0) \tag{3.1}$$

Положим здесь $y = 0$, тогда x будет равен x_1.

$$-F(x_0) = F'(x_0)(x - x_0)$$

и найдем отсюда

$$x_1 = x_0 - \frac{F(x_0)}{F'(x_0)} \tag{3.2}$$

следующее приближение к корню x уравнения (I).

Аналогично можно найти и следующие приближения

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}, \tag{3.3}$$

здесь $k = 0,1,2,...,$ и $F'(x_k) \neq 0$.

Вычисления по формуле (3.3) надо вести до тех пор, пока

$$|F(x_{k+1})| \tag{3.4}$$

не станет меньше $\varepsilon > 0$ или не будет выполняться условие

$$|x_{k+1} - x_k| < \varepsilon \tag{3.5}$$

Метод Ньютона может применяться не только для нахождения простых корней, но для определения кратных корней, т.е. когда на отрезке $[a,b]$, содержащем корень, не выполняется условие $F(a)\cdot F(b)<0$.

Пример. Уравнение $\sin 2x-\ln x=0$ имеет единственный корень на отрезке $[1,3;1,5]$. Решить это уравнение с точностью до 10^{-4} методом Ньютона.

Решение. Зададим начальное приближение $x^{(0)}$. Так как для $F'(x)=2\cos 2x-\frac{1}{x}$, $F''(x)=-4\sin 2x+\frac{1}{x^2}$, то $F(1,5)=-0,264345$, $F''(x)<0$ при $x\in[1,3;1,5]$. Поэтому $F''(1,5)\cdot F(1,5)>0$ и $x^{(0)}=1,5$. Вычисляем $x^{(k+1)}$ по формуле

$$x^{(k+1)}=x^{(k)}-\frac{F(x^{(k)})}{F'(x^{(k)})},\quad k=0,1,\ldots$$

Результаты расчетов помещены в следующей таблице:

k	$x^{(k)}$	$f(x^{(k)})$	$\left\|x^{(k)}-x^{(k-1)}\right\|$
0	1,5	-0,264345	_____
1	1,400121	-0,001798	0,099879
2	1,3994289	$-1,9884\cdot 10^{-7}$	0,000692
3	1,3994288	$-2,5535\cdot 10^{-15}$	$7,6529\cdot 10^{-8}$

Из этой таблицы можно сделать следующий вывод:

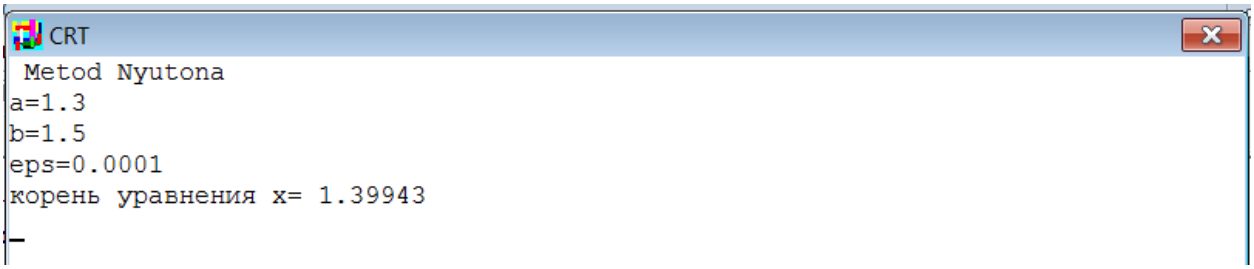
Для достижения заданной точности, проверяемой по модулю разности $\left|x^{(k)}-x^{(k-1)}\right|$, потребовалось выполнить три приближения. Если же выход организовать по условию $\left|F'(x^{(k)})\right|\le\varepsilon$, то достаточно двух приближений.

Программа решения задачи на языке Pascal

```
program newton;
uses crt;
function f(x:real):real;
begin
f:= sin(2*x)-ln(x)
```

```
end;
function df(x:real):real;
begin
df:=2*cos(2*x)-1/x
end;
var a,b,eps,x,xp:real;
n:integer;
begin clrscr;
writeln(' Metod Nyutona ');
 write('a=');          read(a);
 write('b=');          read(b);
    write('eps=');         read(eps);            x:=a;
repeat
    xp:=x;        x:=xp-f(xp)/df(xp);
until abs(x-xp)<=eps;
    writeln ('корень уравнения x=',x:8:5);
end.
```

Результат работы программы



1.4. Модифицированный метод Ньютона.

Чтобы уменьшить количество арифметических операций на каждом шаге итераций в вычислительной практике для решения уравнений вида (I) часто используют следующие модифицированные методы Ньютона:

Разностный метод с постоянным шагом

Отличие этого метода от метода Ньютона состоит в том, что в рабочей формуле (3.3) вместо величин $F'(x_k)$, стоящей в знаменателе, используют величину $F'(x_0)$, которая не зависит от номера итерации и может быть вычислена заранее всего один раз. Таким образом, рабочая формула модифицированного метода Ньютона имеет вид

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_0)},\ k = 0,1,2,..., \tag{4.1}$$

Для сложных функций вычисление $F'(x)$ достаточно трудоемко, поэтому заменим в (4.1) производную по определению

$$F'(x_k) = \lim_{h \to 0} \frac{F(x_k + h) - F(x_k)}{h}.$$

При малых значениях шага h получим приближенное равенство

$$x_{k+1} = x_k - \frac{F(x_k) \cdot h}{F(x_k + h) - F(x_k)}. \tag{4.2}$$

Разностный метод с переменным шагом

Шаг h можно изменять на каждой итерации либо проводить несколько итераций с одним шагом, затем его изменить (в зависимости от свойств функции). Тогда получим набор $h_1, h_2, ...$

Тогда (4.2) примет вид

$$x_{k+1} = x_k - \frac{F(x_k) \cdot h_k}{F(x_k + h_k) - F(x_k)}, \quad n \neq k. \tag{4.3}$$

Преимуществом методов этой группы является отсутствие производной. Недостатком – низкая скорость сходимости. Трудность в использовании метода Ньютона и модифицированного метода Ньютона состоит в выборе начального приближения, которое обязательно должно принадлежать некоторой окрестности корня решаемого уравнения. Поэтому иногда целесообразно применить смешанный алгоритм. Он состоит в том, что сначала применяется всегда сходящийся метод (например, метод деления отрезка пополам или метод хорд), а после некоторого числа итераций – быстро сходящийся метод Ньютона.

1.5. Метод простой итерации.

В ряде случаев весьма удобным приёмом уточнения корня уравнения является метод последовательных приближений (метод итераций).

При использовании этого метода исходное нелинейное уравнение (I) записывается в виде

$$x = \varphi(x) \tag{5.1}$$

Пусть начальное приближение к корню уравнения (6.1) известно и равно

$$x = x_0 \tag{5.2}$$

Подставим (5.2) в правую часть (5.1) и получим

$$x_1 = \varphi(x_0) \tag{5.3}$$

Подставляя (5.3) в правую часть (5.1), получаем

$$x_2 = \varphi(x_1) \tag{5.4}$$

и т.д.

Таким образом, рабочая формула метода простой итерации имеет вид

$$x_{k+1} = \varphi(x_k),\ (k = 0,1,...) \tag{5.5}$$

Уравнение (I) может быть приведено к итерационному виду (5.1) несколькими способами, однако это надо сделать так, чтобы для функции $\varphi(x)$ выполнялись условия следующей теоремы.

Теорема. Пусть уравнение $x = \varphi(x)$ имеет единственный корень на отрезке $[a;b]$ и выполнены условия:

1) $\varphi(x)$ определена и дифференцируема на $[a;b]$;

2) $\varphi(x) \in [a;b]$ для всех $x \in [a;b]$;

3) существует такое вещественное q, что $|\varphi'(x)| \le q < 1$ для всех $x \in [a;b]$. Тогда итерационная последовательность $x_{k+1} = \varphi(x_k),\ (k = 0,1,...)$ сходится при любом начальном члене $x_0 \in [a;b]$.

Заметим, что условия данной теоремы не являются необходимыми, т.е. итерационная последовательность может оказаться сходящейся и при невыполнении этих условий.

Счет по формуле (5.5) проводить до тех пор, пока не будет выполняться условие

$$|x_{k+1} - x_k| < \varepsilon, \quad \text{или} \qquad |x_{k+1}| < \varepsilon \tag{5.6}$$

Пример. Уравнение $\sin 2x - \ln x = 0$ имеет единственный корень на отрезке $[1,3;1,5]$. Решим это уравнение с точностью до 10^{-4} методом простой итерации.

Решение. Исходное уравнение можно привести к итерационному виду несколькими способами, например:

1) $x = \exp(\sin 2x)$;

2) $x = (-1)^n\, 0,5(\arcsin \ln x + \pi n),\ n = \pm 1, \pm 2, ...$

3) $x = x - m(\sin 2x - \ln x), \quad m \neq 0.$

Функцию $\varphi(x)$ будем искать из соотношения $\varphi(x) = x - mF(x)$, где m – отличная от нуля константа. В этом случае

$$\varphi(x) = x - m(\sin 2x - \ln x), \quad \varphi'(x) = 1 - m\left(2\cos 2x - \frac{1}{x}\right).$$

Подберём константу $m \neq 0$ так, чтобы для функции $\varphi(x)$ выполнялись второе и третье условия теоремы о сходимости итераций. Обозначим $F(x) = \sin 2x - \ln x$. Заметим, что производная $F'(x)$ на отрезке $[1,3;1,5]$ отрицательна, следовательно, $F(x)$ на этом отрезке монотонно убывает. На концах отрезка имеем: $F(1,3) = 0,253137$, $F(1,5) = -0,264344$. Функция $\varphi(x)$ тоже убывает на отрезке $[1,3;1,5]$, а её значения на концах зависят от m:

$$\varphi(1,3) = 1,3 - 0,253137m, \quad \varphi(1,5) = 1,5 - 0,264344m.$$

Учитывая монотонность функции $\varphi(x)$, из последних равенств легко заметить, что второе условие теоремы заведомо выполнено, если m – отрицательное число, достаточно малое по модулю.

Займёмся проверкой третьего условия теоремы. Поскольку $F'(x)$ на отрезке $[1,3;1,5]$ отрицательна и монотонно убывает, её модуль имеет максимум на правом конце отрезка: $|F'(1,5)| = 2,6466526$. Если принять

$$m = \frac{1}{|F'(1,5)|} \approx -0,37,$$

то для всех x отрезка $[1,3;1,5]$ имеем

$$0 < m\left(2\cos 2x - \frac{1}{x}\right) < 1,$$

тогда

$$\max|\varphi'(x)| = \max\left|1 + 0,37\left(2\cos 2x - \frac{1}{x}\right)\right| \approx 0,08128 < 0,1$$

при $1,3 \le x \le 1,5$.

Следовательно, можно принять $q = 0,1$. Таким образом, преобразуем исходное уравнение к виду: $x = x + 0,37(\sin 2x - \ln x)$. В качестве начального приближения выберем $x^{(0)} = \frac{1,3 + 1,5}{2} = 1,4$. Выполняем последовательные действия по формуле (5.5): $x^{(k+1)} = x^{(k)} + 0,37\left(\sin 2x^{(k)} - \ln x^{(k)}\right)$.

Результаты расчетов приведены в следующей таблице:

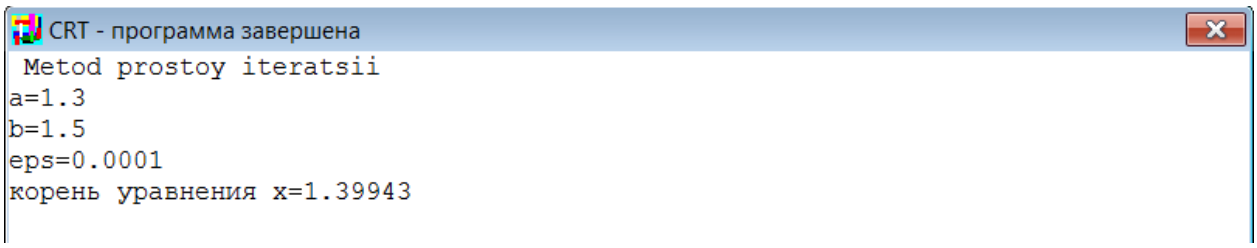
k	$x^{(k)}$	$f(x^{(k)})$	$\left\|x^{(k)} - x^{(k-1)}\right\|$
0	1,4	-0,001484	_____
1	1,399451	$-5{,}72 \cdot 10^{-5}$	0,000549
2	1,39943	$-2{,}21 \cdot 10^{-6}$	0,000021

На второй итерации выполнилось условие $\left|x^{(2)} - x^{(1)}\right| \le \varepsilon = 10^{-4}$, поэтому процесс завершен. В качестве приближенного решения берется $x_* \cong x^{(2)} = 1{,}39943$. Подстановка $x^{(2)}$ в исходное уравнение дает $\sin 2x^{(2)} - \ln x^{(2)} = -0{,}0000029$, т.е. равенство в уравнении также выполняется с заданной точностью.

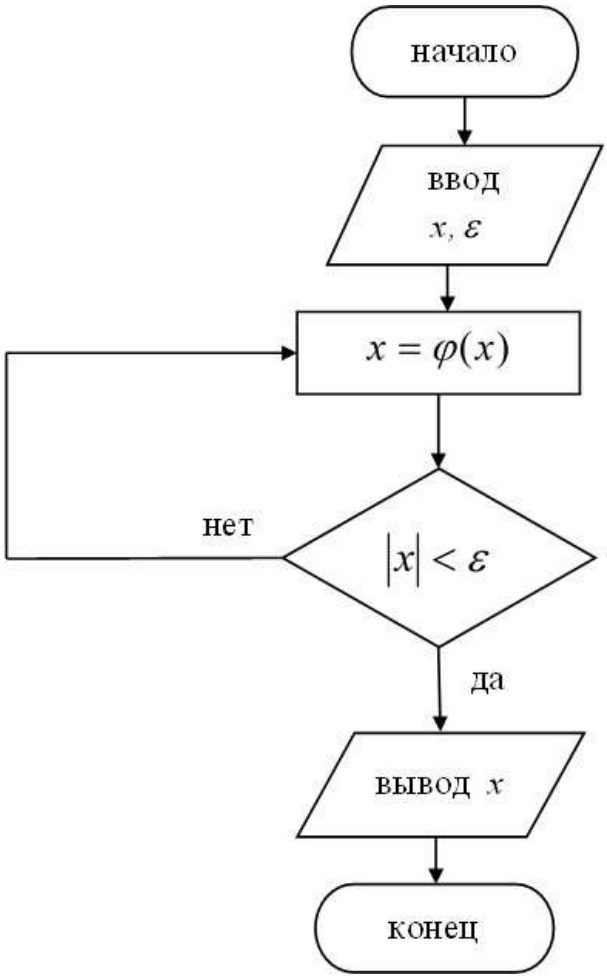
Программа решения задачи на языке Pascal

```
program iteratio;
uses crt;
function f(x:real):real;
begin
f:= sin(2*x)-ln(x)
end;
var a,b,eps,x,xp:real;
begin clrscr;
writeln(' Metod prostoy iteratsii ');
 write('a=');          read(a);
 write('b=');          read(b);
    write('eps=');         read(eps);
    x:=a;
repeat
    xp:=x;         x:=xp+0.37*f(xp);
until abs(x-xp)<=eps;
    writeln ('корень уравнения x=',x:4:5);
end.
```

Результат работы программы



Блок-схема метода простой итерации может быть следующей.

