МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

(ФГБОУ ВО «ВятГУ»)

Факультет автоматики и вычислительной техники

Кафедра ЭВМ

Отчёт

Лабораторная работа № 2 по дисциплине

«Вычислительная математика»

«ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ»

Вариант 7

Выполнил студент группы ИВТб-2301-04-00 / Жеребцов К. А./

Проверил преподаватель / Исупов К. С./

Киров 2021

**Задание**

1. Решить систему линейных уравнений 4-го порядка методом Гаусса с точностью е=0,001.

Уравнения системы:

2,00\*x1-1,00\*x2+1,00\*x3-1,00\*x4=1,00

2,00\*x1-1,00\*x2-3,00\*x4=2,00

3,00\*x1-1,00\*x3+1,00\*x4=-3,00

2,00\*x1+2,00\*x2-2,00\*x3+5,00\*x4=-6,00

2. Решить систему линейных уравнений 4-го порядка с точностью е=0,0001:

- методом Зейделя.

Уравнения системы:

x1=0,17\*x1+0,27\*x2-0,13\*x3-0,11\*x4-1,42

x2=0,13\*x1-0,12\*x2+0,09\*x3-0,06\*x4+0,48

x3=0,11\*x1+0,05\*x2-0,02\*x3+0,12\*x4-2,34

x4=0,13\*x1+0,18\*x2+0,24\*x3+0,43\*x4+0,72

3. Решить систему линейных уравнений 3-го порядка методом обратной матрицы с точностью е=0,001.

Уравнения системы:

2\*x1+4\*x2+3\*x3=13

x1+x2+4\*x3=17

2\*x1+3\*x2+4\*x3=12

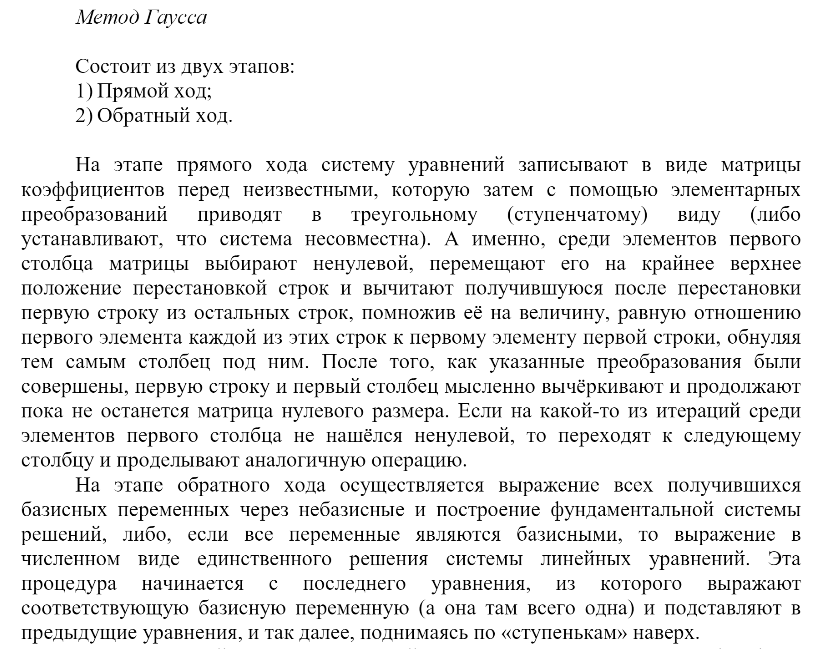
4. Решить систему нелинейных уравнений 2-го порядка методом Ньютона с точностью е=0,001.

Уравнения системы:

x-ln(-0,9-y)=0

y-tg(x+0,8)=0

5. Проверить результаты с помощью системы Mathcad.



