# Приложение к отчету по курсовому проекту «Реализация численных методов»

Выполнил: студент группы 23631/1 Цветков А.Д.

Преподаватель: Павлова Л.В.

Оглавление

[Оглавление 1](#_Toc533375436)

[Часть 1: решение алгебраических и трансцендентных уравнений (вариант 20) 2](#_Toc533375437)

[Часть 2: решение СЛАУ прямыми методами (метод LU-разложения) 8](#_Toc533375444)

[Часть 3: решение СЛАУ итерационными методами (метод Зейделя) 14](#_Toc533375445)

[Часть 4: решение алгебраической проблемы собственных значений (метод Якоби) 15](#_Toc533375446)

## Часть 1: Решение алгебраических и трансцендентных уравнений (вариант 20)

Функция полинома: .

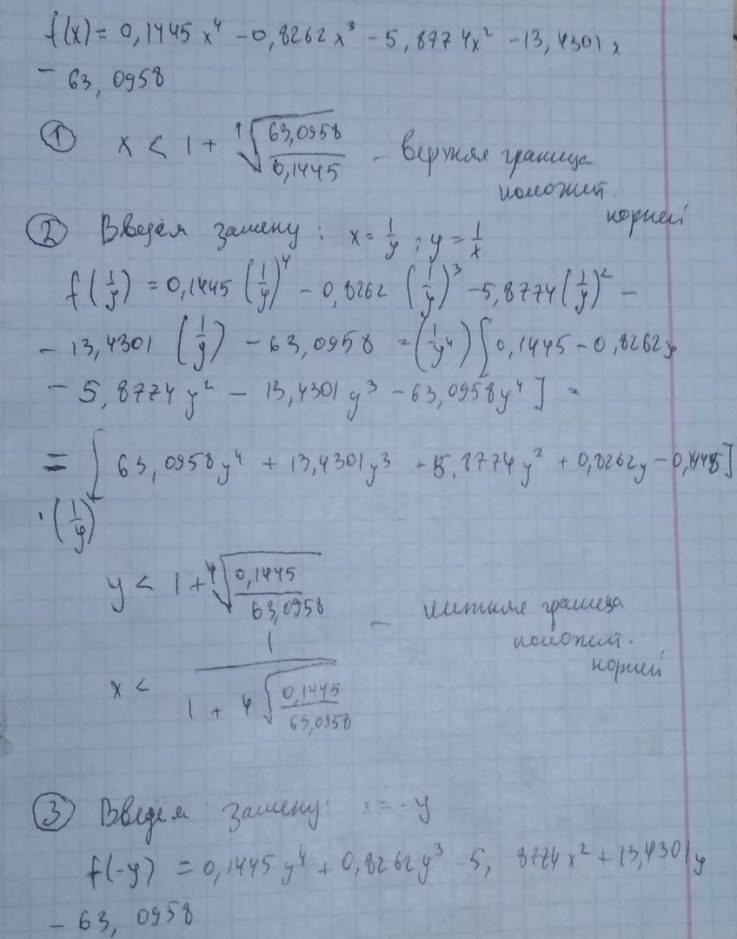
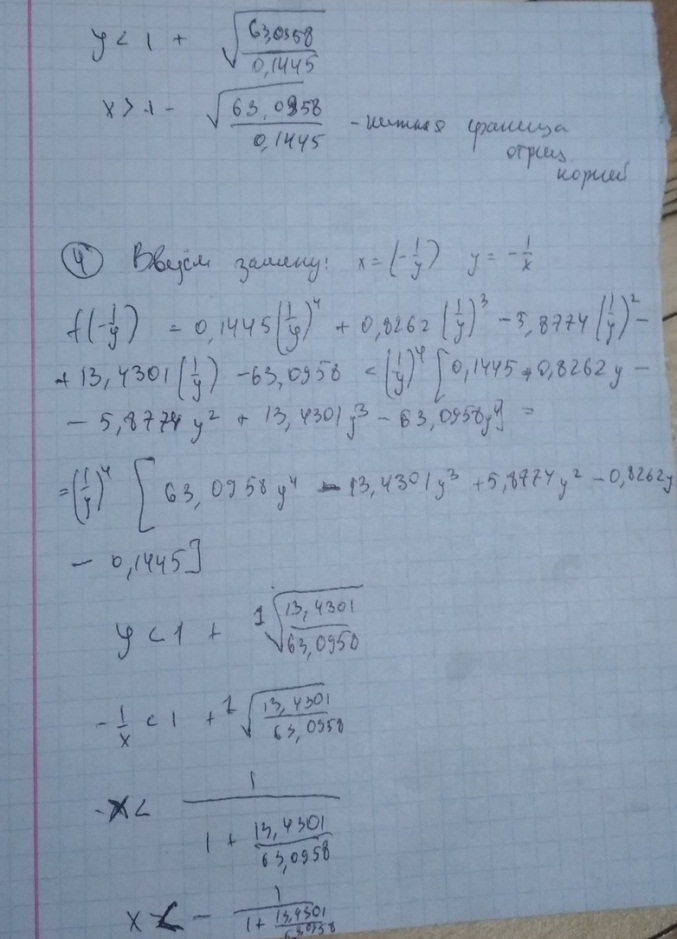
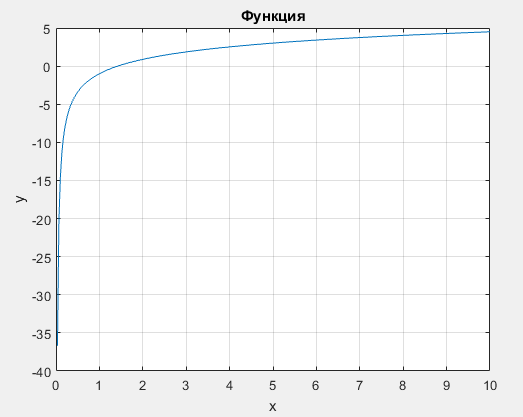