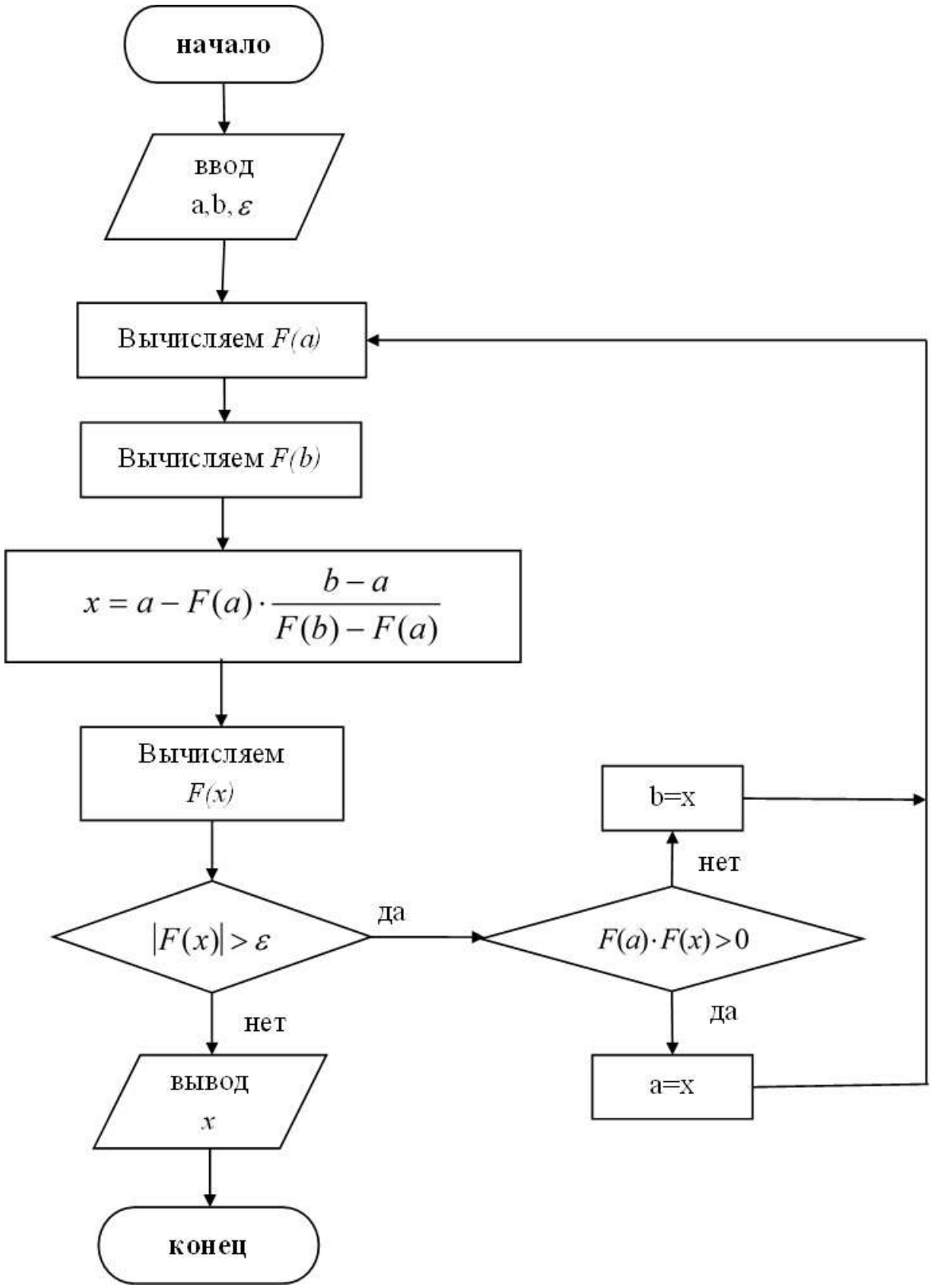


Замечания

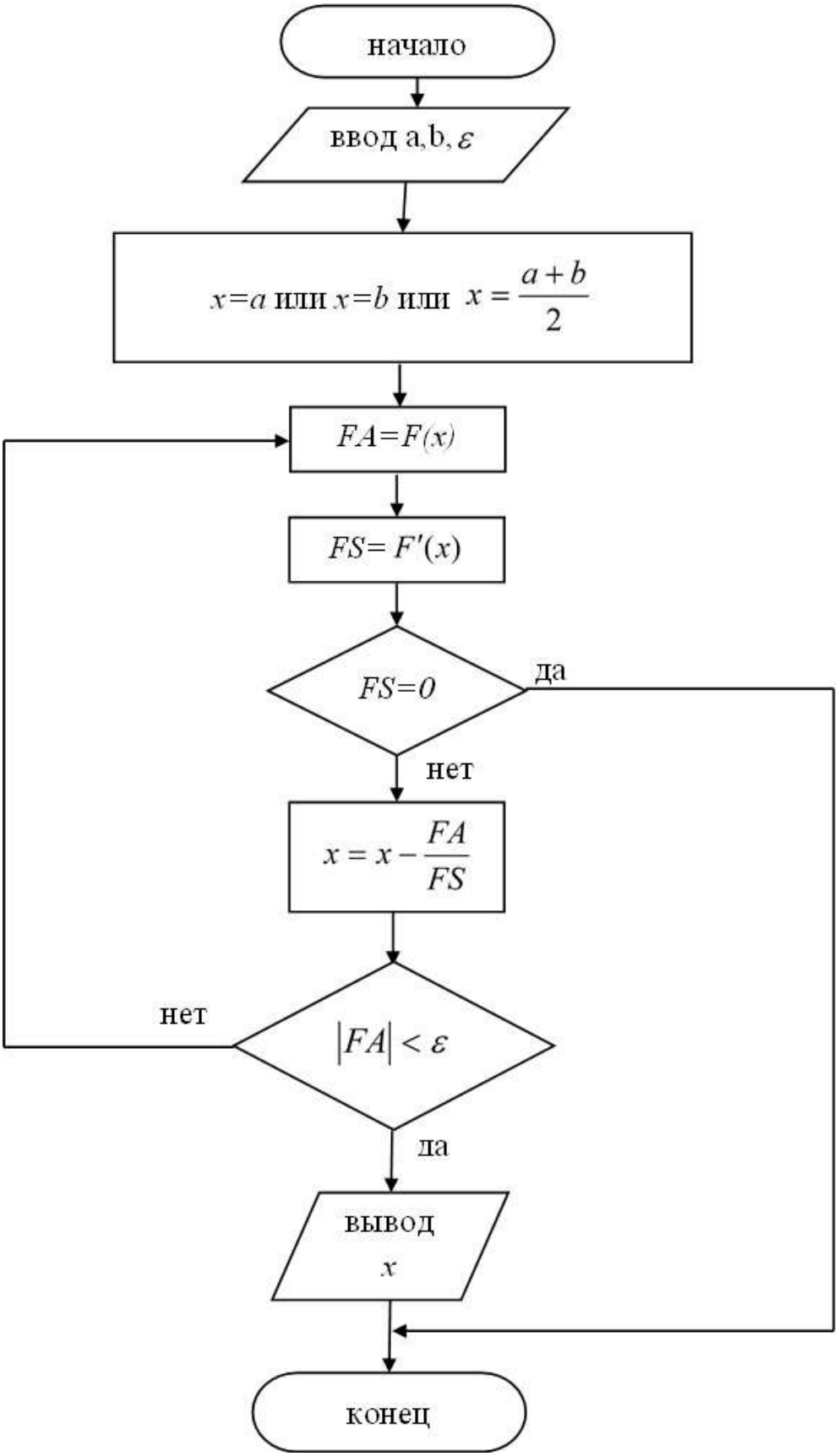
1. Метод секущих является более экономичным по сравнению с методом Ньютона по количеству функций, подлежащих расчету: на каждой итерации в методе секущих необходимо рассчитать только значение $F(x^{(k)})$ так как величина $F(x^{(k-1)})$ уже подсчитана на предыдущей итерации.

2. Для всех описанных модификаций метода Ньютона скорость сходимости p по сравнению с методом касательных снижается: $p < 2$. Однако для некоторых из них (метод секущих) значение $p > 1$ и может достигать $p = 1{,}5$.

Блок-схема метода секущих



Блок-схема метода касательных



Задания для самостоятельной работы.

Во всех заданиях уточнить действительные корни уравнений с заданной точностью ε вышеуказаннымы методами.

№1. $x = \sin x, \quad x_0 \in \left[-\frac{\pi}{2}, -\frac{\pi}{6}\right], \quad \varepsilon = 10^{-3}$

№2. $x = \sin x, \quad x_0 \in \left[-\frac{\pi}{5}, \frac{\pi}{3}\right], \quad \varepsilon = 10^{-3}$

№3. $x = \sin x, \quad x_0 \in \left[\frac{\pi}{6}, \frac{\pi}{2}\right], \quad \varepsilon = 10^{-3}$

№4. $x = \cos x, \quad x_0 \in \left[\frac{\pi}{6}, \pi\right], \quad \varepsilon = 10^{-3}$

№5. $x = tg\, x, \quad x_0 \in \left[-\frac{\pi}{5}, \frac{\pi}{3}\right], \quad \varepsilon = 10^{-3}$

№6. $3tg^2 x - 8tgx + 5 = 0, \quad x_0 \in \left[\frac{\pi}{6}, \pi\right], \quad \varepsilon = 10^{-3}$

№7. $3tg^2 x - 8tgx + 5 = 0, \quad x_0 \in \left[\frac{\pi}{4}, \frac{7\pi}{4}\right], \quad \varepsilon = 10^{-3}$

№8. $tg^2 x - 5tgx - 6 = 0, \quad x_0 \in \left[-\pi, -\frac{\pi}{6}\right], \quad \varepsilon = 10^{-3}$

№9. $tgx = 2\sin x, \quad x_0 \in \left[\frac{3\pi}{4}, \frac{7\pi}{4}\right], \quad \varepsilon = 10^{-3}$

№10. $tgx = 2\sin x, \quad x_0 \in \left[\frac{4\pi}{3}, \frac{8\pi}{3}\right], \quad \varepsilon = 10^{-3}$

№11. $x + \lg x = 0,5, \quad x_0 \in [0.6, 0.7], \quad \varepsilon = 10^{-3}$

№12. $x^3 + 5x^2 - 15x - 7 = 0, \quad x_0 \in [2.4, 2.5], \quad \varepsilon = 10^{-3}$

№13. $x^3 + 3x^2 - 1 = 0, \quad x_0 \in [-3, -2], \quad \varepsilon = 10^{-3}$

№14. $x^3 - 0,4x + 0,08 = 0, \quad x_0 \in [0,15, 0,25], \quad \varepsilon = 10^{-3}$

№15. $e^{0,424x} - 2,831x = 0, \quad x_0 \in [0.45, 0.55], \quad \varepsilon = 10^{-3}$

№16. $\cos(x) - x^3 = 0, \quad x_0 \in [-3, 2], \quad \varepsilon = 10^{-3}$

№17. $x - 10\sin(x) = 0, \quad x_0 \in [2, 7], \quad \varepsilon = 10^{-3}$

№18. $4x^4 - 6,2 - \cos(2x) = 0, \quad x_0 \in [-3, 2], \quad \varepsilon = 10^{-3}$

№19. $3\cos(x) - \sqrt{4x+7} = 0, \quad x_0 \in [0, 5], \quad \varepsilon = 10^{-3}$

№20. $x\sin(x) - 1 = 0, \quad x_0 \in [-2, 3], \quad \varepsilon = 10^{-3}$

№21. $8\cos(x) - x - 6 = 0, \quad x_0 \in [-5, 1], \quad \varepsilon = 10^{-3}$

№22. $\sin(x) - 0,2x = 0, \quad x_0 \in [-2, 3], \quad \varepsilon = 10^{-3}$

№23. $10\cos(x) - 0,1x^2 = 0, \quad x_0 \in [-2, 3], \quad \varepsilon = 10^{-3}$

№24. $5\sin(2x) - \sqrt{1-x} = 0, \quad x_0 \in [-10, -5], \quad \varepsilon = 10^{-3}$

№25. $3\sin(2x) - 2x^3 = 0, \quad x_0 \in [-2, 3], \quad \varepsilon = 10^{-3}$

№26. $3\sin(8x) - 7x = 0, \quad x_0 \in [-1, 1], \quad \varepsilon = 10^{-3}$

№27. $(x+5)^2 + 6x + x^4 - 13 = 0, \quad x_0 \in [0, 5], \quad \varepsilon = 10^{-3}$

№28. $x^3 + 3x^2 - 1 = 0, \quad x_0 \in [-3, -2], \quad \varepsilon = 10^{-3}$

№29. $x^3 - 2x^2 - 4x + 7 = 0, \quad x_0 \in [1, 2], \quad \varepsilon = 10^{-3}$

№30. $2x + \lg(2x+3) - 1 = 0, \quad x_0 \in [0, 0.5], \quad \varepsilon = 10^{-3}$

ГЛАВА 2. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ.

Цель работы. Знакомство с приближёнными методами решения систем нелинейных уравнений и их численной реализацией на ПК.

Предварительные замечания.

Методы решения систем уравнений обычно разделяют на две большие группы. К первой группе относят методы, которые называют точными. Они позволяют для любых систем найти точные значения неизвестных после конечного числа арифметических операций, каждая из которых выполняется точною. Ко второй группе относятся все методы, которые не являются точными. Их называют приближёнными, или численными, или итерационными. Точное решение при использовании этих методов получается в результате бесконечного процесса приближений.

Рассмотрим некоторые приближённые методы решения систем нелинейных уравнений и алгоритмы их численной реализации на ПК. Будем предполагать, что решение этих систем должно быть получено с точностью до ε, где ε - очень маленькое положительное число.

2.1. Метод простой итерации.

Запишем систему нелинейных уравнений в виде

$$\begin{cases} F_1(x_1, x_2, \ldots, x_n) = 0, \\ F_2(x_1, x_2, \ldots, x_n) = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ F_n(x_1, x_2, \ldots, x_n) = 0. \end{cases} \qquad (1.1)$$

или коротко в виде $F_i(x_1, x_2, \ldots, x_n) = 0$, где $i = 1,2,\ldots,n$.

Здесь функции, стоящие слева в (1.1) определены и непрерывны вместе со своими частными производными в некоторой области D, которой принадлежит точное решение рассматриваемой системы уравнений. Точное решение системы (1.1) обозначим

$$X^* = (x_1^*, x_2^*, \ldots, x_n^*) \qquad (1.2)$$

Для того, чтобы систему (1.1) решить методом простой итерации, во-первых, преобразуем её к виду

$$\begin{aligned} x_1 &= f_1(x_1, x_2, \ldots, x_n) \\ x_2 &= f_2(x_1, x_2, \ldots, x_n) \\ &\ldots\ldots\ldots\ldots\ldots \\ x_n &= f_n(x_1, x_2, \ldots, x_n) \end{aligned} \qquad (1.3)$$

или, коротко к виду, $x_i = f_i(x_1, x_2, \ldots, x_n)$, где $i = 1,2,\ldots,n$.

Функции, стоящие справа в (1.3) обладают теми же свойствами, что и функции в (1.1).

Во-вторых, в области D выберем любую точку $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ и назовём её нулевым приближением к точному решению системы (1.3).

В-третьих, координаты точки X^0 подставим в правую часть системы (1.3) и вычислим значения величин, стоящих слева в этой системе.

Будем иметь

$$\begin{aligned} x_1^1 &= f_1(x_1^0, x_2^0, \ldots, x_n^0) \\ x_2^1 &= f_2(x_1^0, x_2^0, \ldots, x_n^0) \\ &\ldots\ldots\ldots\ldots\ldots \\ x_n^1 &= f_n(x_1^0, x_2^0, \ldots, x_n^0) \end{aligned} \qquad (1.4)$$

или коротко $x_i^1 = f_i(x_1^0, x_2^0, \ldots, x_n^0), i = 1,2,\ldots,n.$

Величины $x_1^1, x_2^1, \ldots, x_n^1$, стоящие слева в формулах (1.4), будем считать координатами точки $X^1 = (x_1^1, x_2^1, \ldots, x_n^1)$. Эту точку назовём первым приближением к точному решению исходной системы, т.е. к $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$. Теперь мы имеем два приближённых решения системы (1.3). Этими решениями являются $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ и $X^1 = (x_1^1, x_2^1, \ldots, x_n^1)$.