Листинг:

program MetodGaussa;

Uses Crt;

const

n = 4;

var

a, a1: array[1..10, 1..10] of real;

aa, bb: array[1..10] of real;

b, b1: array[1..10] of real;

x: array[1..10] of real;

i, j, k: integer;

r, g: real;

function ShowMatrix(n: Integer): real;

var

i, j: integer;

begin

writeln('Исходная система:');

writeln('2,00\*x1-1,00\*x2+1,00\*x3-1,00\*x4=1,00');

writeln('2,00\*x1-1,00\*x2-3,00\*x4=2,00');

writeln('3,00\*x1-1,00\*x3+1,00\*x4=-3,00');

writeln('2,00\*x1+2,00\*x2-2,00\*x3+5,00\*x4=-6,00');

writeln('Eps = 0.001');

writeln('------------------');

WriteLn('Матрица A:');

for i := 1 to n do

begin

for j := 1 to n do

begin

write(' ', a1[i, j]);

if j = n then

writeln

end;

end;

WriteLn('Матрица B:');

for i := 1 to n do

writeln(' ', b[i]);

writeln('------------------');

end;

procedure switch(aaa, bbb: integer);

var

i, j: byte;

begin

i := aaa;

for j := 1 to n do

begin

aa[j] := a[i, j];

a[i, j] := a[bbb, j];

a[bbb, j] := aa[j];

end;

bb[bbb] := b[i];

b[i] := b[bbb];

b[bbb] := bb[bbb];

end;

function Metod: real;

var

k, j, i, z, val: integer;

begin

writeln('Прямой ход');

for k := 1 to n do { прямой ход Гаусса }

begin

z := k;

while (z < n) and (a[z, k] = 0) do inc(z);

switch(k, z);

for j := k + 1 to n do

begin

// write(' ',a[j-1,k+1]);

r := a[j, k] / a[k, k];

for i := k to n do

begin

a[j, i] := a[j, i] - r \* a[k, i];

write(' ', a[j, i]);

end;

b[j] := b[j] - r \* b[k];

writeln('|', b[j]);

end;

if k < n then

writeln('------------------');

end;

for k := n downto 1 do { обратный ход Гаусса }

begin

r := 0;

for j := k + 1 to n do

begin

g := a[k, j] \* x[j];

r := r + g;

end;

x[k] := (b[k] - r) / a[k, k];

end;

writeln('Корни:');

for i := 1 to n do

writeln('x[', i, ']=', x[i]:0:4, ' ');

end;

function Check: real;

var

i, j: integer;

begin

clrscr;

writeln('Проверка в соответсвии с матрицой B: ');

for i := 1 to n do

begin

for j := 1 to n do

b1[i] := b1[i] + a1[i, j] \* x[j];

writeln(b1[i]:4:4, ' ');

end;

readln;

end;

begin

WRITELN('МЕТОД ГАУССА');

writeln('------------------');

a1[1, 1] := 2; a1[1, 2] := -1; a1[1, 3] := 1; a1[1, 4] := -1; b[1] := 1;

a1[2, 1] := 2; a1[2, 2] := -1; a1[2, 3] := 0; a1[2, 4] := -3; b[2] := 2;

a1[3, 1] := 3; a1[3, 2] := 0; a1[3, 3] := -1; a1[3, 4] := 1; b[3] := -3;

a1[4, 1] := 2; a1[4, 2] := 2; a1[4, 3] := -2; a1[4, 4] := 5; b[4] := -6;

for i := 1 to n do

begin

for j := 1 to n do

begin

a[i, j] := a1[i, j];

end;

end;

ShowMatrix(n);