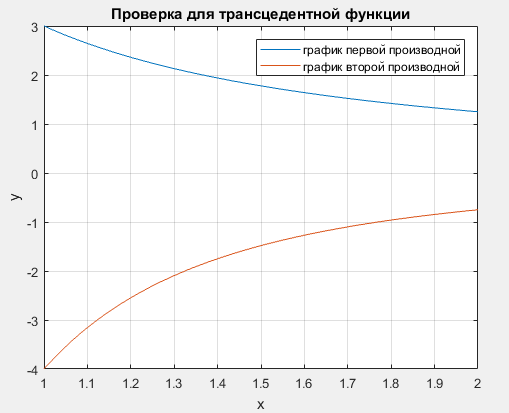
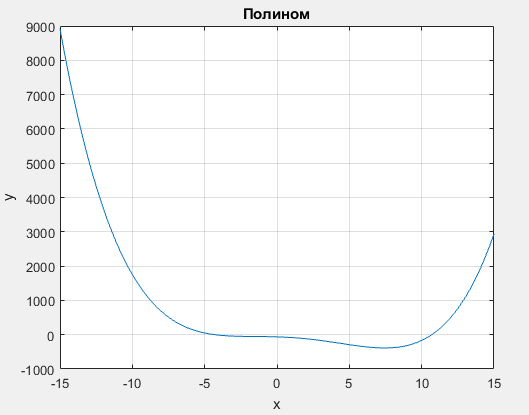
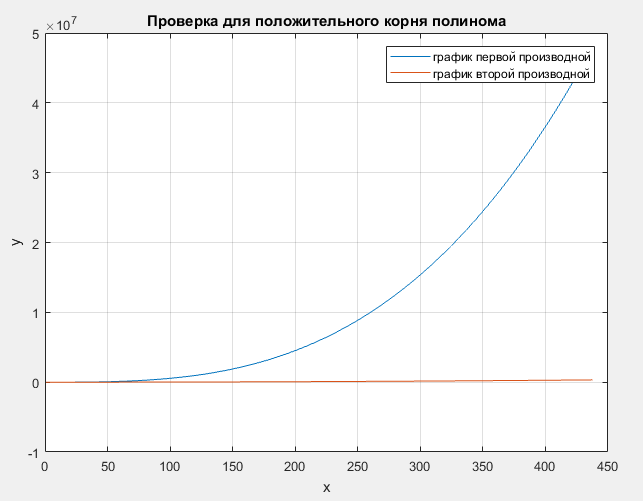
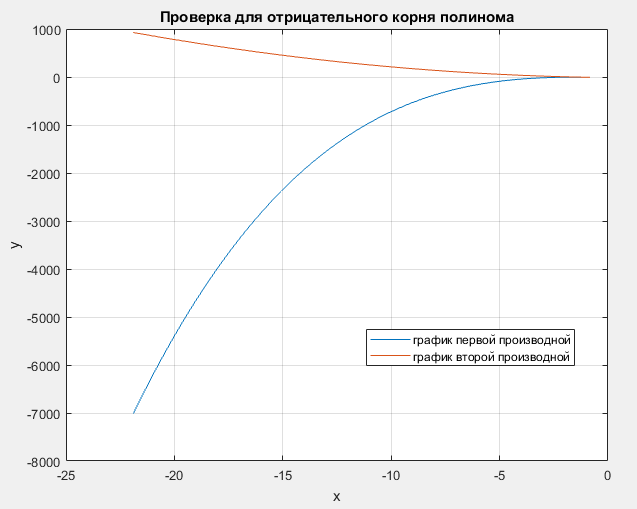
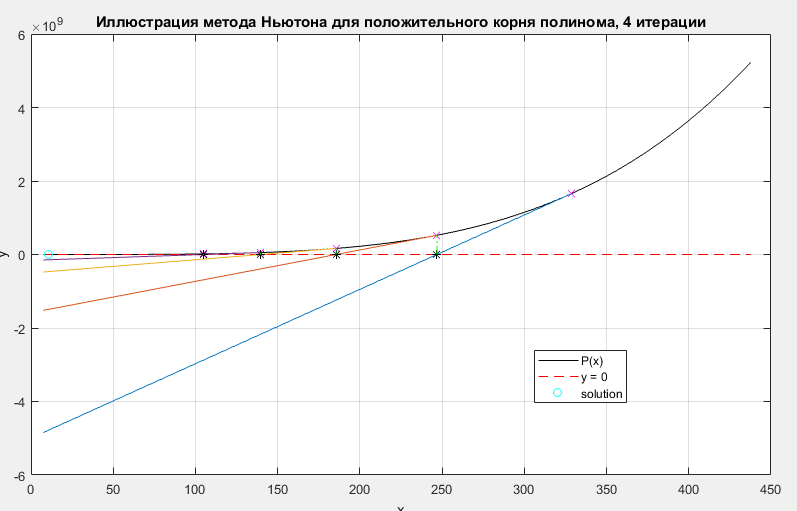
Трансцендентная функция:    
Рисунки 1, 2. Подсчет верхней и нижней границ отрезка для поиска корня полинома по теореме о верхней границе положительных корней полинома.