В четвёртых, сравним эти два приближённых решения на ε:

$$\left| x_i^0 - x_i^1 \right| \le \varepsilon, \quad i = 1, 2, \ldots, n. \qquad (1.5)$$

Если все неравенства (1.5) выполняются, то за приближённое решение исходной системы можно выбрать как $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, так и

$X^1 = (x_1^1, x_2^1, ..., x_n^1)$, поскольку эти два решения отличаются друг от друга не больше чем на ε.

На этом решение исходной системы методом простой итерации заканчивается. Если же хотя бы одно из неравенств (1.5) не выполняется, то надо компоненты первого приближения подставить в правую часть системы (1.3) и вычислить второе приближение $X^2 = (x_1^2, x_2^2, ..., x_n^2)$. Здесь

$$\begin{aligned} x_1^2 &= f_1(x_1^1, x_2^1, ..., x_n^1) \\ x_2^2 &= f_2(x_1^1, x_2^1, ..., x_n^1) \\ &\cdots\cdots\cdots\cdots\cdots \\ x_n^2 &= f_n(x_1^1, x_2^1, ..., x_n^1) \end{aligned} \qquad (1.6)$$

или коротко $x_i^2 = f_i(x_1^1, x_2^1, ..., x_n^1), \ i = 1,2,...,n.$

Далее надо сравнить приближения X^1 и X^2 на ε по формуле (1.5). Строить приближения надо до тех пор, пока два соседних приближения $X^k = (x_1^k, x_2^k, ..., x_n^k)$ и $X^{k+1} = (x_1^{k+1}, x_2^{k+1}, ..., x_n^{k+1})$ будут отличаться друг от друга не больше чем на ε.

Запишем рабочие формулы метода простой итерации для системы (1.3) в компактном виде.

Вычислить

$$x_i^{k+1} = f_i(x_1^k, x_2^k, ..., x_n^k), \ i = 1,2,...,n, \ k = 0,1,2,.... \qquad (1.7)$$

и построить приближения к решению системы (1.3)

$$X^{k+1} = (x_1^{k+1}, x_2^{k+1}, ..., x_n^{k+1}) \text{ для всех } i = 1,2,...,n \text{ и } k = 0,1,2,... \qquad (1.8)$$

Сформулируем алгоритм вычислений по формулам (1.7) и (1.8).

1. Выберем $X^0 = (x_1^0, x_2^0, ..., x_n^0)$, принадлежащую D.

2. В (1.7) положим $k = 0$, получим $x_i^1 = f_i(x_1^0, x_2^0, ..., x_n^0), \ i = 1,2,...,n.$

3. По (1.8) построим $X^1 = (x_1^1, x_2^1, ..., x_n^1)$.

4. Проверим условие (1.5) на ε: $|x_i^0 - x_i^1| \le \varepsilon, i = 1,2,...,n.$

5. Если все условия в п.4 выполнены, то заканчиваем вычисления, выбрав за приближённое решение исходной системы

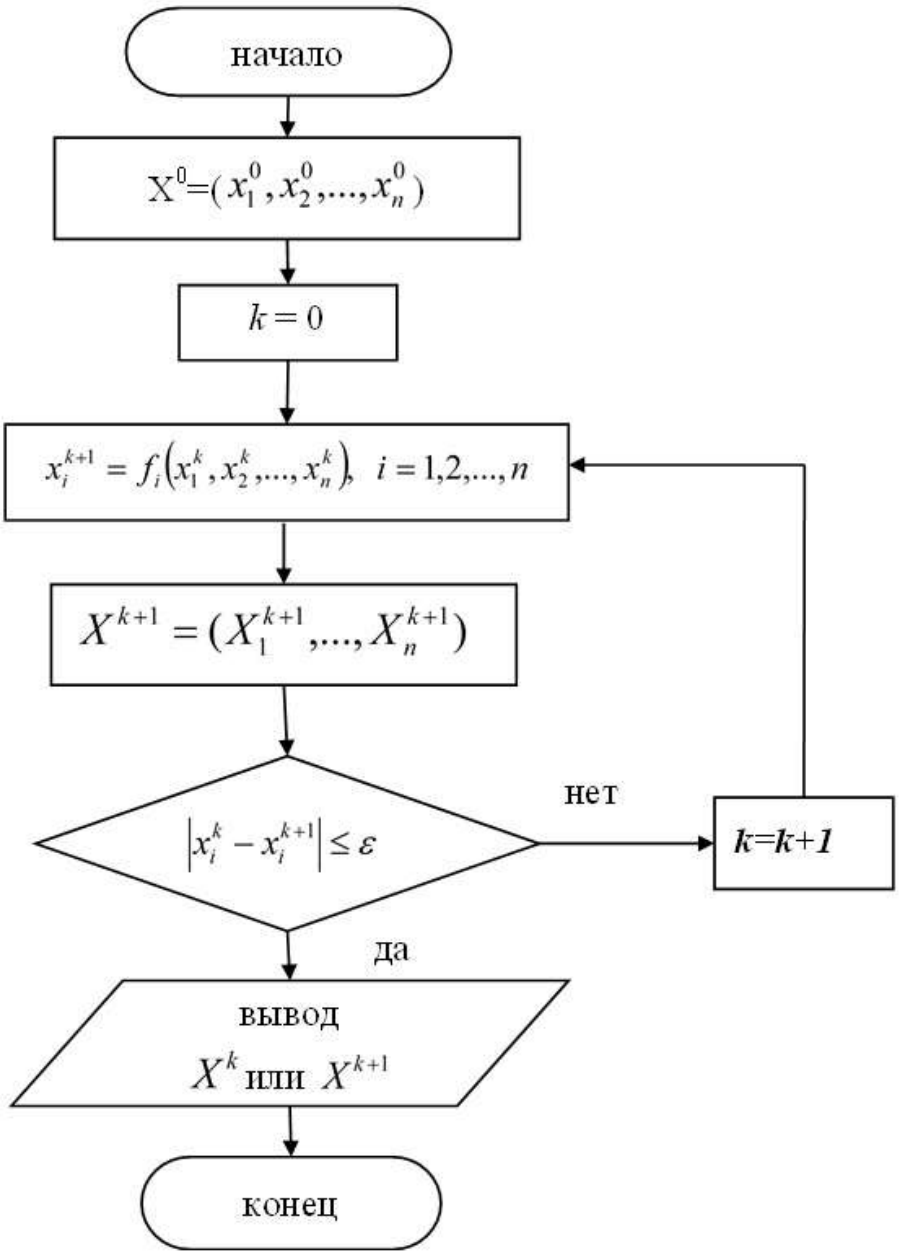
$X^0=(x_1^0,x_2^0,...,x_n^0)$ или $X^1=(x_1^1,x_2^1,...,x_n^1)$ всё равно, т.к. эти решения отличаются друг от друга не больше чем на ε. Если хотя бы одно из условий в п.4 не выполнилось, то переходим к п.6.

6. В (1.7) положим $k=k+1$ и получим $x_i^2=f_i\left(x_1^1,x_2^1,...,x_n^1\right), i=1,2,...,n.$

7. По (1.8) построим $X^2=\left(x_1^2,x_2^2,...,x_n^2\right).$

8. Перейдём к п.4, при этом верхние индексы в условии (1.5) изменятся и станут на единицу больше.

Блок-схема этого алгоритма приведена на следующем рисунке.



Пример. Решить методом простых итераций систему

$$\begin{cases} \sin(x-0,6)-y=1,6; \\ 3x-\cos y=0,9. \end{cases}$$

В качестве нулевого приближения возьмём точку $x=0,15,\ y=-2.$

Преобразуем систему к виду:

$$\begin{cases} x=\dfrac{1}{3}\cos y+0,3 \\ y=\sin(x-0,6)-1,6 \end{cases}$$

Частные производные:

$f_x=0,\ f_y=-\dfrac{1}{3}\sin y,\ g_x=\cos(x-0,6),\ g_y=0.$ Система имеет решение в области $D: x\in(0;0,3);\ y\in(-2,2;-1,8).$ В этой области имеем:

$$\left|\frac{\partial f}{\partial x}\right|+\left|\frac{\partial g}{\partial x}\right|=\left|\cos(x-0,6)\right|\le\cos 0,3<1;$$

$$\left|\frac{\partial f}{\partial y}\right|+\left|\frac{\partial g}{\partial y}\right|=\left|-\frac{1}{3}\sin y\right|<\left|\frac{1}{3}\sin(-1,8)\right|<1.$$

Таким образом, условия сходимости выполняются. Вычисления производится по формулам:

$$\begin{cases} x_{k+1}=\dfrac{1}{3}\cos y_k+0,3; \\ y_{k+1}=\sin(x_k-0,6)-1,6. \end{cases}$$

Программа решения задачи на языке Pascal.

```
Program MetodProstIter;    {метод простых итераций}
uses crt;
label vihod;
Var k:integer;             {Счетчик итераций}
var eps,x0,y0,x1,y1,x,y,U1,U2,d:real;
{Приведение системы к итерационному виду}

function fnX(x:real;y:real):real;
begin
fnX:=cos(y)/3+0.3;
end;
function fnY(x:real;y:real):real;
begin
fnY:=sin(x-0.6)-1.6;
```

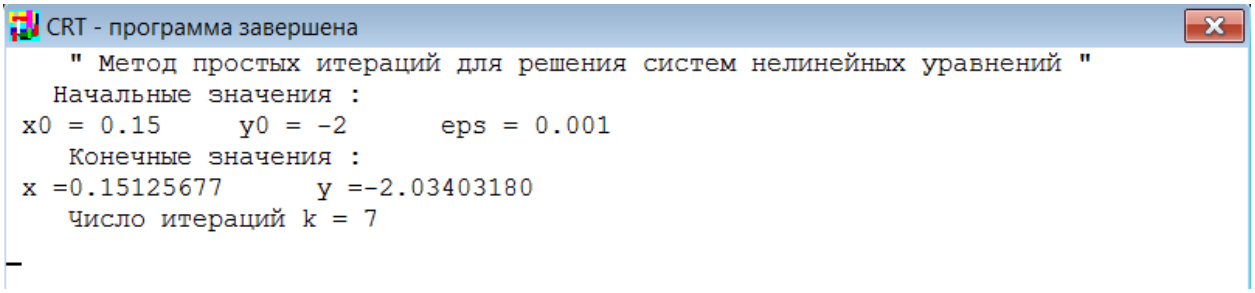
```
    end;

    {Определение частных производных для проверки условия
сходимости}
    function fnXx(x:real;y:real):real;
    begin
    fnXx:=0;
    end;
    function fnXy(x:real;y:real):real;
    begin
    fnXy:=-sin(y)/3;
    end;
    function fnYx(x:real;y:real):real;
    begin
    fnYx:=cos(x-0.6);
    end;
    function fnYy(x:real;y:real):real;
    begin
    fnYy:=0;
    end;
    begin clrscr;
    writeln('    " Метод простых итераций для решения
систем нелинейных уравнений "    ');
    x0:=0.15;      y0:=-2;     {Начальные приближения
решения}
    eps:=0.001;                {Точность вычислений}
    x:=x0;         y:=y0;
    k:=0;                  {Обнуление счетчика}
    repeat
    begin
    if (k>50)     then
    begin
    writeln('Уточните начальное приближение. Число
итераций велико!');
    goto vihod;                {Выход при зацикливании}
    end;
    k:=k+1;                    {Счетчик числа итераций}
    x1:= fnX(x,y);   y1:=fnY(x,y); {Уточнение решений}
```

```
    {Проверка условия сходимости и предупреждение о
нарушении}
    U1:=abs(fnXx(x,y))+abs(fnYx(x,y));
    U2:=abs(fnXy(x,y))+abs(fnYy(x,y));
    if ((U1>=1) or (U2 >=1 )) then
    writeln(' Условия сходимости нарушено при k=',k);

    { Расстояние между точками (x,y) и (x1,y1) }
    d:=sqrt(sqr(x1-x)+sqr(y1-y));
    x:=x1;     y:=y1;       { Переобозначение координат }
    end;
    until d < eps;
    vihod:
    writeln('   Начальные значения  :');
    writeln(' x0 = ',x0,'   y0 = ',y0,'   eps = ',eps);
    writeln('    Конечные значения  :');
    writeln(' x =',x:8:8,'    y =',y:8:8);
    writeln('    Число итераций  k = ',k);
    end.
```

Результаты работы программы



2.2. Метод Ньютона решения систем нелинейных уравнений.

Этот метод обладает гораздо более быстрой сходимостью, чем метод простой итерации. Основная идея метода Ньютона состоит в выделении из системы линейных частей, которые являются главными при малых приращениях аргументов. Это позволяет свести исходную задачу к решению последовательности систем линейных уравнений.

В основе метода Ньютона для системы уравнений (1.1) лежит использование разложения функций

$$F_i(x_1, x_2, \ldots, x_n) = 0, \qquad \text{где} \quad i = 1,2,\ldots,n \qquad (2.1)$$

в ряд Тейлора, причём члены, содержащие вторые и более высокие порядки производных, отбрасываются. Такой подход позволяет решение одной нелинейной системы (1.1) заменить решением ряда линейных систем.

Итак, систему (1.1) будем решать методом Ньютона. В области D выберем любую точку $X^0=(x_1^0,x_2^0,...,x_n^0)$ и назовём её нулевым приближением к точному решению $X^*=(x_1^*,x_2^*,...,x_n^*)\in D$ исходной системы. Теперь функции (2.1) разложим в ряд Тейлора в окрестности точки $X^0=(x_1^0,x_2^0,...,x_n^0)$. Будем иметь

$$F_i(x_1,x_2,...,x_n)\cong F_i(x_1^0,x_2^0,...,x_n^0)+\frac{\partial F_i}{\partial x_1}(x_1-x_1^0)+...+\frac{\partial F_i}{\partial x_n}(x_n-x_n^0), i=1,2,...n. \qquad (2.2)$$

Так как левые части (2.2) должны обращаться в ноль согласно (1.1), то и правые части (2.2) тоже должны обращаться в ноль. Поэтому из (2.2) имеем

$$\frac{\partial F_i}{\partial x_1}\Delta x_1^0+...+\frac{\partial F_i}{\partial x_n}\Delta x_n^0=-F_i(x_1^0,x_2^0,...,x_n^0), i=1,2,...,n. \qquad (2.3)$$

Здесь

$$\Delta x_i^0=x_i-x_i^0,\ i=1,2,...,n. \qquad (2.4)$$

Все частные производные в (2.3) должны быть вычислены в точке $X^0=(x_1^0,x_2^0,...,x_n^0)$.