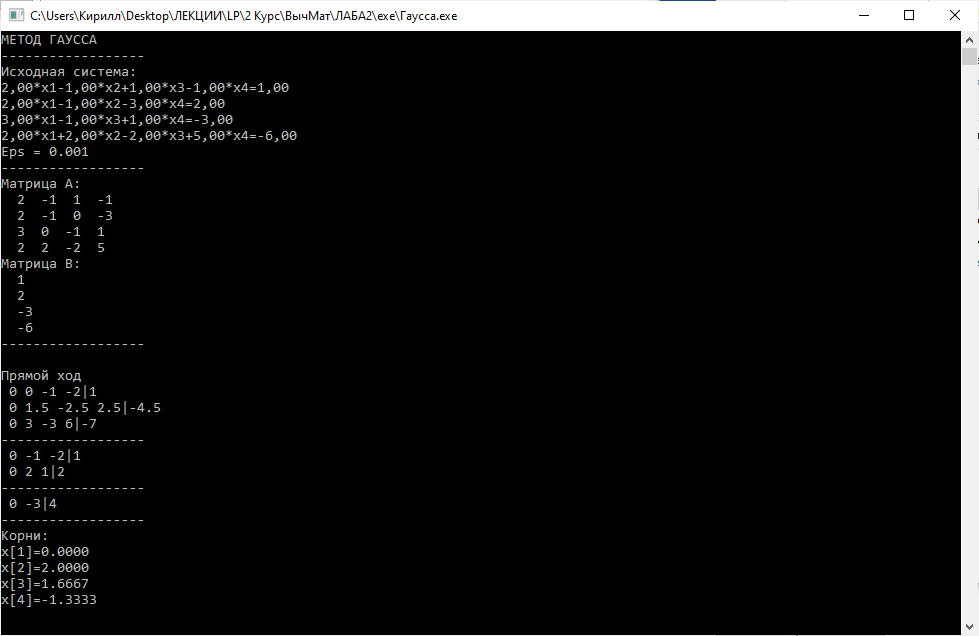
writeln;

Metod;

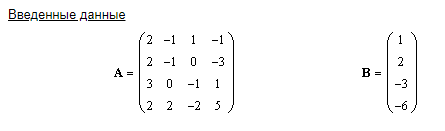
Readln;

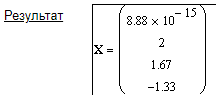
end.

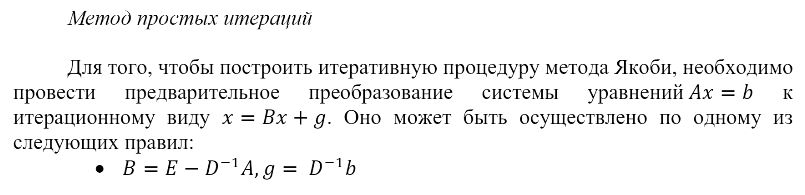
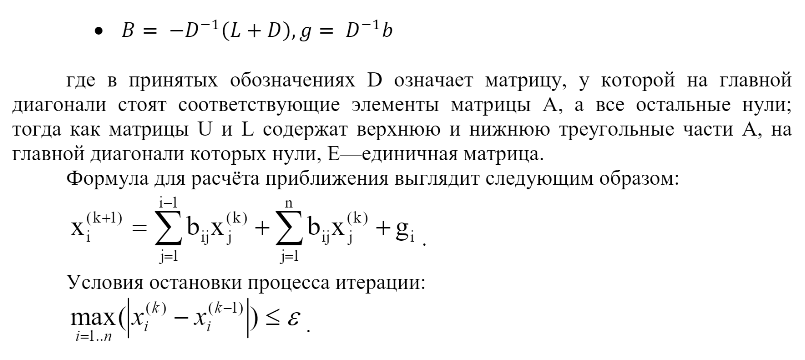
Экранные формы:



Проверка:





Листинг:

Program Zeidel;

const

eps = 0.0001;

type

mas = array [1..4] of real;

matr = array [1..4, 1..4] of real;

var

x1, x0: mas;

m: matr;

i, j, k: integer;

p, q: real;

o: real;

smax, smax1: real;

s: mas;

procedure norm;

var

i, j: integer;

begin

m[1,1]:=0.17; m[1,2]:=0.27; m[1,3]:=-0.13; m[1,4]:=-0.11;

m[2,1]:=0.13; m[2,2]:=-0.12; m[2,3]:=0.09; m[2,4]:=-0.06;

m[3,1]:=0.11; m[3,2]:=0.05; m[3,3]:=-0.02; m[3,4]:=0.12;