Рисунок 3. График трансцедентной функции. По графику замечаем, что корень находится в промежутке [1, 2].  
  
Рисунок 4. Графическая проверка знакопостоянства первой и второй производной на выбранном промежутке.  
  
Рисунок 5. График полинома – алгебраической функции.

  
Рисунок 6. Графическая проверка знакопостоянства первой и второй производной на выбранном промежутке для положительного корня полинома. Чтобы выполнялись условия применимости метода Ньютона, сдвигаем левую границу рассматриваемого промежутка до 7.45 (видно из детального рассмотрения графиков).

  
Рисунок 7. Графическая проверка знакопостоянства первой и второй производной на выбранном промежутке для отрицательного корня полинома. Чтобы выполнялись условия применимости метода Ньютона, сдвигаем правую границу рассматриваемого промежутка до -1.55 (видно из детального рассмотрения графиков).

Рисунок 8. Графическая иллюстрация метода Ньютона для положительного корня полинома (поднимание корня), представлено 4 итерации.

for i = 1:1:9

count = 1;

eps = 10^(-i);

c = (a + b) / 2;

while abs(b - a) > 2 \* eps

count = count + 1;

c = (a + b) / 2;

if fun(c) == 0

break

end

if fun(a) \* fun(c) < 0

b = c;

end

if fun(b) \* fun(c) < 0

a = c;

end

end

a=a1;

b=b1;

end

Код программы 1. Реализация метода половинного деления средствами пакета МатЛаб.

if (abs(df1(a))>abs(df1(b)))

m = abs(df1(b));

else m = abs(df1(a));

end

if (abs(df2(a))>abs(df2(b)))

M = abs(df2(a));

else M = abs(df2(b));

end

for i = 1:1:9

count = 0;

eps = 10^(-i);

x0 = a;

if fun(x0)\*df2(x0)<=0 x0=b;

end

c = x0-(fun(x0)/df1(x0));

est = sqrt((2\*eps\*m)/M);

while abs(c-x0)> est

x0 = c;

c = x0-(fun(x0)/df1(x0));

count = count + 1;

end

end

Код программы 2. Реализация метода Ньютона средствами пакета МатЛаб. Используется апостериорная оценка как условие выхода из цикла.

x0 = a;

if fun(x0)\*df2(x0)<=0 x0=b;

end

c = x0-(fun(x0)/df1(x0));

newy = fun(c);

plot (c, newy, 'mx');

for i = 1:1:4

x = linspace(a, c, N);

y = fun(c)+df1(c)\*(x-c);

plot(x, y);

c = (c\*df1(c) - fun(c))/df1(c);

newy = fun(c);

plot (c, newy, 'mx');

plot (c, 0, 'k\*');

plot ([c, c], [0, newy], 'g-.');

hold on;

end

Код программы 3. Реализация графической иллюстрации метода Ньютона средствами пакета МатЛаб для первых четырех итераций.



Таблица 1. Полная иллюстрация работы программы с детальным отчетом.



Таблица 2. Полная иллюстрация работы программы с детальным отчетом.

## Часть 2: решение СЛАУ прямыми методами (метод LU-разложения)

n = 10;

D = eye(n);

E = eye(n);

D(n, n) = 2; %задали диагональную

W = rand(n, 1); %зарандомили вектор W

nr = norm(W, 2);

H = E-2\*W\*W'\*(nr^2);

A = H\*D\*H'

cond = norm(A, 1)\*norm(A^(-1))

b = randn(n, 1);

Код программы 1. Задание хорошо обусловленной матрицы и вектора b.

n=10;

for i = 1:1:n

for j = 1:1:n

C(i, j) = 1/(i+j-1);

end

end

cond1 = norm(C, 1)\*norm(C^(-1))

determinant = det(C)

b1 = randn(n, 1);

Код программы 2. Задание плохо обусловленной матрицы и вектора b.

for(k = 1; k < size; k++)

{

for(i = k-1; i < size; i++)

for(j = i; j < size; j++)

(\*L)[j][i]=(\*U)[j][i]/(\*U)[i][i];

for(i = k; i < size; i++)

for(j = k-1; j < size; j++)

(\*U)[i][j]=(\*U)[i][j]-(\*L)[i][k-1]\*(\*U)[k-1][j];

}

Код программы 3. Код LU-разложения матрицы.

y[0] = (\*b)[0];

for (i=1; i<size; i++)

{

sum =0;

for (j=0; j<=i-1; j++)

sum+=(\*L)[i][j]\*y[j];

y[i] = (\*b)[i] - sum;

}

x[size-1] = y[size-1]/(\*U)[size-1][size-1];

for (i=size-2; i>=0; i--)

{

sum = 0;

for (j=size-1; j>=i+1; j--)

sum+=(\*U)[i][j]\*x[j];

x[i] = (y[i] - sum)/(\*U)[i][i];

}

Код программы 4. Код обратной и прямой подстановки для нахождения векторов х и у.

for(i = 0; i < size; i++)

{

C[i] = 0;

for(k = 0; k < size; k++)

{

C[i] += (\*A)[i][k] \* (\*x)[k];

nev[i] = C[i] - (\*b)[i];

}

}

Код программы 5. Нахождение вектора невязки.

double left=1.0, right=1.01;

for (i=0; i<size; i++)

{

newb[i] = (left + (((double) rand())/(RAND\_MAX+1)) \* (right - left))\*(\*b)[i];

for (j=0; j<size; j++)

{

newA[i][j] = (left + (((double) rand())/(RAND\_MAX+1)) \* (right-left))\*(\*A)[i][j];

}

}

Код программы 6. Внесение изменений в вектор b и в матрицу A не более одного процента.