(2.3) есть система линейных алгебраических уравнений относительно неизвестных $\Delta x_i^0=x_i-x_i^0, i=1,2,...,n.$ Эту систему можно решить методом Крамера, если её основной определитель будет отличен от нуля и найти величины $\Delta x_i^0=x_i-x_i^0, i=1,2,...,n.$

Теперь можно уточнить нулевое приближение $X^0=(x_1^0,x_2^0,...,x_n^0)$, построив первое приближение с координатами

$$x_1^1=x_1^0+\Delta x_1^0, x_2^1=x_2^0+\Delta x_2^0,...,x_n^1=x_n^0+\Delta x_n^0, \qquad (2.5)$$

т.е.

$$X^1=(x_1^1,x_2^1,...,x_n^1). \qquad (2.6)$$

Выясним, получено ли приближение (2.6) с достаточной степенью точности. Для этого проверим условие

$$\max\left|\Delta x_i^0\right|\le\varepsilon,\ i=1,2,...,n \qquad (2.7)$$

где ε наперёд заданное малое положительное число (точность, с которой должна быть решена система (1.1)). Если условие (2.7) будет

выполнено, то за приближённое решение системы (1.1) выберем (2.6) и закончим вычисления. Если же условие (2.7) выполняться не будет, то выполним следующее действие. В системе (2.3) вместо $x_i^0, i = 1,2,\ldots,n$ возьмём уточнённые значения

$$x_i^1, i = 1,2,\ldots,n, \tag{2.8}$$

т.е. выполним следующие действия

$$x_i^0 = x_i^1, i = 1,2,\ldots,n. \tag{2.9}$$

После этого система (2.3) будет системой линейных алгебраических уравнений относительно величин $\Delta x_i^1 = x_i - x_i^1, i = 1,2,\ldots,n.$ Определив эти величины, следующее второе приближение $X^2 = (x_1^2, x_2^2,\ldots,x_n^2)$ к решению системы (1.1) найдём по формулам

$$x_1^2 = x_1^1 + \Delta x_1^1, x_2^2 = x_2^1 + \Delta x_2^1,\ldots,x_n^2 = x_n^1 + \Delta x_n^1. \tag{2.10}$$

Теперь проверим условие (2.7) $\max\left|\Delta x_i^1\right| \le \varepsilon, i = 1,2,\ldots,n.$

Если это условие выполняется, то заканчиваем вычисления, приняв за приближённое решение системы (1.1) второе приближение $X^2 = (x_1^2, x_2^2,\ldots,x_n^2)$. Если же это условие не выполняется, то продолжаем строить следующее приближение, приняв в (2.3) $x_i^1 = x_i^2, i = 1,2,\ldots,n.$ Строить приближения нужно до тех пор, пока условие на ε не будет выполнено.

Рабочие формулы метода Ньютона для решения системы (1.1) можно записать в виде.

Вычислить последовательность

$$X^{k+1} = (x_1^{k+1}, x_2^{k+1},\ldots,x_n^{k+1}), k = 0,1,2,\ldots, \tag{2.11}$$

где $x_1^{k+1} = x_1^k + \Delta x_1^k, x_2^{k+1} = x_2^k + \Delta x_2^k,\ldots,x_n^{k+1} = x_n^k + \Delta x_n^k.$ (2.12)

Здесь $\Delta x_i^{k+1}, i = 1,2,\ldots,n$ являются решением системы

$$\frac{\partial F_i}{\partial x_1}\Delta x_1^k + \ldots + \frac{\partial F_i}{\partial x_n}\Delta x_n^k = -F_i(x_1^k, x_2^k,\ldots,x_n^k), i = 1,2,\ldots,n. \tag{2.13}$$

Сформулируем алгоритм вычислений по формулам (2.11) – (2.13).

1. Выберем нулевое приближение $X^0 = (x_1^0, x_2^0,\ldots,x_n^0)$, принадлежащее области D.

2. В системе линейных алгебраических уравнений (2.13) положим $k = 0$, а $i = 1,2,\ldots,n$.

3. Решим систему (2.13) и найдём величины $\Delta x_i^k, i = 1,2,\ldots,n$.

4. В формулах (2.12) положим $k = 0$ и вычислим компоненты следующего приближения $X^{k+1} = (x_1^{k+1}, x_2^{k+1},\ldots,x_n^{k+1})$.

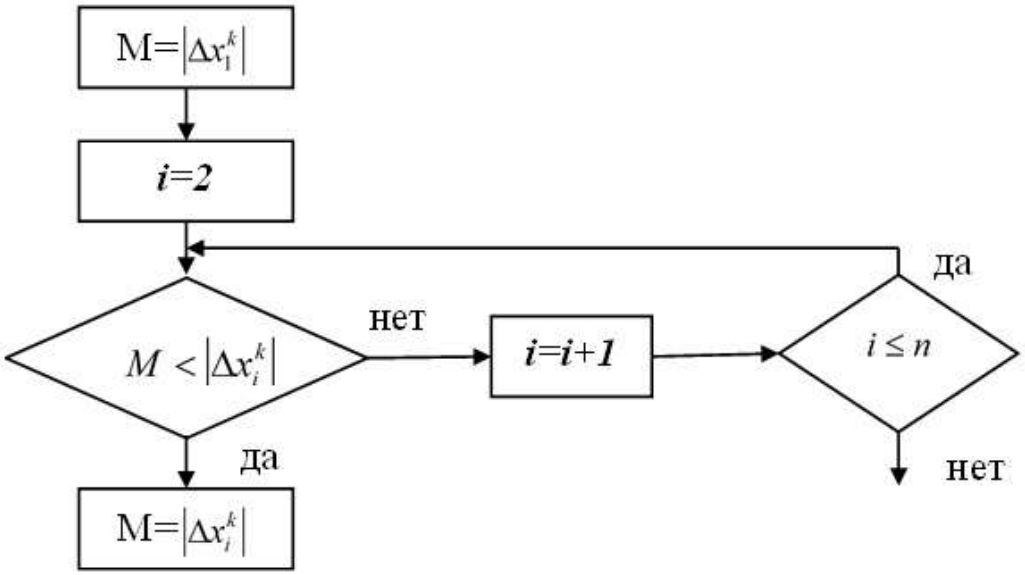
5. Проверим условие (2.7) на ε: $\max\left|\Delta x_i^k\right| \le \varepsilon, i = 1,2,\ldots,n.$ (См. алгоритм вычисления максимума нескольких величин.)

6. Если это условие выполняется, то заканчиваем вычисления, выбрав за приближённое решение системы (1.1) приближение $X^{k+1} = (x_1^{k+1}, x_2^{k+1},\ldots,x_n^{k+1})$. Если же это условие не выполняется, то перейдём к п.7.

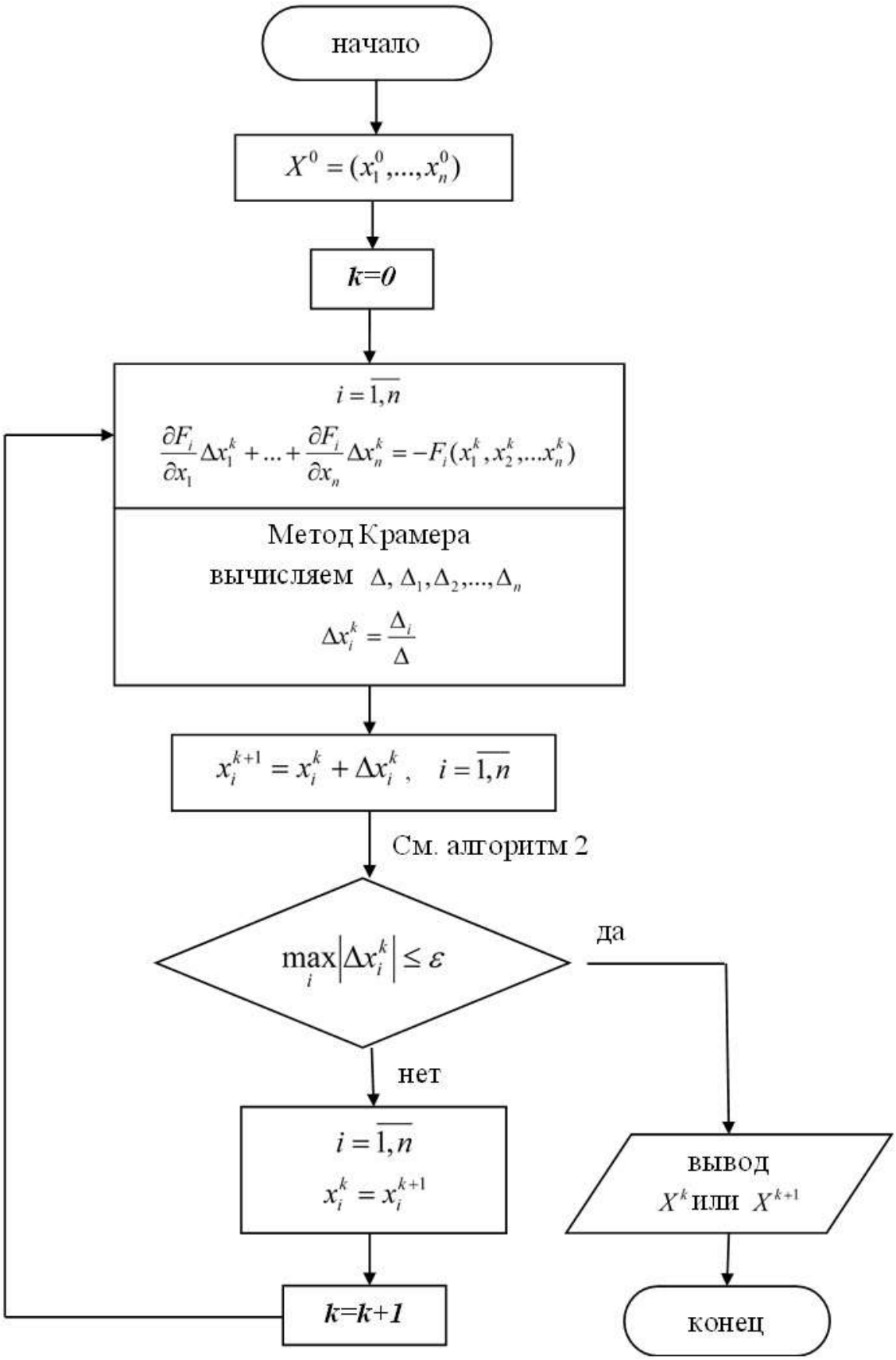
7. Положим $x_i^k = x_i^{k+1}$ для всех $i = 1,2,\ldots,n$.

8. Выполним п.3, положив $k = k + 1$.

Алгоритм вычисления максимума нескольких величин приведена ниже:



Блок-схема алгоритма метода Ньютона решения систем нелинейных уравнений приведена на следующем рисунке.


начало
$X^0 = (x_1^0, ..., x_n^0)$
k=0
$i = \overline{1,n}$
$\frac{\partial F_i}{\partial x_1}\Delta x_1^k + ... + \frac{\partial F_i}{\partial x_n}\Delta x_n^k = -F_i(x_1^k, x_2^k, ...x_n^k)$
Метод Крамера
вычисляем $\Delta, \Delta_1, \Delta_2, ..., \Delta_n$
$\Delta x_i^k = \frac{\Delta_i}{\Delta}$
$x_i^{k+1} = x_i^k + \Delta x_i^k, \quad i = \overline{1,n}$
См. алгоритм 2
$\max_i \left|\Delta x_i^k\right| \le \varepsilon$
да
нет
$i = \overline{1,n}$
$x_i^k = x_i^{k+1}$
k=k+1
вывод
X^k или X^{k+1}
конец

Рассмотрим использование метода Ньютона для решения системы двух уравнений.

Пример. Методом Ньютона с точностью до $\varepsilon = 0.001$ решить следующую систему нелинейных уравнений

$$\begin{cases} \sin(2x - y) - 1,2x - 0,4 = 0 \\ 0,8x^2 + 1,5y^2 = 1. \end{cases}, \tag{2.14}$$

Решение. Выберем нулевое приближение $X^0 = (x^0, y^0)$, принадлежащее области D.

Выберем в качестве нулевого приближения $x_0 = 0,4; y_0 = -0,75.$ Построим систему линейных алгебраических уравнений (2.3). Она будет иметь вид

$$\begin{aligned} \frac{\partial F_1}{\partial x}(x - x^0) + \frac{\partial F_1}{\partial y}(y - y^0) &= -F_1(x^0, y^0) \\ \frac{\partial F_2}{\partial x}(x - x^0) + \frac{\partial F_2}{\partial y}(y - y^0) &= -F_2(x^0, y^0) \end{aligned} \tag{2.15}$$

Обозначим

$$x - x^0 = \Delta x^0, y - y^0 = \Delta y^0 \tag{2.16}$$

Решим систему (2.15) относительно неизвестных $\Delta x^0, \Delta y^0$, например методом Крамера . Формулы Крамера запишем в виде

$$\Delta x^0 = \frac{\Delta_1}{\Delta}, \Delta y^0 = \frac{\Delta_2}{\Delta}, \tag{2.17}$$

где основной определитель системы (2.15)

$$\Delta = \begin{vmatrix} \dfrac{\partial F_1(x^0, y^0)}{\partial x} & \dfrac{\partial F_1(x^0, y^0)}{\partial y} \\ \dfrac{\partial F_2(x^0, y^0)}{\partial x} & \dfrac{\partial F_2(x^0, y^0)}{\partial y} \end{vmatrix} \neq 0 \tag{2.18}$$

а вспомогательные определители системы (2.15) имеют вид

$$\Delta_1 = \begin{vmatrix} -F_1(x^0, y^0) & \dfrac{\partial F_1(x^0, y^0)}{\partial y} \\ -F_2(x^0, y^0) & \dfrac{\partial F_2(x^0, y^0)}{\partial y} \end{vmatrix}$$

$$\Delta_2 = \begin{vmatrix} \dfrac{\partial F_1(x^0, y^0)}{\partial x} & -F_1(x^0, y^0) \\ \dfrac{\partial F_2(x^0, y^0)}{\partial x} & -F_2(x^0, y^0) \end{vmatrix}.$$

Элементы этих матриц и матрицы (2.18) для этой системы представляются в виде:

$$F_1(x,y)=\sin(2x-y)-1,2x-0,4;$$
$$F_2(x,y)=0,8x^2+1,5y^2-1;$$
$$\frac{\partial F_1}{\partial x}=2\cos(2x-y)-1,2;$$
$$\frac{\partial F_1}{\partial y}=-\cos(2x-y);$$
$$\frac{\partial F_2}{\partial x}=1,6x;$$
$$\frac{\partial F_2}{\partial y}=3y.$$

Найденные значения $\Delta x^0, \Delta y^0$ подставим в (2.16) и найдём компоненты $x^1=x^0+\Delta x^0, y^1=y^0+\Delta y^0$ первого приближения X^1 к решению системы (2.15).

Проверим условие

$$\max(|\Delta x^0|,|\Delta y^0|)\le\varepsilon, \qquad (2.19)$$

если это условие выполняется, то заканчиваем вычисления, приняв за приближённое решение системы (2.15) первое приближение, т. е. $X^1=(x^1,y^1)$. Если условие (2.19) не выполняется, то положим $x^0=x^1$, $y^0=y^1$ и построим новую систему линейных алгебраических уравнений (2.15). Решив её, найдём второе приближение $X^2=(x^2,y^2)$. Проверим его на ε. Если это условие будет выполняться, то за приближённое решение системы (2.15) выберем $X^2=(x^2,y^2)$. Если условие на ε не будет выполняться, положим $x^1=x^2$, $y^1=y^2$ и построим следующую систему (2.15) для нахождения $X^3=(x^3,y^3)$ и т. д.