m[4,1]:=0.13; m[4,2]:=0.18; m[4,3]:=0.24; m[4,4]:=0.43;

writeln('Проверка 1 нормы');

for i := 1 to 4 do

begin

for j := 1 to 4 do

begin

s[i] := s[i] + abs(m[i, j]);

end;

writeln(' ',s[i]);

end;

smax := max(s[1], s[2], s[3], s[4]);

if smax < 1 then

begin

writeln('1 норма = ', smax);

writeln('Условие сходимости: ||a||1 < 1');

exit

end;

writeln('Проверка 2 нормы');

for j := 1 to 4 do

begin

for i := 1 to 4 do

begin

s[i] := s[i] + abs(m[i, j]);

end;

end;

smax := max(s[1], s[2], s[3], s[4]);

if smax < 1 then

begin

writeln('2 норма = ', smax);

writeln('Условие сходимости: ||a||2 < 1');

exit

end;

writeln('Проверка 3 нормы');

for j := 1 to 4 do

begin

for i := 1 to 4 do

begin

smax := smax + abs(m[i, j] \* m[i, j]);

end;

end;

smax1 := sqrt(smax);

if smax1 < 1 then

begin

writeln('3 норма = ', smax);

writeln('Условие сходимости: ||a||3 < 1');

exit

end;

end;

begin

for i := 1 to 4 do

s[i] := 0;

Writeln('МЕТОД ЗЕЙДЕЛЯ');

writeln('Исходная система:');

writeln('x1=0,17\*x1+0,27\*x2-0,13\*x3-0,11\*x4-1,42 ');

writeln('x2=0,13\*x1-0,12\*x2+0,09\*x3-0,06\*x4+0,48 ');

writeln('x3=0,11\*x1+0,05\*x2-0,02\*x3+0,12\*x4-2,34');

writeln('x4=0,13\*x1+0,18\*x2+0,24\*x3+0,43\*x4+0,72');

writeln('Eps = 0,0001');

Writeln('-----------------');

norm;

Writeln('-----------------');

writeln(' X1 | X`1 | X2 | X`2 | X3 | X`3 | X4 | X`4 | q ');

repeat

for i := 1 to 4 do

x0[i] := x1[i];

x1[1] := -1.42 - 0.17 \* x0[1] + 0.27 \* x0[2] - 0.13 \* x0[3] - 0.11 \* x0[4];

x1[2] := 0.48 + 0.13 \* x1[1] - 0.12 \* x0[2] + 0.09 \* x0[3] - 0.06 \* x0[4];

x1[3] := -2.34 - 0.11 \* x1[1] + 0.05 \* x1[2] - 0.02 \* x0[3] + 0.12 \* x0[4];

x1[4] := 0.72 + 0.13 \* x1[1] + 0.18 \* x1[2] - 0.24 \* x1[3] + 0.43 \* x0[4];

q := Max(abs(x1[1] - x0[1]), abs(x1[2] - x0[2]), abs(x1[3] - x0[3]), abs(x1[4] - x0[4]));

writeln(' ',x0[1]:0:4,' | ',x1[1]:0:4,' | ',x0[2]:0:4,' | ',x1[2]:0:4,' | ',x0[3]:0:4,' | ',x1[3]:0:4,' | ',x0[4]:0:4,' | ',x1[4]:0:4,' | ',q:0:5);

inc(k)

until(q < eps);

Writeln('-----------------');