%cчитаем коэффицент для возмущений в b

delx = xb-x;

delb = newb-b;

k1 = (norm(delx, 2)\*norm(b, 2))/(norm(x, 2)\*norm(delb, 2))

%cчитаем коэффициент для возмущений в А

delA = newA-A;

delx = xA-x;

k2 = (norm(delx, 2)\*norm(A, 2))/(norm(xA, 2)\*norm(delA, 2))

Код программы 7. Вычисление коэффициентов для возмущений в b и в A.

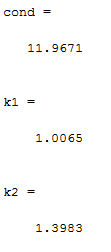


Рисунок 1. Результат вычисления коэффициентов для матрицы 10х10 с маленьким числом обусловленностей.

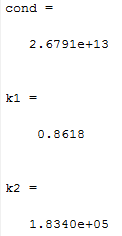


Рисунок 2. Результат вычисления коэффициентов для матрицы 10х10 с большим числом обусловленностей.

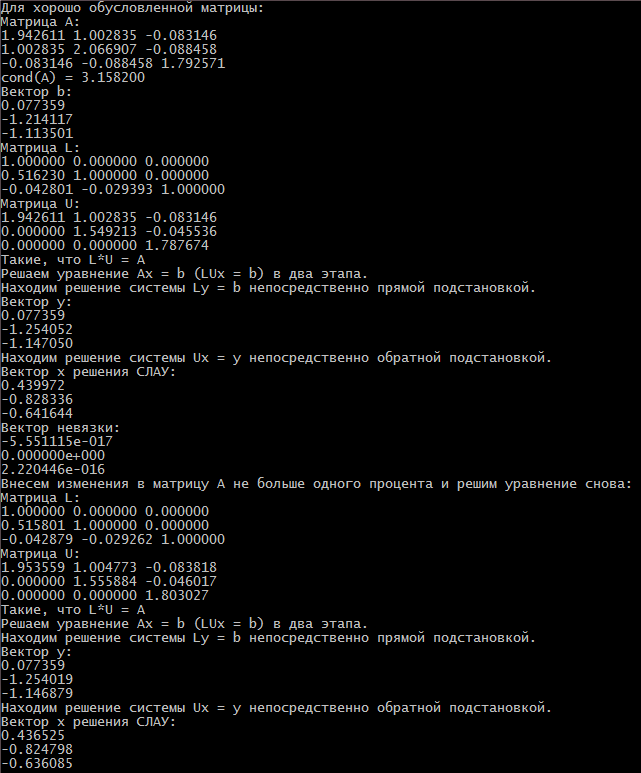


Рисунок 3. Результат работы программы для задачи малой размерности.

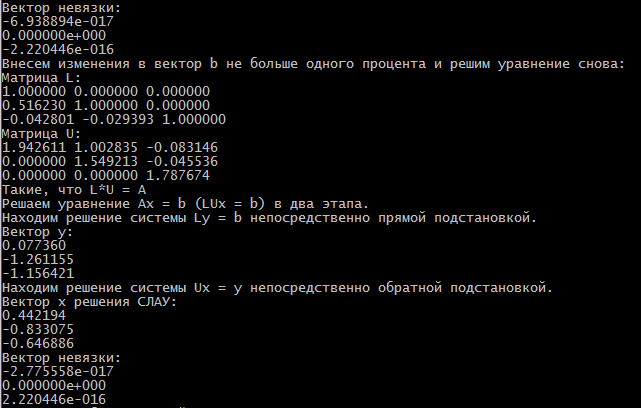


Рисунок 4. Результат работы программы для задачи малой размерности.