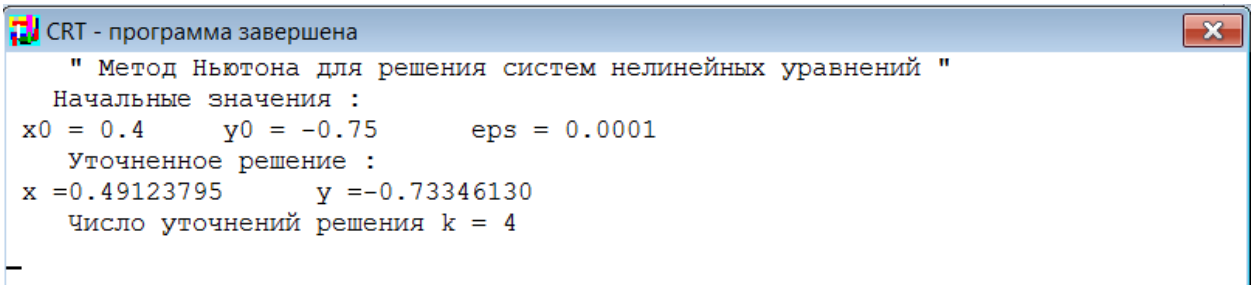
Программа решения задачи на языке Pascal.

```
Program MetodNewton;                    {метод Ньютона}
uses crt;
label vihod;
Var k:integer;
var eps,x0,y0,x1,y1,x,y,Dx,Dy,D,Hx,Hy,ro:real;
function fnF1(x:real;y:real):real;
{ Определение функции F1 }
begin
```

```
    fnF1:=sin(2*x-y)-1.2*x-0.4;
    end;
    function fnF2(x:real;y:real):real;
    { Определение функции F2 }
    begin
    fnF2:=0.8*sqr(x)+1.5*sqr(y)-1;
    end;
    {Определение частных производных функций}
    function fnF1x(x:real;y:real):real;
    begin
    fnF1x:=2*cos(2*x-y)-1.2;
    end;
    function fnF1y(x:real;y:real):real;
    begin
    fnF1y:=-cos(2*x-y);
    end;
    function fnF2x(x:real;y:real):real;
    begin
    fnF2x:=1.6*x;
    end;
    function fnF2y(x:real;y:real):real;
    begin
    fnF2y:=3*y;
    end;
    begin clrscr;
    writeln('    " Метод Ньютона для решения систем
нелинейных уравнений " ');
    x0:=0.4;      y0:=-0.75;       {Начальные приближения
решения}
    eps:=0.0001;                {Точность вычислений}
    x:=x0; y:=y0;
    k:=0;                     {Обнуление счетчика}
    repeat
    begin
    if (k>50) then        begin
    writeln('Число итераций велико! Проверьте x0 и y0');
    goto vihod;                end;
    {Описание определителей уточнения решения}
    Dx:=fnF1y(x,y)*fnF2(x,y)-fnF1(x,y)*fnF2y(x,y);
```

```
Dy:=fnF1(x,y)*fnF2x(x,y)-fnF1x(x,y)*fnF2(x,y);
D:=fnF1x(x,y)*fnF2y(x,y)-fnF1y(x,y)*fnF2x(x,y);
if (D=0) then     begin
writeln('Определитель D=0 в данной точке ');
goto vihod;       end;              {Выход при D=0}
{Вычисление значений уточняющих добавок к x и y }
Hx:=Dx/D;     Hy:=Dy/D;
{ Уточнение решений }
x1:= x+Hx;     y1:= y+Hy;
{ Расстояние между точками (x,y) и (x1,y1) }
ro:=sqrt(sqr(x1-x)+sqr(y1-y));
    x:=x1;        y:=y1;   {          Переобозначение
 координат }
k:=k+1;                    {Счетчик числа итераций}
end;
until (ro < eps);
vihod:
writeln('   Начальные значения :');
writeln(' x0 = ',x0,'   y0 = ',y0,'  eps = ',eps);
writeln('  Уточненное решение :');
writeln(' x =',x:8:8,'      y =',y:8:8);
writeln('    Число уточнений решения k = ',k);
end.
```

Результаты работы программы



Задания для самостоятельной работы.

Используя вышеуказанные методы, решить систему нелинейных уравнений с точностью до 0,001.

1. $\begin{cases}\sin(x+1)-y=1.2\\ 2x+\cos y=2\end{cases}$

2. $\begin{cases}\sin(y-1)+x=1.3\\ y-\sin(x+1)=0.8\end{cases}$

3. $\begin{cases}\cos(x-1)+y=0.5\\ x-\cos y=3\end{cases}$

4. $\begin{cases}2x-\cos(y+1)=0\\ y+\sin x=-0.4\end{cases}$

5. $\begin{cases}\sin x+2y=2\\ \cos(y-1)+x=0.7\end{cases}$

6. $\begin{cases}\cos(y+0.5)-x=2\\ \sin x-2y=1\end{cases}$

7. $\begin{cases}\cos x+y=1.5\\ 2x-\sin(y-0.5)=1\end{cases}$

8. $\begin{cases}\sin(y+2)-x=1.5\\ y+\cos(x-2)=0.5\end{cases}$

9. $\begin{cases}\sin(x+0.5)-y=1\\ \cos(y-2)+x=0\end{cases}$

10. $\begin{cases}\sin(x+1)-y=1\\ 2x+\cos y=2\end{cases}$

11. $\begin{cases}\sin(x-1)=1.3-y\\ x-\sin(y+1)=0.8\end{cases}$

12. $\begin{cases}\cos(x-1)+y=0.8\\ x-\cos y=2\end{cases}$

13. $\begin{cases}2y-\cos(x+1)=0\\ x+\sin y=-0.4\end{cases}$

14. $\begin{cases}\sin x+2y=1.6\\ \cos(y-1)+x=1\end{cases}$

15. $\begin{cases}\cos(x+0.5)-y=2\\ \sin y-2x=1\end{cases}$

16. $\begin{cases}\cos x+y=1.2\\ 2x-\sin(y-0.5)=2\end{cases}$

17. $\begin{cases}\sin(x+2)-y=1.5\\ x+\cos(y-2)=0.5\end{cases}$

18. $\begin{cases}\sin(x+0.5)-y=1.2\\ \cos(y-2)+x=0\end{cases}$

19. $\begin{cases}\sin(y+1)-x=1.2\\ 2y+\cos x=2\end{cases}$

20. $\begin{cases}\cos(x+0.5)+y=1\\ \sin y-2x=2\end{cases}$

21. $\begin{cases}\cos(y-1)+x=0.5\\ y-\cos x=3\end{cases}$

22. $\begin{cases}\sin(x+1)+y=1,5\\ x-\sin(y+1)=1\end{cases}$

23. $\begin{cases}\sin y+2x=2\\ \cos(x-1)+y=0.7\end{cases}$

24. $\begin{cases}\sin(y+1)-x=1\\ 2y+\cos x=2\end{cases}$

25. $\begin{cases}\cos y+x=1.5\\ 2y-\sin(x-0.5)=1\end{cases}$

26. $\begin{cases}\cos(y-1)+x=0.8\\ y-\cos x=2\end{cases}$

27. $\begin{cases}\sin(y+0.5)-x=1\\ \cos(x-2)+y=0\end{cases}$

28. $\begin{cases}\cos(y-1)+x=0.8\\ y-\cos x=2\end{cases}$

29. $\begin{cases}\sin(y-1)+x=0.8\\ \sin x-2y=1.6\end{cases}$

30. $\begin{cases}\cos(x-1)+y=1\\ \sin y+2x=1.6\end{cases}$

ГЛАВА 3. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ).

Цель работы. Знакомство с некоторыми приближёнными методами решения СЛАУ и их численной реализацией на ПК.

Предварительные замечания.

К решению систем линейных алгебраических уравнений сводится подавляющее большинство задач вычислительной математики. В настоящее время предложено колоссальное количество алгоритмов решения задач линейной алгебры, большинство из которых рассчитано на матрицы A специального вида (трехдиагональное, симметричные, ленточные, большие разреженные матрицы).

Все методы решения СЛАУ обычно разделяют на две большие группы. К первой группе относятся методы, которые принято называть точными. Эти методы позволяют для любых систем найти точные значения неизвестных после конечного числа арифметических операций, каждая из которых выполняется точно.

Ко второй группе относятся все методы, не являющиеся точными. Их называют итерационными, или численными, или приближёнными. Итерационные методы дают решение системы в виде предела последовательности некоторых векторов, построение которых осуществляется посредством единообразного процесса, называемого процессом итераций. Привлекательной чертой таких методов является их самоисправляемость и простота реализации на ПК.

Рассмотрим некоторые приближённые методы решения СЛАУ и построим алгоритмы их численной реализации.

3.1. Метод итерации.

Пусть СЛАУ задана в виде

$$\begin{aligned} x_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_1 \\ x_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_n \end{aligned} \qquad (1.1)$$

Эту систему можно записать в матричном виде

$$X = AX + B, \tag{1.2}$$

где $A = \begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ a_{n1} & a_{n2} & a_{nn} \end{pmatrix}$ - матрица коэффициентов при неизвестных в системе (1.1), $B = (b_1, b_2, \ldots, b_n)'$ - столбец свободных членов, $X = (x_1, x_2, \ldots, x_n)'$ - столбец неизвестных системы (1.1).

Будем считать, что система (1.1) определена и совместна, т.е. имеет единственное решение. Это решение обозначим

$$X^* = (x_1^*, x_2^*, \ldots, x_n^*). \tag{1.3}$$

Решим систему (1.1) методом итерации. Для этого выполним следующие действия.

Во-первых. Выберем нулевое приближение

$$X^0 = (x_1^0, x_2^0, \ldots, x_n^0) \tag{1.4}$$

к точному решению (1.3) системы (1.1). Компонентами нулевого приближения могут быть любые числа. Но удобнее за компоненты нулевого приближения взять либо нули $(x_i^0 = 0, i = 1,2,\ldots,n)$, либо свободные члены системы (1.1) $(x_i^0 = b_i, i = 1,2,\ldots,n)$.

Во-вторых. Компоненты нулевого приближения подставим в правую часть системы (1.1) и вычислим

$$\begin{aligned} x_1^1 &= a_{11}x_1^0 + a_{12}x_2^0 + \ldots + a_{1n}x_n^0 + b_1 \\ x_2^1 &= a_{21}x_1^0 + a_{22}x_2^0 + \ldots + a_{2n}x_n^0 + b_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^1 &= a_{n1}x_1^0 + a_{n2}x_2^0 + \ldots + a_{nn}x_n^0 + b_n \end{aligned} \tag{1.5}$$

Величины, стоящие слева в (1.5) являются компонентами первого приближения $X^1 = (x_1^1, x_2^1 \ldots, x_n^1)$. Действия, в результате которых получилось первое приближение, называются итерацией.

В-третьих. Проверим нулевое и первое приближения на ε

$$\begin{aligned} \left|x_1^1 - x_1^0\right| &\le \varepsilon \\ \left|x_2^1 - x_2^0\right| &\le \varepsilon \\ &\ldots\ldots\ldots \\ \left|x_n^1 - x_n^0\right| &\le \varepsilon \end{aligned} \tag{1.6}$$

Если все условия (1.6) выполняются, то за приближённое решение системы (1.1) выберем, либо $X^0=(x_1^0,x_2^0,...,x_n^0)$, либо $X^1=(x_1^1,x_2^1....,x_n^1)$.всё равно, т.к. они отличаются друг от друга не больше чем на ε и закончим вычисления. Если хотя бы одно из условий (1.6) не будет выполнено, то перейдём к следующему действию.

В-четвёртых. Выполним следующую итерацию, т.е. в правую часть системы (1.1) подставим компоненты первого приближения и вычислим компоненты второго приближения $X^2=(x_1^2,x_2^2,...,x_n^2)$, где

$$x_1^2=a_{11}x_1^1+a_{12}x_2^1+...+a_{1n}x_n^1+b_1$$
$$x_2^2=a_{21}x_1^1+a_{22}x_2^1+...+a_{2n}x_n^1+b_2$$
$$\text{……..……..……..……..……..……}$$
$$x_n^2=a_{n1}x_1^1+a_{n2}x_2^1+...+a_{nn}x_n^1+b_n$$

В-пятых. Проверим $X^1=(x_1^1,x_2^1....,x_n^1)$ и $X^2=(x_1^2,x_2^2,...,x_n^2)$ на ε, т.е. проверим условие (1.6) для этих приближений. Если все условия (1.6) будут выполнены, то за приближённое решение системы (1.1) выберем, либо $X^1=(x_1^1,x_2^1....,x_n^1)$, либо $X^2=(x_1^2,x_2^2,...,x_n^2)$ всё равно, т.к. они отличаются друг от друга не больше чем на ε. В противном случае будем строить следующую итерацию, подставив компоненты второго приближения в правую часть системы (1.1).

Итерации нужно строить до тех пор, пока два соседних приближения $X^k=(x_1^k,x_2^k,...,x_n^k)$ и $X^{k+1}=(x_1^{k+1},x_2^{k+1},...,x_n^{k+1})$ будут отличаться друг от друга не больше, чем на ε.

Рабочую формулу метода итерации решения системы (1.1) можно записать в виде

$$x_i^{k+1}=a_{i1}x_1^k+a_{i2}x_2^k+...+a_{in}x_n^k+b_i=\sum_{j=1}^{n}a_{ij}x_j^k+b_i,i=1,2,...,n;k=0,1,2,... \quad (1.7)$$

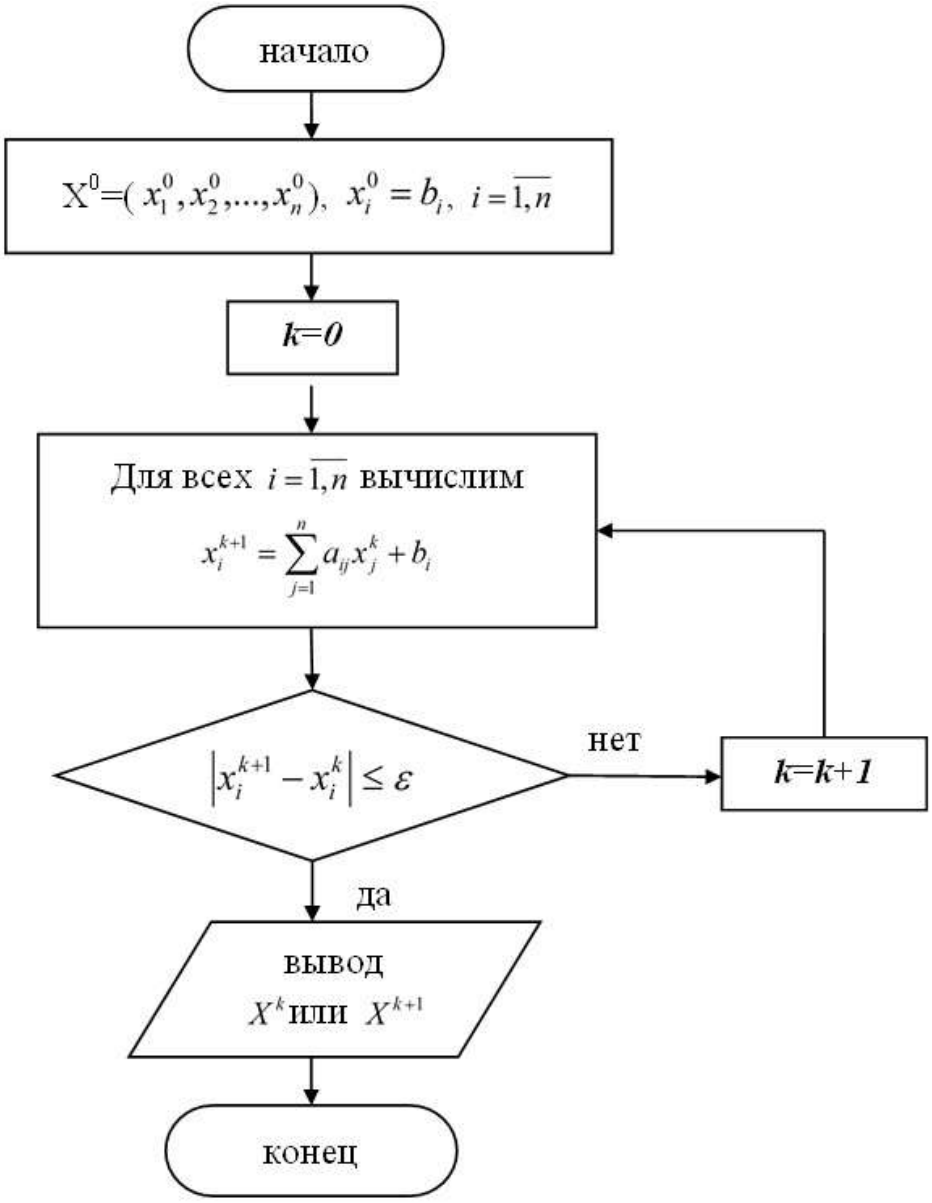
Алгоритм численной реализации формулы (1.7) может быть таким.