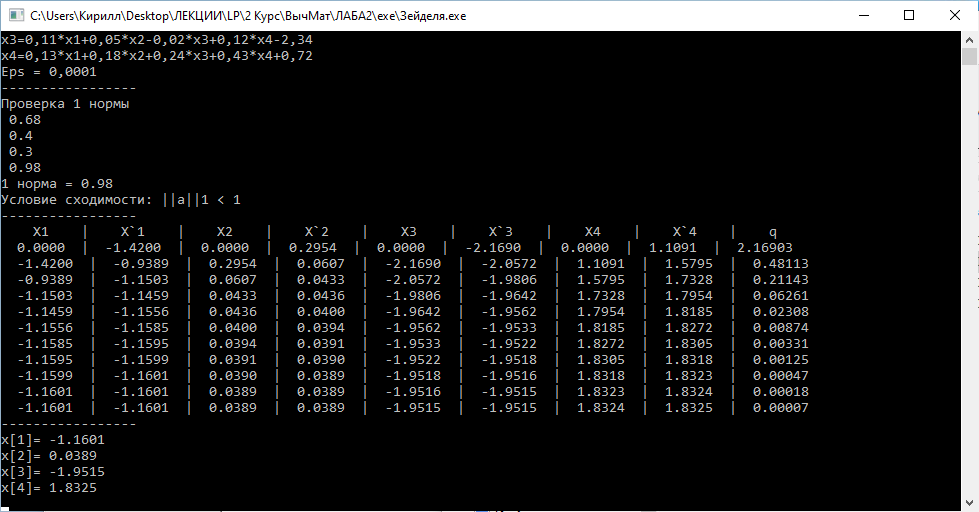
for i := 1 to 4 do

writeln('x[', i, ']= ', x1[i]:0:4);

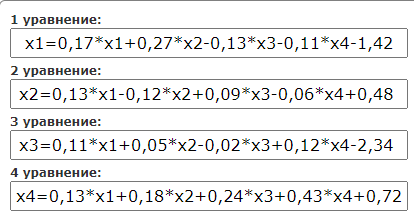
readln;

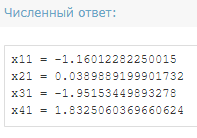
end.

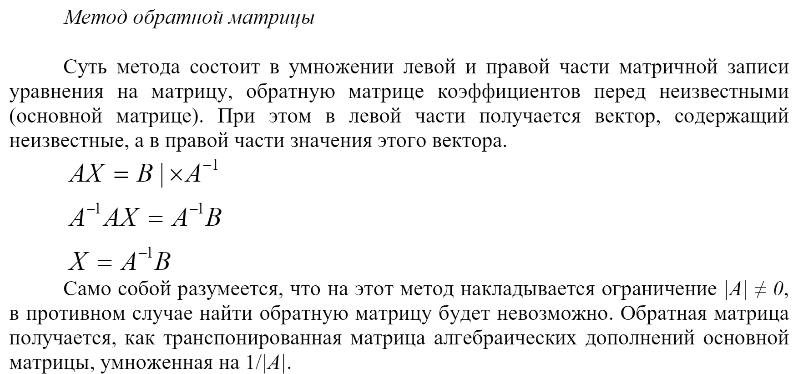
Экранные формы:



Проверка:







Листинг:

Program Obratnaya;

uses crt;

const

nmax = 3;

var

n: integer;

type

Tmass = array[1..nmax] of real;

Tmatrix = array[1..nmax, 1..nmax] of real;

{перестановка строк при главном элементе=0}

procedure Per(k: integer; var a: Tmatrix; var p: integer);

var

z: Real;

j, i: integer;

begin

z := abs(a[k, k]);{модуль главного элемента}

i := k;{номер строки}

p := 0;{количество перестановок}

for j := k + 1 to n do {ищем в столбце ниже}

begin

if abs(a[j, k]) > z then {элемент по модулю больше}

begin

z := abs(a[j, k]);

i := j;

p := p + 1;//счетчик перестановок

end;

end;

if i > k then{если нашли}

for j := k to n do

begin

z := a[i, j];

a[i, j] := a[k, j];{обмениваем строки}

a[k, j] := z;

end;

end;

{определение знака определителя}

function Znak(p: integer): integer;

begin

if p mod 2 = 0 then

Znak := 1 else Znak := -1;

end;