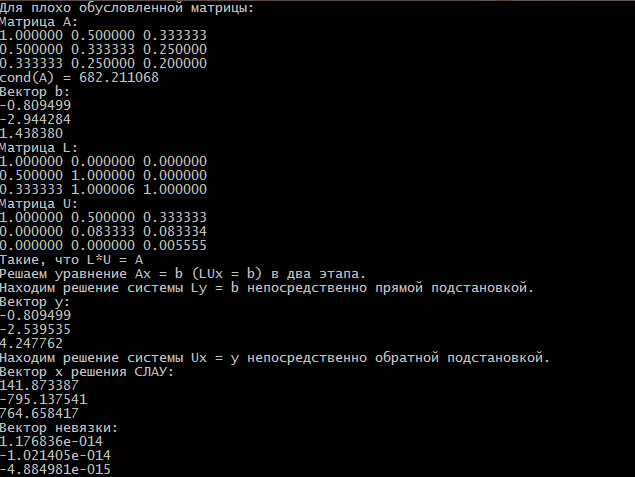


Рисунок 5. Результат работы программы для задачи малой размерности.

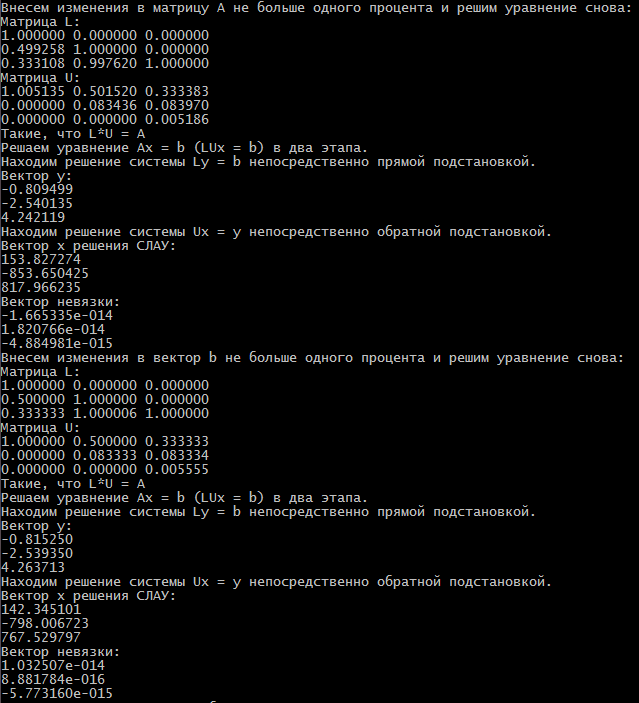


Рисунок 6. Результат работы программы для задачи малой размерности.

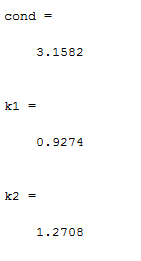


Рисунок 7. Результат вычисления коэффицентов для матрицы с маленьким числом обусловленностей для задачи малой размерности.

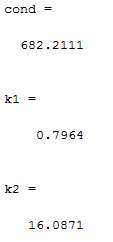


Рисунок 8. Результат вычисления коэффицентов для матрицы с большим числом обусловленностей для задачи малой размерности.

## Часть 3: решение СЛАУ итерационными методами (метод Зейделя)

for (i=0; i<size; i++)

{

for (j=0; j<size; j++)

{

if (i!=j) (\*alpha)[i][j] = -((\*matrnew)[i][j]/(\*matrnew)[i][i]);

else (\*alpha)[i][j] = 0;

}

}

for (i=0; i<size; i++)

{

(\*beta)[i] = (\*bnew)[i]/(\*matrnew)[i][i];

}

Код программы 1. Приведение системы к виду x = αx + β .

while(normir(&xsub) > eps)

{

for (i=0; i<size; i++) x[i] = 0;

for (i=0; i<size; i++)

{

for (j=0; j<i; j++)

{

x[i] += (\*alpha)[i][j]\*x[j];

}

for (j=i; j<size; j++)

{

x[i] += (\*alpha)[i][j]\*xfirst[j];

}

x[i] +=(\*beta)[i];

}

count++;

for (i=0; i<size; i++)

{

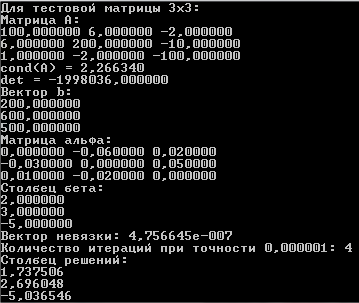
xsub[i] = x[i]-xfirst[i];

xfirst[i] = x[i];

}

}

Код программы 2. Решение СЛАУ методом Зейделя.