1. Выберем $X^0=(x_1^0,x_2^0,...,x_n^0)$, где $x_i^0=b_i,i=1,2,...,n$, если не оговорено особо.
2. Положим $k=0$.
3. Вычислим для всех $i=\overline{1,n}$ $x_i^{k+1}=\sum_{j=1}^{n}a_{ij}x_j^k+b_i$
4. Проверим условия $\left|x_i^{k+1}-x_i^k\right|\le\varepsilon$, $i=\overline{1,n}$.

5. Если все условия в п.4 будут выполнены, то за приближённое решение системы (1.1) выберем либо $X^k=(x_1^k,x_2^k,...,x_n^k)$, либо $X^{k+1}=(x_1^{k+1},x_2^{k+1},...,x_n^{k+1})$ и закончим вычисления. Если хотя бы одно условие в п.4 не будет выполнено, перейдём к п.6.

6. Положим $k=k+1$ и перейдём к п.3.

Блок-схема метода итерации представлено ниже:



3.2. Метод простой итерации.

Пусть система линейных алгебраических уравнений (СЛАУ) задана в виде

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n
\end{aligned}
\qquad (2.1)
$$

Чтобы систему (2.1) решить методом простой итерации, её сначала надо привести к виду

$$
\begin{aligned}
&x_1 = -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \ldots - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \\
&x_2 = -\frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \ldots - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&x_n = -\frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \ldots - \frac{a_{nn-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}}
\end{aligned}
\qquad (2.2)
$$

В системе (2.2) i-ое уравнение представляет собой i-ое уравнение системы (2.1), разрешённое относительно i–ой неизвестной ($i = \overline{1, n}$).

Метод решения системы (2.1), состоящий в сведении её к системе (2.2) с последующим решением системы (2.2) методом итерации, называется методом простой итерации для системы (2.1). Используя систему (2.2) и выбрав начальную точку $x^0 = \left\{x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\right\}$, можно построить итерационную последовательность точек n-мерного пространства (аналогично методу простой итерации для одномерного случая $x = \varphi(x)$).

Таким образом, рабочие формулы метода простой итерации решения системы (2.1) будут иметь вид

$$
\begin{aligned}
&x_1^{k+1} = -\frac{a_{12}}{a_{11}}x_2^k - \frac{a_{13}}{a_{11}}x_3^k - \ldots - \frac{a_{1n}}{a_{11}}x_n^k + \frac{b_1}{a_{11}} = -\sum_{j=2}^{n}\frac{a_{1j}}{a_{11}}x_j^k + \frac{b_1}{a_{11}} \\
&x_2^{k+1} = -\frac{a_{21}}{a_{22}}x_1^k - \frac{a_{23}}{a_{22}}x_3^k - \ldots - \frac{a_{2n}}{a_{22}}x_n^k + \frac{b_2}{a_{22}} = -\sum_{j=1, j\neq 2}^{n}\frac{a_{2j}}{a_{22}}x_j^k + \frac{b_2}{a_{22}} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&x_n^{k+1} = -\frac{a_{n1}}{a_{nn}}x_1^k - \frac{a_{n2}}{a_{nn}}x_2^k - \ldots - \frac{a_{nn-1}}{a_{nn}}x_{n-1}^k + \frac{b_n}{a_{nn}} = -\sum_{j=1}^{n-1}\frac{a_{nj}}{a_{nn}}x_j^k + \frac{b_n}{a_{nn}}, k = 0,1,2,\ldots
\end{aligned}
\qquad (2.3)
$$

Формулы (2.3) можно записать в виде

$$
x_i^{k+1} = -\sum_{j=1, j\neq i}^{n}\frac{a_{ij}}{a_{ii}}x_j^k + \frac{b_i}{a_{ii}}, i = 1,2,\ldots n, k = 0,1,2,\ldots \qquad (2.4)
$$

Алгоритм численной реализации метода простой итерации для системы (2.1) по формулам (2.4) может быть таким.

1. Выберем $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, $x_i^0 = \dfrac{b_i}{a_{ii}}, i = 1,2,\ldots,n,$ если не оговорено особо.

2. Положим $k = 0$.

3. Для всех $i = 1,2,\ldots,n$ вычислим

$$x_i^{k+1} = -\sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j^k + \frac{b_i}{a_{ii}}.$$

4. Проверим условия $\left|x_i^{k+1} - x_i^k\right| \le \varepsilon$, $i = 1,2,\ldots,n$.

5. Если все условия в п.4 будут выполнены, то за приближённое решение системы (2.1) выберем, либо $X^k = (x_1^k, x_2^k, \ldots, x_n^k)$, либо $X^{k+1} = (x_1^{k+1}, x_2^{k+1}, \ldots, x_n^{k+1})$ и закончим вычисления. Если хотя бы одно условие в п.4 не будет выполнено, перейдём к п.6.

6. Положим $k = k + 1$ и перейдём к п.3.

Достаточные условия сходимости метода простой итерации для системы (2.1) имеют вид:

$$\alpha = \max_{1\le j\le n} \sum_{j=1}^{n} \left|\alpha_{ij}\right| < 1, \tag{2.5}$$

т.е. максимальная из сумм модулей коэффициентов при неизвестных в правой части системы (2.2), взятых по строкам, должна быть меньше единицы;

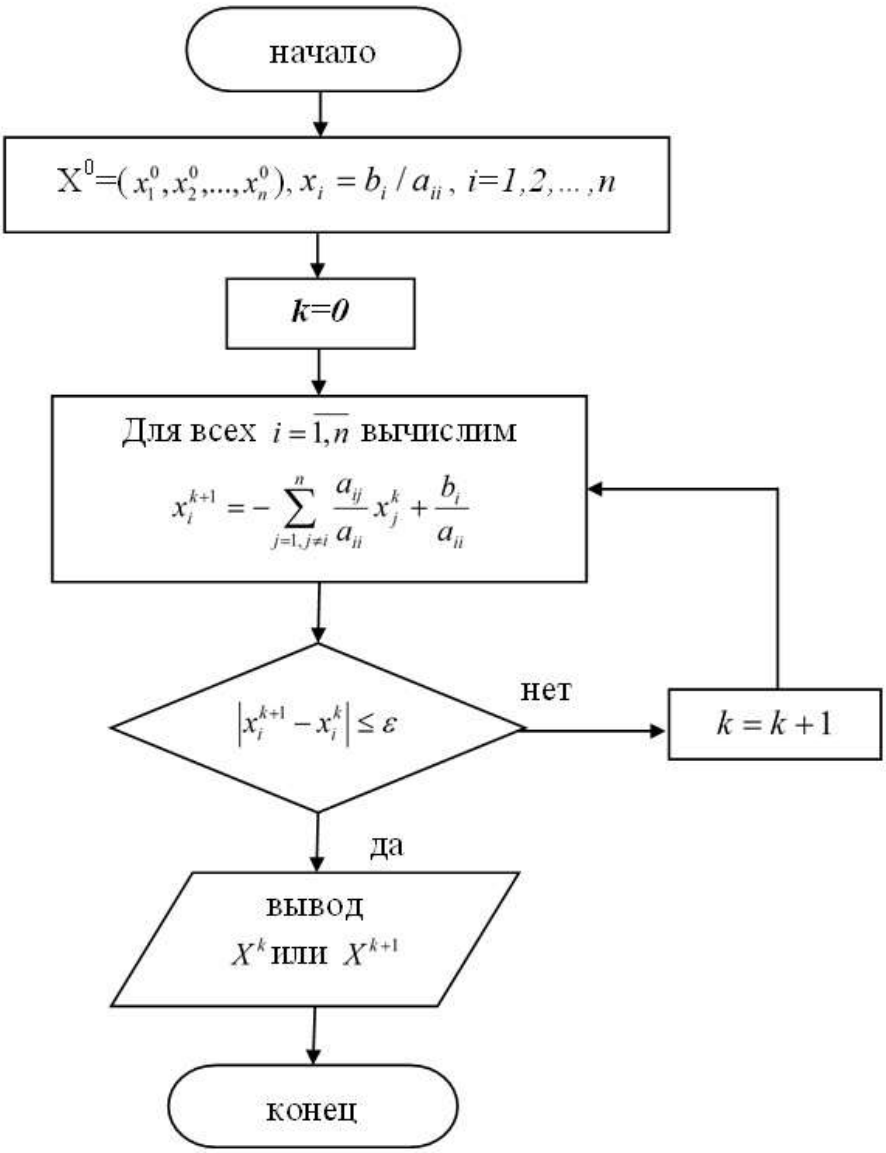
$$\alpha = \max_{1\le j\le n} \sum_{i=1}^{n} \left|\alpha_{ij}\right| < 1, \tag{2.6}$$

т.е. максимальная из сумм модулей коэффициентов при неизвестных в правой части системы (2.2), взятых по столбцам, должна быть меньше единицы;

$$\alpha = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{ij}^2} < 1, \tag{2.7}$$

т.е. сумма квадратов всех коэффициентов при неизвестных в правой части системы (2.2) должна быть меньше единицы.

Блок-схема метода простой итерации представлено на следующем рисунке.



Пример. Решить систему

$$\begin{cases}2,34x_1-4,21x_2-11,61x_3=14,41\\8,04x_1+5,22x_2+0,27x_3=-6,44\\3,92x_1-7,99x_2+8,37x_3=55,56\end{cases}$$

методом простой итераций с точностью $\varepsilon=10^{-4}$.

Получим сначала систему с преобладающими диагональными коэффициентами. Для этого первым уравнением возьмем второе, первое поставим на третье место, а в качестве второго уравнения возьмем сумму первого с третьим:

$$\begin{cases}8,04x_1+5,22x_2+0,27x_3=-6,44\\6,26x_1-12,20x_2-3,24x_3=69,97\\2,37x_1-4,21x_2-11,61x_3=14,41.\end{cases}$$

Разделим каждое уравнение на его диагональный коэффициент и выразим диагональное неизвестное:

$$\begin{cases} x_1 = -0,6492537x_2 - 0,033582x_3 - 0,800995 \\ x_2 = 0,5131147x_1 - 0,2655737x_3 - 5,7352459 \\ x_3 = 0,2015503x_1 - 0,3626184x_2 - 1,2411714. \end{cases}$$

Проверим одно из условий сходимости в соответствии с формулами (2.5) – (2.7). Воспользуемся формулой (2.6):

$$\alpha = \max_{1 \le j \le n} \sum_{i=1}^{n} \left| a_{ij} \right| < 1,$$

т.е. максимальная из сумм модулей коэффициентов при неизвестных в правой части системы, взятых по столбцам, должна быть меньше единицы. Максимальной суммой модулей коэффициентов по столбцам будет сумма модулей коэффициентов при x_2:

$$0,6492537 + 0,3626184 > 1.$$

Условие (2.6) не выполняется. Однако это не означает, что метод итераций применять нельзя. Проверим условие сходимости в пространстве с евклидовой метрикой согласно формуле (2.7):

$$\alpha = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij}^2} < 1,$$

$$0,6492537^2 + 0,033582^2 + 0,5131147^2 + 0,2655737^2 + 0,2015503^2 + 0,3626184^2 = 0,9291931 < 1.$$

Итак, итерационный процесс в евклидовом пространстве сходится, причём коэффициент сжатия $\alpha = \sqrt{0,9291931} \approx 0,96.$

Для достижения точности $\varepsilon = 10^{-4}$ приближения нужно находить до тех пор, пока не будет выполняться неравенство $\rho\left(x^{(k-1)}, x^{(k)}\right) \le \dfrac{\varepsilon(1-\alpha)}{\alpha}$, где $\rho\left(x^{(k-1)}, x^{(k)}\right)$ - расстояние между двумя соседними приближениями в смысле евклидовой метрики, причём $\varepsilon = 10^{-4}, \alpha = 0,96.$

Порядок выполнения самостоятельной работы

1. Привести систему к виду, в котором элементы главной диагонали превосходили бы остальные элементы строк.

2. Привести систему к виду, удобному для итераций, выразив диагональные неизвестные.

3. Проверить условие сходимости в соответствии с формулами (2.5) – (2.7). Определить коэффициент сжатия α.

4. Составить программу решения системы методом простой итераций с условием прекращения итераций.

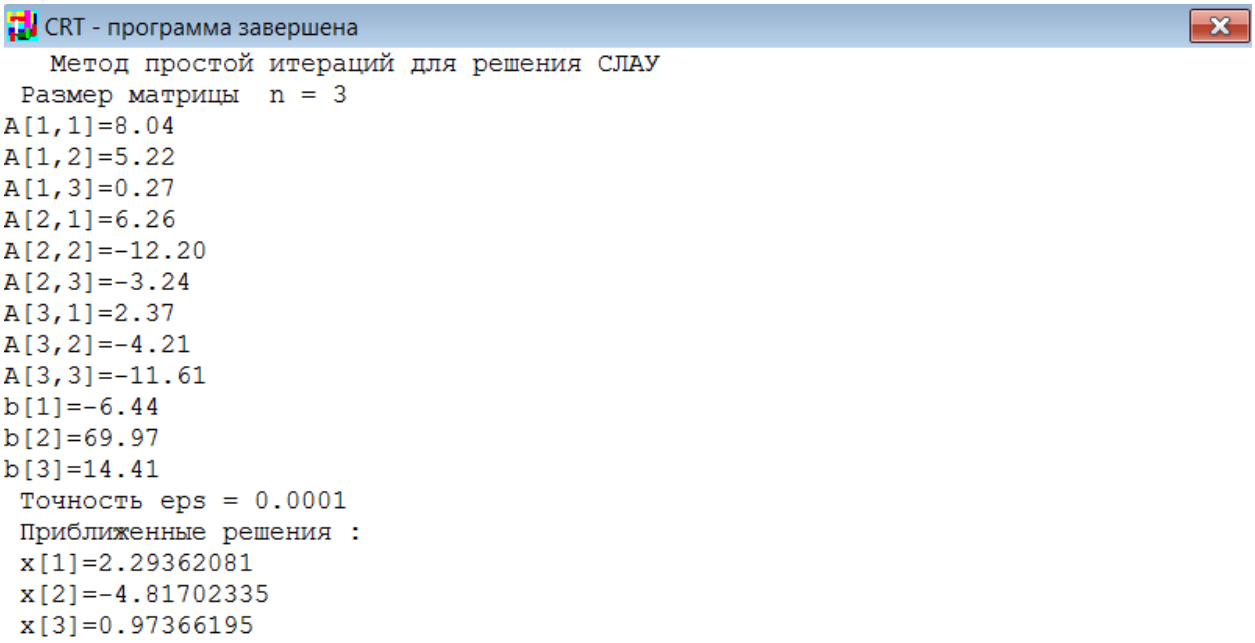
5. Округлить полученный результат в соответствии с заданной точностью.

Программа решения задачи на языке Pascal.

```
program prost_iter_slau;
    uses crt;
    var A: array [1..3,1..3] of real;
    b,x,otv: array [1..3] of real;
    i,j,n: byte;
    eps: real;
    pr: boolean;
begin clrscr;
writeln('Метод простой итераций для решения СЛАУ');
writeln(' Размер матрицы  n = ');
readln(n);
for i:=1 to n do
for j:=1 to n do
begin
write('A[',i,',',j,']=');
readln(A[i,j]);
end;
for i:=1 to n do
    if a[i,i]=0 then  begin
writeln(' ошибка ввода ');
exit;         end;
for i:=1 to n do
begin write('b[',i,']=');
readln(b[i]);
end;
for i:=1 to n do
begin
for j:=1 to n do
begin
if i=j then continue;
a[i,j]:=-a[i,j]/a[i,i];
end;
b[i]:=b[i]/a[i,i];
```

```
a[i,i]:=0;
end;
for i:=1 to n do
x[i]:=0;
write(' Точность eps = '); readln(eps);
repeat
for i:=1 to n do
begin
for j:=1 to n do
    otv[i]:=otv[i]+a[i,j]*x[j];
    otv[i]:=otv[i]+b[i];
end;
for i:=1 to n do
    if abs(otv[i]-x[i])<eps then pr:=true;
for i:=1 to n do
begin
x[i]:=otv[i];otv[i]:=0;
end;
until pr;
writeln(' Приближенные решения : ');
for i:=1 to n do
writeln(x[i]:8:8);
end.
```

Результат работы программы



3.3. Стационарный метод Зейделя.