{вычисление определителя матрицы коэффициентов по Гауссу}

procedure Opr(var det: real; var a: tmatrix);

var

k, i, j, p: integer;r: real;

begin

det := 1.0;

for k := 1 to n do

begin

if a[k, k] = 0 then Per(k, a, p);//перестановка строк

det := znak(p) \* det \* a[k, k];//вычисление определителя

for j := k + 1 to n do //пересчет коэффициентов

begin

r := a[j, k] / a[k, k];

for i := k to n do

a[j, i] := a[j, i] - r \* a[k, i];

end;

end;

end;

{вычисление алгебраических дополнений}

procedure Dop(d: tmatrix; var det1: real);

var

k, i, j, p: integer;r: real;

begin

det1 := 1.0;

for k := 2 to n do

begin

Per(k, d, p);

det1 := znak(p) \* det1 \* d[k, k];

for j := k + 1 to n do

begin

r := d[j, k] / d[k, k];

for i := k to n do

d[j, i] := (d[j, i] - r \* d[k, i]);

end;

end;

end;

{установление знака алгебраических дополнений}

function Znak1(i, m: integer): integer;

begin

if (i + m) mod 2 = 0 then

Znak1 := 1 else Znak1 := -1;

end;

{формирование присоединенной матрицы}

procedure Peresch(b: Tmatrix; var e: Tmatrix);

var

i, m, k, j: integer;z, det1: real;d, c: Tmatrix;

begin

for i := 1 to n do

begin

for m := 1 to n do

begin

for j := 1 to n do {перестановка строки}

begin

z := b[i, j];

for k := i downto 2 do

d[k, j] := b[k - 1, j];

for k := i + 1 to n do

d[k, j] := b[k, j];

d[1, j] := z;

end;

for k := 1 to n do {перестановка столбца}

begin

z := d[k, m];

for j := m downto 2 do

c[k, j] := d[k, j - 1];

for j := m + 1 to n do

c[k, j] := d[k, j];

c[k, 1] := z;

end;

Dop(c, det1); {вычисление дополнений}

e[i, m] := (det1) \* znak1(i, m); {установление знака дополнений и }

end; {формирование присоединенной матрицы }

end;

end;

{транспонирование матрицы}

procedure Trans(b: Tmatrix; var e: Tmatrix);

var

i, j: integer;

begin

writeln('--------------------------');

writeln('Матрица алгебраических дополнений:');

for i := 1 to n do

begin

for j := 1 to n do

begin

write(' ', b[i, j]);

end;

writeln;

end;

for i := 1 to n do

for j := 1 to n do

e[i, j] := b[j, i];

end;

{нахождение корней умножением обратной матрицы на столбец свободных членов}

procedure Resh(n: integer; a: Tmatrix; b: Tmass; var x: Tmass);

var

k, j, i: integer;z: real;

begin

writeln('--------------------------');

writeln('Обратная матрица:');

for i := 1 to n do

begin

for j := 1 to n do

begin

write(' ', a[i, j]:4:3);

end;

writeln;

end;

writeln('--------------------------');

for k := 1 to n do

begin

x[k] := 0;

for j := 1 to n do

begin

z := a[k, j] \* b[j];

x[k] := x[k] + z;

end;

end;

end;

var

a, a1, at, b, c: Tmatrix;

f, x: Tmass;

det: Real;

i, j: integer;

begin

clrscr;

{решение системы}

n := 3;

a[1, 1] := 2; a[1, 2] := 4; a[1, 3] := 3;

a[2, 1] := 1; a[2, 2] := 1; a[2, 3] := 4;

a[3, 1] := 2; a[3, 2] := 3; a[3, 3] := 4;

f[1] := 13; f[2] := 17; f[3] := 12;

clrscr;

WRITELN('МЕТОД ОБРАТНОЙ МАТРИЦЫ');

WRITELN('-----------------');

writeln('Исходная система:');

WRITELN;

for i := 1 to n do

begin

for j := 1 to n do

write(a[i, j]:5:1);

writeln(' ', f[i]:7:1);

end;

writeln;

a1 := a;{сделаем копию матрицы для нахождения определителя, она изменится}

Opr(det, a1);{вычисление определителя матрицы}

writeln('Определитель=', det:0:0);

if det = 0 then

begin

write('Решений не существует');

readln;

exit;

end;

Peresch(a, b); { вычисление присоединенной матрицы}

Trans(b, c);{транспонирование присоединенной матрицы}

for i := 1 to n do

for j := 1 to n do

c[i, j] := c[i, j] / det;{деление на определитель=обратная матрица}

{нахождение корней}

Resh(n, c, f, x);

for i := 1 to n do

writeln('x[', i, ']=', x[i]:0:3);

readln;

end.