  
Рисунок 1. Иллюстрация работы программы для задачи малой размерности.

## Часть 4: решение алгебраической проблемы собственных значений (метод Якоби)

n = 3; %размерность

R=[1 2 3];

A=sprandsym(10, 1, R); %задание матрицы по заданным собственным числам

A1 = zeros(10, 10);

arr = A+A1; %поскольку матрица разреженная, делаем из нее обычную, добавляя нулевую

Код программы 1. Задание матрицы по собственным числам

while (find(matr) > eps)

{

for (i=0; i<size; i++)

{

sum = 0;

for (j=0; j<size; j++)

{

if (i!=j) sum += ((\*matr)[i][j])\*((\*matr)[i][j]);

}

if (max<sum)

{

max = sum;

ii = i;

}

}

max = 0;

for (j=0; j<size; j++)

{

if (j!=ii)

{

if (fabs((\*matr)[ii][j])>fabs(max))

{

max = (\*matr)[ii][j];

jj = j;

}

}

}

turn(g, gt, matr);

multiply (vector, g, vector);

multiply (gt, matr, matr);

multiply (matr, g, matr);

count++;  
}

Код программы 2. Вычисление собственных векторов и собственных чисел методом Якоби

for (i=0; i<size; i++)

{

for (j=0; j<size; j++)

{

if (i!=j) (\*g)[i][j] = 0;

else (\*g)[i][j] = 1;

}

}

if (((\*matr)[jj][jj] - (\*matr)[ii][ii])!=0)

{

tan = 2\*(\*matr)[ii][jj]/((\*matr)[jj][jj] - (\*matr)[ii][ii]);

printf("fi = %lf\n", 0.5\*atan(tan));

s = sqrt(0.5\*(1-1/(sqrt(1+tan\*tan))));

c = sqrt(0.5\*(1+1/(sqrt(1+tan\*tan))));

if (tan<0) s=-s;

}

else

{

s = sqrt(0.5);

c = sqrt(0.5);

}

(\*g)[ii][ii] = c;

(\*g)[jj][ii] = -s;

(\*g)[ii][jj] = s;

(\*g)[jj][jj] = c;

Код программы 3. Вычисление матрицы поворота.

double max=0;

s = 0;

while (s<size)

{

for (i=0; i<size; i++)

{

for (j=0; j<1; j++)

{

res[i] = 0;

for (k=0; k<size; k++)

{

res[i]+= A[i][k]\*(\*vector)[k][s];

}

}

}

for (i=0; i<size; i++) res1[i] = (\*vector)[i][s]\*(\*matr)[s][s];

for (i=0; i<size; i++) if (res[i]-res1[i]>max) max = res[i]-res1[i];

s++;

}

Код программы 4. Вычисление вектора невязки

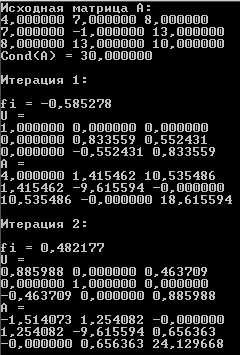


Рисунок 1. Иллюстрация работы программы для задачи малой размерности

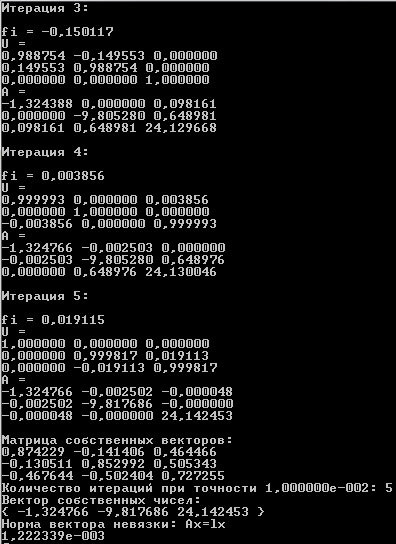


Рисунок 2. Иллюстрация работы программы для задачи малой размерности