Метод Зейделя решения СЛАУ отличается от метода итерации тем, что найдя какое-то приближение для i-той компоненты, мы сразу же используем его для отыскания следующих $(i+1)$, $(i+2)$, …, n-ой компонент. Такой подход позволяет обеспечить более высокую скорость сходимости метода Зейделя по сравнению с методом итерации.

Пусть СЛАУ задана в виде

$$\begin{aligned} x_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_1 \\ x_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_n \end{aligned} \qquad (3.1)$$

Пусть $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ - нулевое приближение к точному решению $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ системы (3.1). И пусть найдено k-ое приближение $X^\kappa = (x_1^\kappa, x_2^\kappa, \ldots, x_n^\kappa)$. Определим компоненты $(k+1)$-ого приближения $X^{\kappa+1} = (x_1^{\kappa+1}, x_2^{\kappa+1}, \ldots, x_n^{\kappa+1})$ по формулам

$$\begin{aligned} x_1^{k+1} &= a_{11}x_1^k + a_{12}x_2^k + \ldots + a_{1n}x_n^k + b_1 \\ x_2^{k+1} &= a_{21}x_1^{k+1} + a_{22}x_2^k + \ldots + a_{2n}x_n^k + b_2 \\ x_3^{k+1} &= a_{31}x_1^{k+1} + a_{32}x_2^{k+1} + a_{33}x_3^k + \ldots + a_{3n}x_n^k + b_3 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{k+1} &= a_{n1}x_1^{k+1} + a_{n2}x_2^{k+1} + \ldots + a_{nn-1}x_{n-1}^{k+1} + a_{nn}x_n^k + b_n \end{aligned} \qquad (3.2)$$

Формулы (3.2) можно записать в компактном виде

$$x_i^{k+1} = \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i}^{n} a_{ij}x_j^k + b_i \ , \ i = \overline{1,n}, \ k = 0,1,2,.... \qquad (3.3)$$

Алгоритм численной реализации метода Зейделя решения системы (3.1) по формулам (3.3) может быть таким.

1. Выберем $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, например, $x_i^0 = b_i$, $i = \overline{1,n}$

2. Положим $k = 0$.

3. Для всех $i = \overline{1,n}$ вычислим $x_i^{k+1} = \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i}^{n} a_{ij}x_j^k + b_i$.

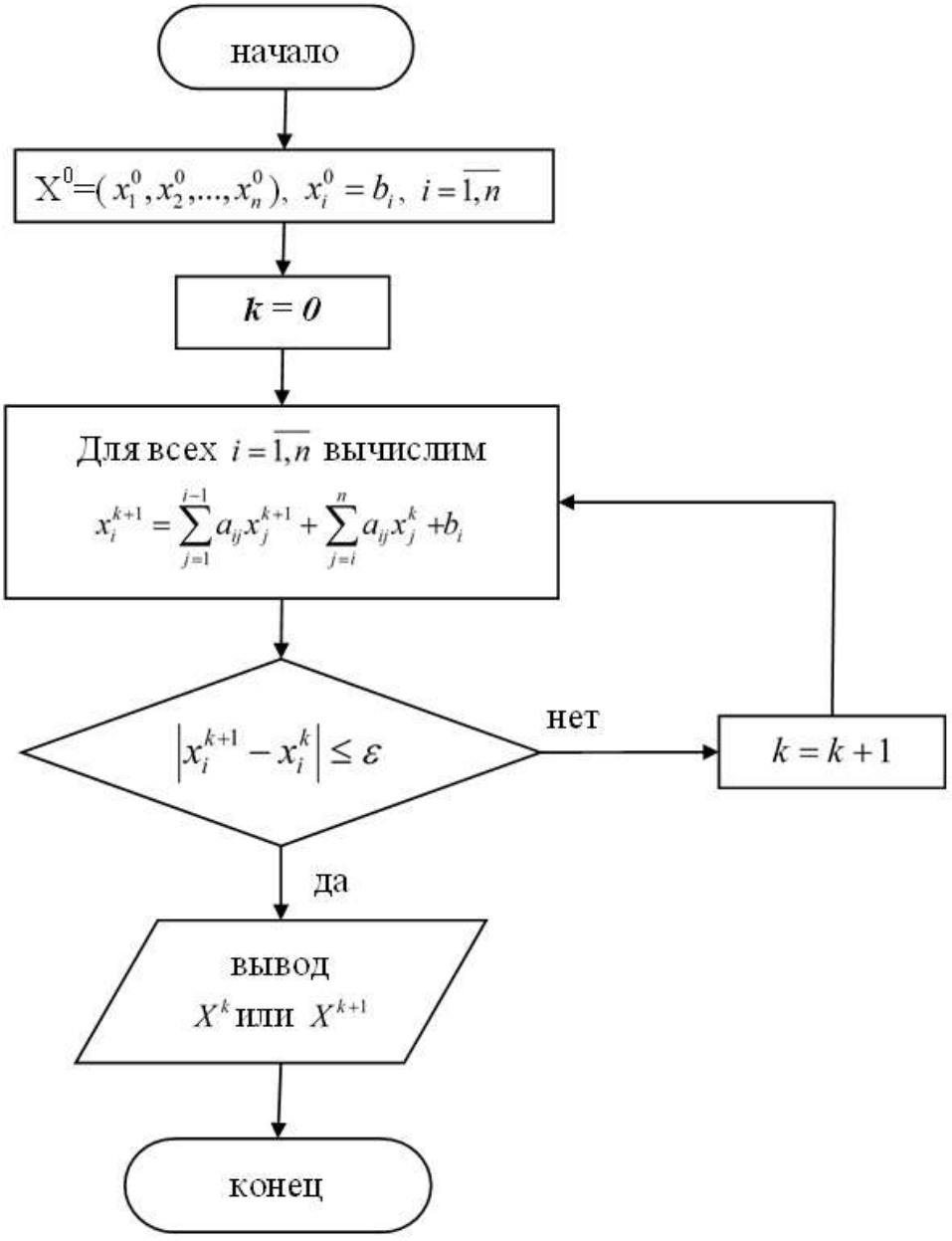
4. Для всех $i = \overline{1,n}$ проверим условия $\left|x_i^k - x_i^{k+1}\right| \le \varepsilon$.

5. Если все условия в п.4 будут выполнены, то за приближенное решение системы (3.1) выберем либо $X^\kappa = (x_1^\kappa, x_2^\kappa, \ldots, x_n^\kappa)$, либо

$X^{k+1} = (x_1^{k+1}, x_2^{k+1}, ..., x_n^{k+1})$ и закончим вычисления. Если хотя бы одно условие в п.4 не будет выполнено, перейдем к п.6.

6. Положим $k = k + 1$и перейдем к п.3.

Этот алгоритм можно записать геометрически.



Достаточное условие сходимости метода Зейделя для системы (3.1) имеет вид $\sum_{j=1}^{n} |a_{ij}| < 1,\ i = \overline{1,n}$. Преимущество метода Зейделя состоит в том, что он обычно обеспечивает более быструю сходимость, чем метод простой итерации.

Пример. Методом Зейделя решить систему линейных уравнений

$$\begin{cases} 3,7x_1 - 2,3x_2 + 4,5x_3 = 2,4 \\ 2,5x_1 + 4,7x_2 - 7,8x_3 = 3,5 \\ 1,6x_1 + 5,3x_2 + 1,3x_3 = 5,9 \end{cases}$$

преобразовав её к виду, удобному для итераций с точностью $\varepsilon = 10^{-4}$.

Приведем систему к виду, в котором элементы главной диагонали превосходили бы остальные элементы строк:

$$\begin{cases} 6,2x_1 + 2,4x_2 - 3,3x_3 = 5,9 & (I + II) \\ 1,6x_1 + 5,3x_2 + 1,3x_3 = 5,9 & (III) \\ 2,5x_1 + 4,7x_2 - 7,8x_3 = 3,5 & (II) \end{cases}$$

$$\begin{cases} x_1 = 0,38x_1 - 0,24x_2 + 0,33x_3 + 0,59 \\ x_2 = 0,16x_1 + 0,47x_2 - 0,13x_3 + 0,59 \\ x_3 = -0,25x_1 - 0,47x_2 + 0,22x_3 + 0,35. \end{cases}$$

Норма матрицы, состоящей из коэффициентов при неизвестных в правых частях уравнений, равна $\max\{0,95;0,76;0,94\} = 0,95 < 1$; значит процесс Зейделя сходится. Определим компоненты $(k+1)-$ого приближения $X^{(k+1)} = \left(x_1^{(k+1)}, x_2^{(k+1)}, ..., x_n^{(k+1)}\right)$ по формулам

$$\begin{cases} x_1^{(k+1)} = 0,38x_1^{(k)} - 0,24x_2^{(k)} + 0,33x_3^{(k)} + 0,59 \\ x_2^{(k+1)} = 0,16x_1^{(k+1)} + 0,47x_2^{(k)} - 0,13x_3^{(k)} + 0,59 \\ x_3^{(k+1)} = -0,25x_1^{(k+1)} - 0,47x_2^{(k+1)} + 0,22x_3^{(k)} + 0,35. \end{cases}$$

В качестве начального приближения возьмём следующий вектор: $X^{(0)} = (0,59;0,59;0,35)$.

Программа решения задачи на языке Pascal.

```
Program Zeydel;
uses crt;
const maxn=10;          {Максимальный порядок матрицы}
{-----Описываем типы для СЛАУ Start -----}
type matrix=array [1..maxn, 1..maxn] of real;
{Коэффициенты системы}
vector=array [1..maxn] of real;    {Свободные члены}
{----- Описываем типы для СЛАУ End -----}
var n,i:integer;              { порядок матрицы/счетчик}
a:matrix;                     { Коэффициенты системы }
b,x:vector;          { Свободные члены/Корни уравнения}
```

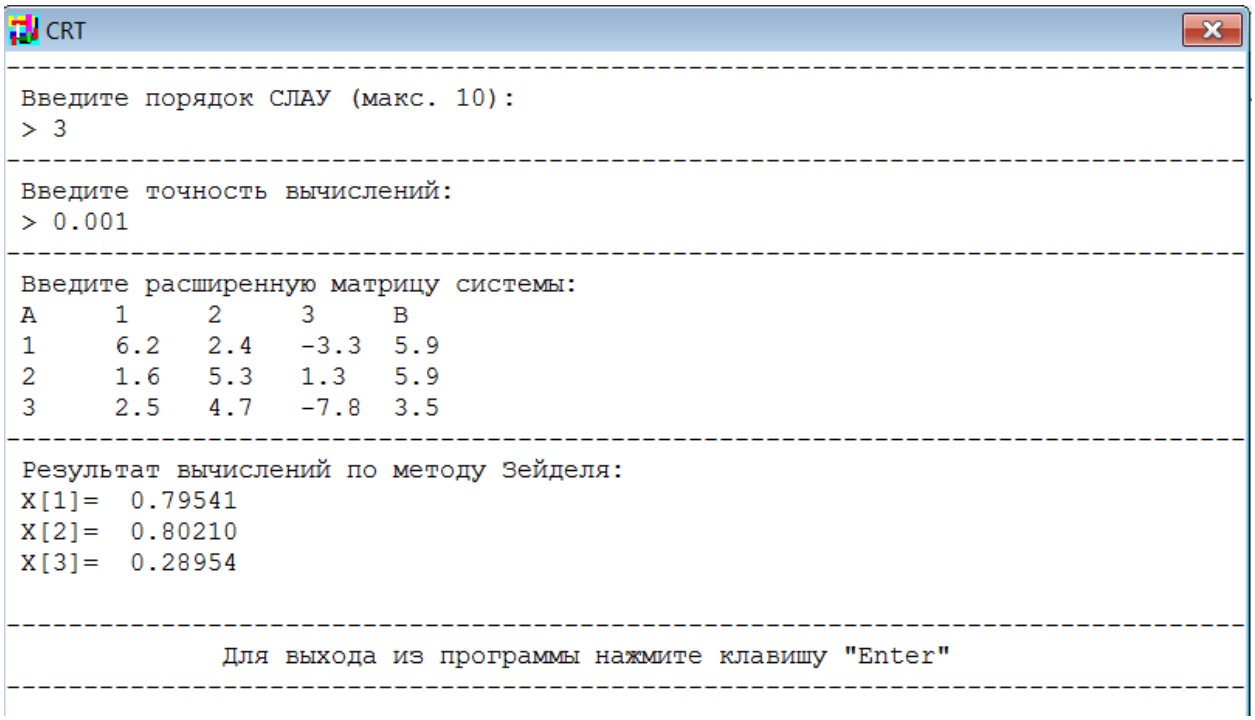
```
   eps:real;                                   {Точность}
   {----- Процедура ввода СЛАУ Start -----}
      Procedure ReadSystem( var a:matrix; var b:vector;
   n:integer);
   var i,j,yline:integer;
   Begin
   yLine:=WhereY;                          {Текущая строка}
   GotoXY(2,yLine);                    {Переводим курсор}
   Write('A');
   for i:=1 to n do
   Begin
      GotoXY((i*6+2),yLine);
      Write(i);                            {Выводим столбцы}
   GotoXY(1,(yLine+i));
   Write(i:2);                             {Выводим строки}
   end;
   GotoXY(((n+1)*6+2),yLine);
   Write('B');                    {Столбец свободных членов}
   for i:=1 to n do
   Begin
   for j:=1 to n do
   Begin
   GotoXY((j*6+2),(yLine+i));
   {Перемещаем курсор по столбцам}
   Read(A[i,j]);           {Вводим коэффициенты системы}
   end;
   GotoXY(((n+1)*6+2),(yLine+i));          {переводим
курсор на столбец свободных членов }
   Read(B[i]);                    {вводим свободные члены}
   end;
   end;
   {----- Процедура ввода СЛАУ End -----}
   {----- Процедура вывода результатов Start -----}
   Procedure WithResults(x:vector; n:integer);
   var i:integer;
   Begin
   for i:=1 to n do
   Writeln(' X[',i,']= ', X[i]:8:5);      Readln;
   end;
```

```
    {----- Процедура вывода результатов End -----}
    {----- Сердце метода Зейделя Start-----}
    Function Seidel(a:matrix; b:vector; var x:vector;
eps:real; n:integer):boolean;
    var i,j:integer;                          {Счетчики}
    sum1,sum2,sum,v,approach:real;
    Begin
    {-----Проверяем условие сходимости Start -----}
    for i:=1 to n do
    Begin
    Sum:=0;                        {Обнуляем значение суммы}
       for j:=1 to n do if (j <> i) then
    Sum:=Sum+Abs(A[i,j]);
       if (Sum >= Abs(A[i, i])) then
    Begin
    Seidel:=False;                          {Сходимости нет!}
    Exit;                                   {Выход}
    end;
    end;
    {----- Проверяем условие сходимости End -----}
    Repeat
    Approach:=0;      {Берем за начальное приближение}
    for i:=1 to n do
    Begin
    {Вычисляем суммы...}
    Sum1:=0;                { Обнуляем значение суммы }
    Sum2:=0;                { Обнуляем значение суммы }
    for j:=1 to (i-1) do
    Sum1:=Sum1+A[i,j]*X[j];
    for j:=i to n do
    Sum2:=Sum2+A[i,j]*X[j];
        {Вычисляем новое приближение...}
    V:=X[i];
    X[i]:=X[i]-(1/A[i,i])*(Sum1+Sum2-B[i]);
    if (Abs(V-X[i]) > Approach) then
    Approach:=Abs(V-X[i]);
    end;
    Until (Approach < Eps);        {Условие завершения}
    Seidel:=True;                      {СЛАУ решена...}
```

```
    end;
    {----- Сердце метода Зейделя End -----}
    Begin clrscr;
    Write('-----------------------------------------');
    Writeln(' Программа для решения СЛАУ.');
    Writeln(' Программа может решать СЛАУ до ',MaxN,'-го
порядка.');
    Writeln(' Метод решения СЛАУ: Метод Зейделя.');
    Writeln;
    Write('-----------------------------------------');
    GotoXY(17,WhereY);
    Writeln('Для запуска программы нажмите клавишу
"Enter"');
    Write('-----------------------------------------');
    Readln; clrscr;
    Write('-----------------------------------------');
    Writeln(' Введите порядок СЛАУ (макс. 10): ');
    Repeat
    Write(' > '); Readln(N);
    if not(N in [1..MaxN]) then
       Writeln('ОШИБКА:   Число   должно   принадлежать
    интервалу [0..',MaxN,']. Повторите ввод...');
    Until ((N > 0) and (N <= MaxN));
    Write('-----------------------------------------');
    Writeln(' Введите точность вычислений: ');
    Repeat
    Write(' > '); Readln(Eps);
    if not((0 < Eps) and (Eps < 1)) then
       Writeln(' ОШИБКА: Число должно принадлежать
    интервалу 0 < Eps < 1. Повторите ввод...');
    Until ((Eps > 0) and (Eps < 1));
    Write('-----------------------------------------');
    Writeln(' Введите расширенную матрицу системы: ');
ReadSystem(A,B,N);
    for i:=1 to n do X[i]:=0;
    Write('-----------------------------------------');
    if Seidel(A,B,X,Eps,N) then
    Begin
    Writeln('Результат вычислений по методу Зейделя:');
```

```
    WithResults(X,N);
    end
    else Writeln('Метод Зейделя не сходится для данной
системы!');
    Write('-----------------------------------------');
    GotoXY(15,WhereY);
    Writeln(' Для выхода из программы нажмите клавишу
"Enter"');
    Write('-----------------------------------------');
    Readln;
    end.
```

Результат работы программы



3.4. Нестационарный метод Зейделя.