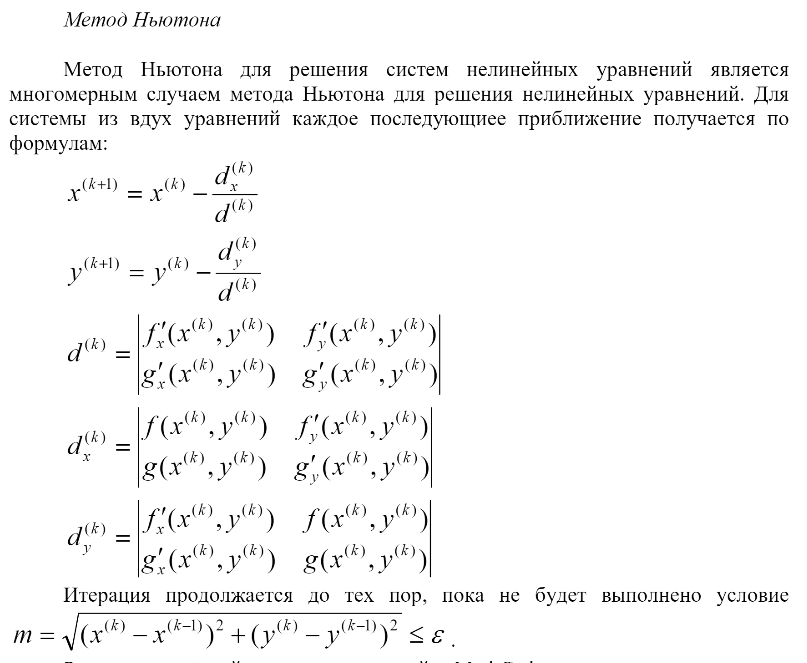
Экранные формы:



Проверка:







Листинг:

program newton;

uses

crt;

const

n = 2;

eps = 0.001;

type

ss = array[1..2] of real;

var

s: ss;

x0, y0:real;

function f1(x, y: real): real;

begin

f1 := x - ln(-0.9 - y);

end;

function f2(x, y: real): real;

begin

f2 := y - (sin(x + 0.8)/cos(x + 0.8));

end;

function f1x(x, y: real): real;

begin

f1x := 1;

end;

function f1y(x, y: real): real;

begin

f1y := 1 / (-0.9 - y);

end;

function f2x(x, y: real): real;

begin

f2x := -1 / (cos(x + 0.8) \* cos(x + 0.8));

end;

function f2y(x, y: real): real;

begin

f2y := 1;

end;

procedure newt(s1: ss);

var

a: array[1..2,1..2] of real;

b: array[1..2] of real;

d,dx,dy: real;

f:Boolean;

q:real;

begin

x0:=s1[1];

y0:=s1[2];

f:=true;

while f do begin

a[1,1]:=f1x(x0,y0); a[1,2]:=f1y(x0,y0);

a[2,1]:=f2x(x0,y0); a[2,2]:=f2y(x0,y0);

{for i:=1 to 2 do

for j:=1 to 2 do

write(' ', a[i,j]:0:3);

writeln;}

b[1]:=-f1(x0,y0);

b[2]:=-f2(x0,y0);

{for i:=1 to 2 do

write(' ', b[i]:0:3);

writeln;}

d:=(a[1,1]\*a[2,2])-(a[2,1]\*a[1,2]);

dx:=(b[1] \* a[2,2] - b[2] \* a[1,2]) / d;

dy:=(a[1,1] \* b[2] - b[1] \* a[2,1]) / d;

q:=max(dx\*dx,dy\*dy);

writeln(' ',x0:0:4,' | ',y0:0:4,' | ',a[1,1]:0:4,' | ',a[1,2]:0:4,' | ',a[2,1]:0:4,' | ',a[2,2]:0:4,' | ',dx:0:4,' | ', dy:0:4,' | ',q:0:4);

if q < eps then f:= false

else

begin

x0:=x0+dx;

y0:=y0+dy;

end;

end;

end;

begin

writeln('МЕТОД НЬЮТОНА');

writeln('Исходная ситсема');

writeln('x-ln(-0,9-y)=0 (f1)');

writeln('y-tg(x+0,8)=0 (f2)');

writeln('Частные производные:');

writeln('f1x = 1');

writeln('f1у = 1 / (-0.9 - y)');

writeln('f2x = -1 / (cos(x + 0.8) \* cos(x + 0.8))');

writeln('f2у = 1');

writeln('-------------------------------------------');

writeln('Eps = 0.001');

writeln('-------------------------------------------');

s[1] := 1;

s[2] := -1;

writeln(' x | y | f1x | f1y | f2x | f2y | dx | dy | q ');

newt(s);

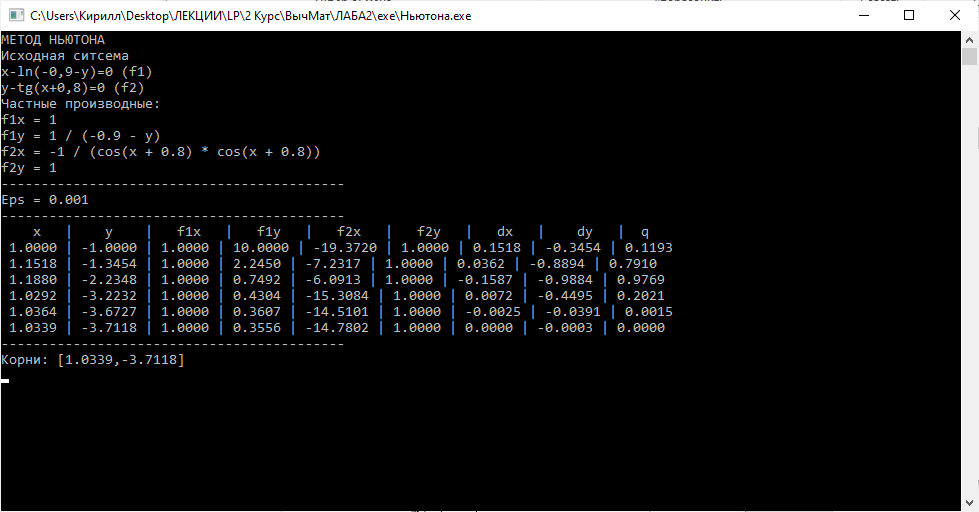
writeln('-------------------------------------------');

writeln('Корни: [',x0:0:4,',',y0:0:4,']');

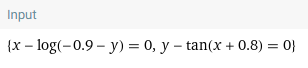
readln;

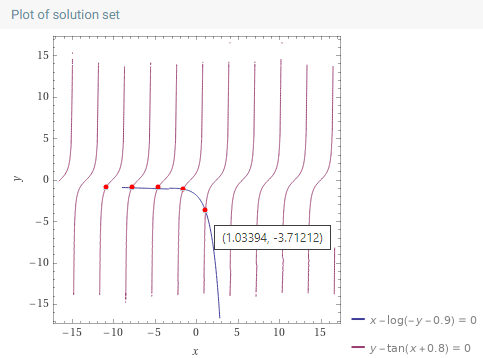
end.

Экранные формы:



Проверка:





**Вывод**

В ходе выполнения лабораторной работы были изучены численные методы решения систем линейных и нелинейных уравнений, изучены их преимущества и недостатки. Для всех методов была написана программа, реализующая их.