Этот метод решения СЛАУ (3.1) обеспечивает еще более высокую скорость сходимости метода Зейделя.

Пусть каким-либо образом для системы (3.1) найдены компоненты k-ого приближения $X^{\kappa} = (x_1^{\kappa}, x_2^{\kappa}, ..., x_n^{\kappa})$ и $(k+1)$-ого приближения $X^{\kappa+1} = (x_1^{\kappa+1}, x_2^{\kappa+1}, ..., x_n^{\kappa+1})$.

Вычислим вектор поправки

$$\Delta X^{k+1} = X^{k+1} - X^{k} = (x_1^{k+1} - x_1^{k}, ..., x_n^{k+1} - x_n^{k}). \qquad (4.1)$$

Подсчитаем величины

$$\left|\Delta x_i^{k+1}\right| = \left|x_i^{k+1} - x_i^k\right|,\ i = \overline{1,n} \qquad (4.2)$$

Расположим величины $\left|\Delta x_i^{k+1}\right|,\ i = \overline{1,n}$ в порядке их убывания. В таком же порядке перепишем уравнения в системе (3.1) и неизвестные в этой системе.

И уже к «новой» системе применим стационарный метод Зейделя. При этом в первую очередь будут уточняться значения тех неизвестных, для которых погрешность в предыдущем приближении была наибольшей. Это и обеспечивает более высокую сходимость метода Зейделя.

Задания для самостоятельной работы.

Во всех заданиях требуется:

1. Итерационными методами решения СЛАУ решить с точностью 0,001 систему линейных уравнений, приведя ее к виду, удобному для итераций.

2. Составить программу численной реализации метода, согласно предложенному алгоритму.

3. Получить результаты вычислений.

4. Сравнить число итераций при определении приближенного решения с заданной точностью методом простой итерации и методом Зейделя.

№ 1. $\begin{cases} 2,7x_1 + 3,3x_2 + 1,3x_3 = 2,1; \\ 3,5x_1 - 1,7x_2 + 2,8x_3 = 1,7; \\ 4,1x_1 + 5,8x_2 - 1,7x_3 = 0,8. \end{cases}$

№ 2. $\begin{cases} 1,7x_1 + 2,8x_2 + 1,9x_3 = 0,7; \\ 2,1x_1 + 3,4x_2 + 1,8x_3 = 1,1; \\ 4,2x_1 - 1,7x_2 + 1,3x_3 = 2,8. \end{cases}$

№ 3. $\begin{cases} 3,1x_1 + 2,8x_2 + 1,9x_3 = 0,2; \\ 1,9x_1 + 3,1x_2 + 2,1x_3 = 2,1; \\ 7,5x_1 + 3,8x_2 + 4,8x_3 = 5,6. \end{cases}$

№ 4. $\begin{cases} 9,1x_1 + 5,6x_2 + 7,8x_3 = 9,8; \\ 3,8x_1 + 5,1x_2 + 2,8x_3 = 6,7; \\ 4,1x_1 + 5,7x_2 + 1,2x_3 = 5,8. \end{cases}$

№ 5. $\begin{cases} 3,3x_1 + 2,1x_2 + 2,8x_3 = 0,8; \\ 4,1x_1 + 3,7x_2 + 4,8x_3 = 5,7; \\ 2,7x_1 + 1,8x_2 + 1,x_3 = 3,2. \end{cases}$

№ 6. $\begin{cases} 7,6x_1 + 5,8x_2 + 4,7x_3 = 10,1; \\ 3,8x_1 + 4,1x_2 + 2,7x_3 = 9,7; \\ 2,9x_1 + 2,1x_2 + 3,8x_3 = 7,8. \end{cases}$

№ 7. $\begin{cases} 3,2x_1 - 2,5x_2 + 3,7x_3 = 6,5; \\ 0,5x_1 + 0,34x_2 + 1,7x_3 = -0,24; \\ 1,6x_1 + 2,3x_2 - 1,5x_3 = 4,3. \end{cases}$

№ 8. $\begin{cases} 5,4x_1 - 2,3x_2 + 3,4x_3 = -3,5; \\ 4,2x_1 + 1,7x_2 - 2,3x_3 = 2,7; \\ 3,x_1 + 2,4x_2 + 7,4x_3 = 1,9. \end{cases}$

№ 9. $\begin{cases} 3,6x_1 + 1,8x_2 - 4,7x_3 = 3,8; \\ 2,7x_1 - 3,6x_2 + 1,9x_3 = 0,4; \\ 1,5x_1 + 4,5x_2 + 3,3x_3 = -1,6. \end{cases}$

№ 10. $\begin{cases} 5,6x_1 + 2,7x_2 - 1,7x_3 = 1,9; \\ 3,4x_1 - 3,6x_2 - 6,7x_3 = -2,4; \\ 0,8x_1 + 1,3x_2 + 3,7x_3 = 1,2. \end{cases}$

№ 11. $\begin{cases} 2,7x_1 + 0,9x_2 - 1,5x_3 = 3,5; \\ 4,5x_1 - 2,8x_2 + 6,7x_3 = 2,6; \\ 5,1x_1 + 3,7x_2 - 1,4x_3 = -0,14. \end{cases}$

№ 12. $\begin{cases} 4,5x_1 - 3,5x_2 + 7,4x_3 = 2,5; \\ 3,1x_1 - 0,6x_2 - 2,3x_3 = -1,5; \\ 0,8x_1 + 7,4x_2 - 0,5x_3 = 6,4. \end{cases}$

№ 13. $\begin{cases} 3,8x_1 + 6,7x_2 - 1,2x_3 = 5,2; \\ 6,4x_1 + 1,3x_2 - 2,7x_3 = 3,8; \\ 2,4x_1 - 4,5x_2 + 3,5x_3 = -0,6. \end{cases}$

№ 14. $\begin{cases} 5,4x_1 - 6,2x_2 - 0,5x_3 = 0,52; \\ 3,4x_1 + 2,3x_2 + 0,8x_3 = -0,8; \\ 2,4x_1 - 1,1x_2 + 3,8x_3 = 1,8. \end{cases}$

№ 15. $\begin{cases} 7,8x_1 + 5,3x_2 + 4,8x_3 = 1,8; \\ 3,3x_1 + 1,1x_2 + 1,8x_3 = 2,3; \\ 4,5x_1 + 3,3x_2 + 2,8x_3 = 3,4. \end{cases}$

№ 16. $\begin{cases} 3,8x_1 + 4,1x_2 - 2,3x_3 = 4,8; \\ -2,1x_1 + 3,9x_2 - 5,8x_3 = 3,3; \\ 1,8x_1 + 1,1x_2 - 2,1x_3 = 5,8. \end{cases}$

№ 17. $\begin{cases} 1,7x_1 - 2,2x_2 + 3,0x_3 = 1,8; \\ 2,1x_1 + 1,9x_2 - 2,3x_3 = 2,8; \\ 4,2x_1 + 3,9x_2 - 3,1x_3 = 5,1. \end{cases}$

№ 18. $\begin{cases} 2,8x_1 + 3,8x_2 - 3,2x_3 = 4,5; \\ 2,5x_1 - 2,8x_2 + 3,3x_3 = 7,1; \\ 6,5x_1 - 7,1x_2 + 4,8x_3 = 6,3. \end{cases}$

№ 19. $\begin{cases} 3,3x_1 + 3,7x_2 + 4,2x_3 = 5,8; \\ 2,7x_1 + 2,3x_2 - 2,9x_3 = 6,1; \\ 4,1x_1 + 4,8x_2 - 5,0x_3 = 7,0. \end{cases}$

№ 20. $\begin{cases} 7,1x_1 + 6,8x_2 + 6,1x_3 = 7,0; \\ 5,0x_1 + 4,8x_2 + 5,3x_3 = 6,1; \\ 8,2x_1 + 7,8x_2 + 7,1x_3 = 5,8. \end{cases}$

№ 21. $\begin{cases} 3,7x_1 + 3,1x_2 + 4,0x_3 = 5,0; \\ 4,1x_1 + 4,5x_2 - 4,8x_3 = 4,9; \\ -2,1x_1 - 3,7x_2 + 1,8x_3 = 2,7. \end{cases}$

№ 22. $\begin{cases} 4,1x_1 + 5,2x_2 0 - 5,8x_3 = 7,0; \\ 3,8x_1 - 3,1x_2 + 4,0x_3 = 5,3; \\ 7,8x_1 + 5,3x_2 - 6,3x_3 = 5,8. \end{cases}$

№ 23. $\begin{cases} 3,7x_1 - 2,3x_2 + 4,3x_3 = 2,4; \\ 2,5x_1 + 4,7x_2 - 7,8x_3 = 3,5; \\ 1,6x_1 + 5,3x_2 + 1,3x_3 = -2,4. \end{cases}$

№ 24. $\begin{cases} 6,3x_1 + 5,2x_2 - 0,6x_3 = 1,5; \\ 3,4x_1 - 2,3x_2 + 3,4x_3 = 2,7; \\ 0,8x_1 + 1,4x_2 + 3,5x_3 = -2,3. \end{cases}$

№ 25. $\begin{cases} 1{,}5x_1 + 2{,}3x_2 - 3{,}7x_3 = 4{,}5; \\ 2{,}8x_1 + 3{,}4x_2 + 5{,}8x_3 = -3{,}2; \\ 1{,}2x_1 + 7{,}3x_2 - 2{,}3x_3 = 5{,}6. \end{cases}$

№ 26. $\begin{cases} 0{,}9x_1 + 2{,}7x_2 - 3{,}8x_3 = 2{,}4; \\ 2{,}5x_1 + 5{,}8x_2 - 0{,}5x_3 = 3{,}5; \\ 4{,}5x_1 - 2{,}1x_2 + 3{,}2x_3 = -1{,}2. \end{cases}$

№ 27. $\begin{cases} 2{,}4x_1 + 2{,}5x_2 - 2{,}9x_3 = 4{,}5; \\ 0{,}8x_1 + 3{,}5x_2 - 1{,}4x_3 = 3{,}2; \\ 1{,}5x_1 - 2{,}3x_2 + 8{,}6x_3 = -5{,}5. \end{cases}$

№ 28. $\begin{cases} 5{,}4x_1 - 2{,}4x_2 + 3{,}8x_3 = 5{,}5; \\ 2{,}5x_1 + 6{,}8x_2 - 1{,}1x_3 = 4{,}3; \\ 2{,}7x_1 - 0{,}6x_2 + 1{,}5x_3 = -3{,}5. \end{cases}$

№ 29. $\begin{cases} 2{,}4x_1 + 3{,}7x_2 - 8{,}3x_3 = 2{,}3; \\ 1{,}8x_1 + 4{,}3x_2 + 1{,}2x_3 = -1{,}2; \\ 3{,}4x_1 - 2{,}3x_2 + 5{,}2x_3 = 3{,}5. \end{cases}$

№ 30. $\begin{cases} 3{,}2x_1 - 11{,}5x_2 + 3{,}8x_3 = 2{,}8; \\ 0{,}8x_1 + 1{,}3x_2 - 6{,}4x_3 = -6{,}5; \\ 2{,}4x_1 + 7{,}2x_2 - 1{,}2x_3 = 4{,}5. \end{cases}$

Литература

1. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Лаборатория Базовых Знаний, 2001.
2. Боглаев Ю.П. Вычислительная математика и программирование. – М.: Высшая школа, 1990.
3. Воробьева Г.Н. Данилова А.Н. Практикум по вычислительной математике. –М.: Высшая школа, 1990.
4. Демидович В.П. Марон И.А. Основы вычислительной математики. – М.: Физ.мат.литература, 1960.
5. Исроилов М., Хисоблаш методлари. – Тошкент: Узбекистон, 2003
6. Киреев В.И., Пантелеев А.В. Численные методы в примерах и задачах. – М.: Высшая школа, 2008.
7. Колдаев В.Д. Численные методы и программирование. – М.: ИД «ФОРУМ» ИНФРА-М, 2009.
8. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах. –М.: Наука, 1972.
9. Под. редакцией П. И. Монастырного. Сборник задач по методам вычислений (учебное пособие). – Минск: БГУ. 1983г
10. Отаров А.О., Алланазаров Ж.П. Есаплаў усыллары. I бөлим. – Нөкис: Билим, 2001.
11. Отаров А.О., Алланазаров Ж.П. Есаплаў усыллары. II бөлим. – Нөкис: Билим, 2006.
12. Отаров А.О. Сызықлы алгебралық теңлемелериниң системаларын итерациялық усыл менен шешиў. Нөкис : Билим. 1997.

Содержание

Аламинов Муратбай Хайтбаевич– кандидат физико-математических наук, доцент кафедры «Методика преподавания информатики» Нукусского государственного педагогического института имени Ажинияза.

Бектурсынова Дилнура Пулатовна– стажёр-преподаватель кафедры «Методика преподавания информатики» Нукусского государственного педагогического института имени Ажинияза.

Уразымбетова Эльзура Пулатовна– ассистент кафедры «Прикладная математика» Каракалпакского государственного университета имени Бердаха.

ЧИСЛЕННЫЕ МЕТОДЫ

(Учебно-методическое пособие)

Главный редактор: К.М.Кощанов
Тех.редактор: Х.К. Шамуратова
Корректор: З.Б.Балтабаева
Оператор: Н.Нысанбаев

Редакционно-издательский отдел НГПИ им. Ажинияза
Отпечатано в типографии НГПИ им. Ажинияза 2020 год.
Заказ №0164 Тираж 50. Формат 60x80. Объем 4,0 п.л.
Республика Каракалпакстан, город Нукус, 230105.
ул. П.Сейтов б/н. №11-3084 Реестр.