Задание №1. Минимизация квадратичной функции

Вариант №9 (N = 9)

***Цель задания***: *практическое освоение итерационных методов поиска минимума функции с заданной точностью*

1. С помощью программной реализации ниже указанных методов:

* **МНГС (метод наискорейшего градиентного спуска)**
* **МНПС (метод наискорейшего покоординатного спуска)**

найти точку минимума функции:

с точностью = 10-6 . Здесь *N —* номер студента по списку группы.

1. В отчете для каждого из методов указать значения:
2. *N —* номер в списке группы,
3. *А, b —* матрица и вектор из матричного вида квадратичной функции,
4. — выбранное начальное приближение к решению,
5. *—* полученная точка минимума,
6. —значение функции в полученной точке минимума,
7. — абсолютная погрешность полученного решения.

Решение

Согласно варианту задания:

Аналитическое решение:

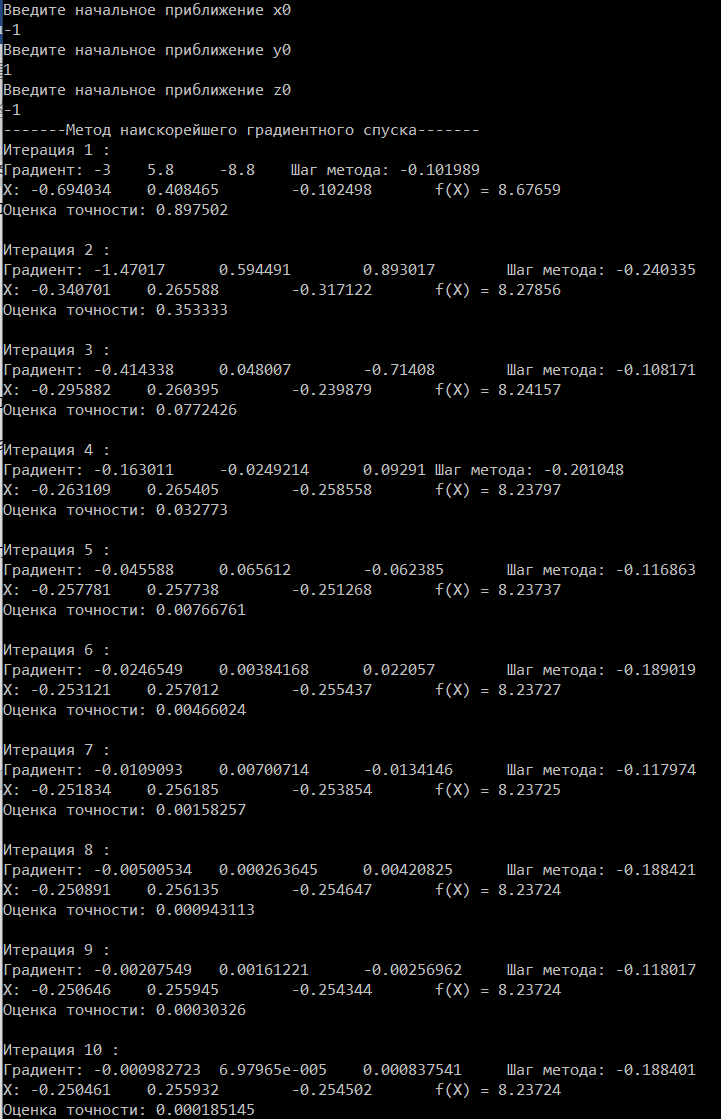
Запишем в векторном виде:

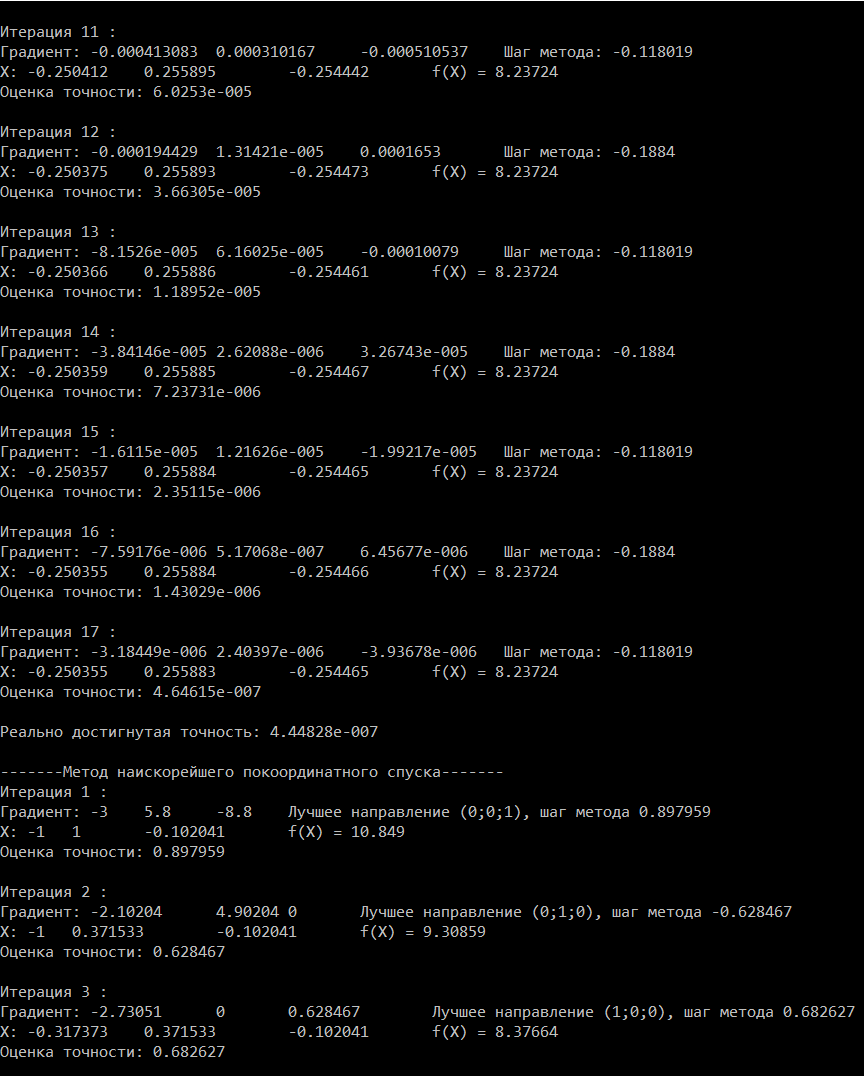
Здесь

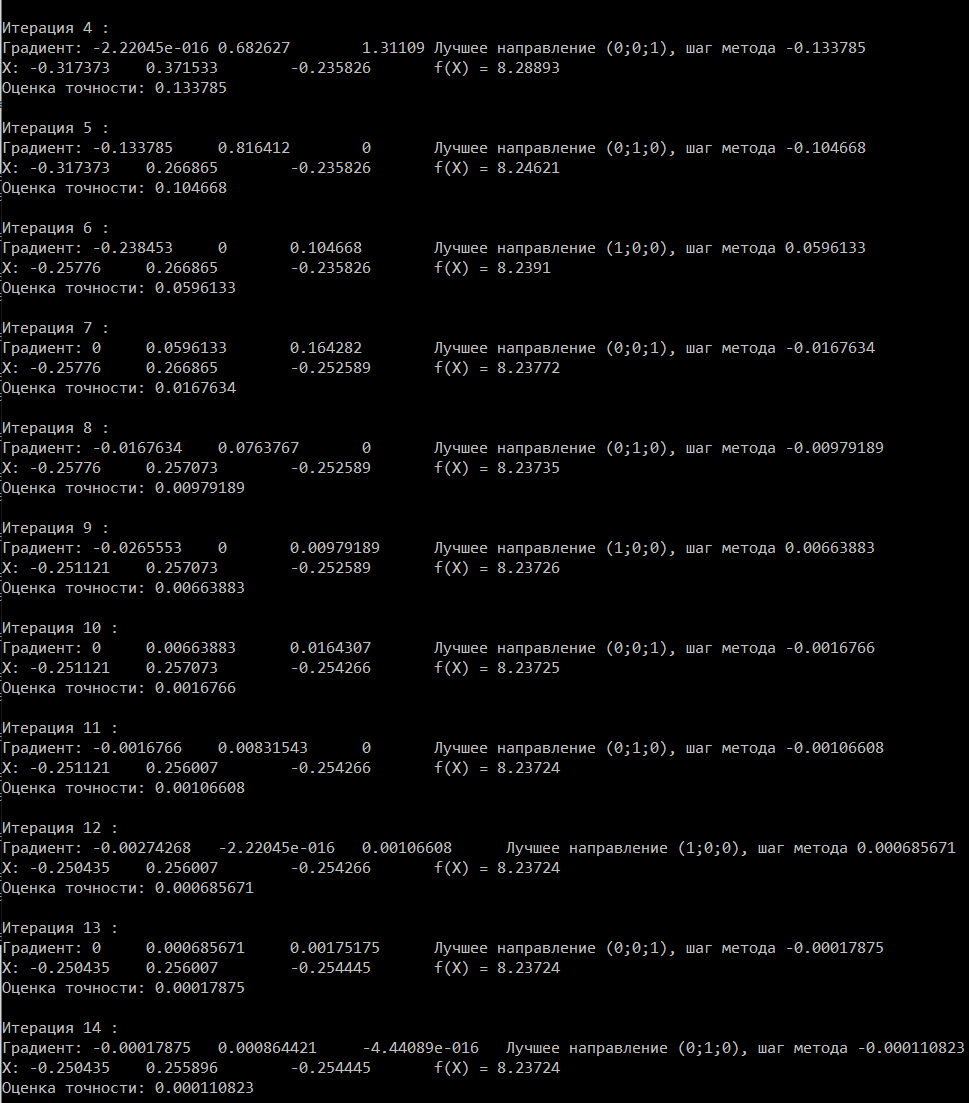
Для метода наискорейшего спуска имеем

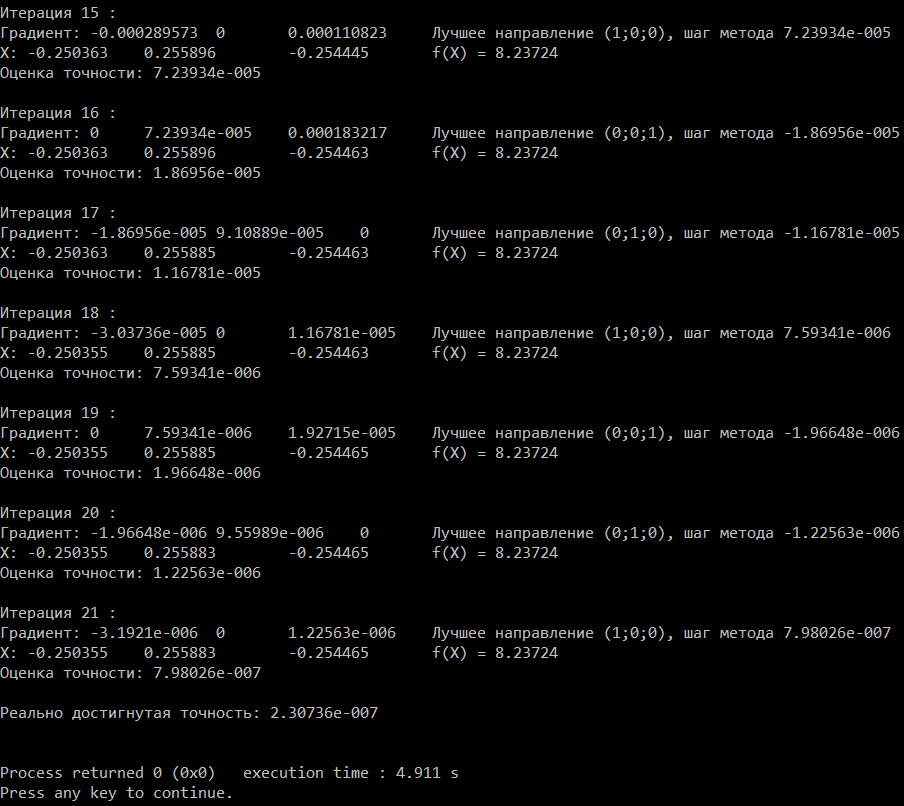
Для метода наискорейшего покоординатного спуска имеем

Результаты выполнения программы:









В качестве начального приближения выбрали вектор .

Метод наискорейшего градиентного спуска нашёл решение с заданной точностью за 17 итераций.

Метод наискорейшего покоординатного спуска нашёл решение с заданной точностью за 21 итерацию.

Таким образом, метод наискорейшего градиентного спуска сходится быстрее, но на каждой итерации объём вычислений больше.

Приложение

Листинг программы

#include <iostream>

#include <locale>

#include <cmath>

using namespace std;

int const N=3; //размерность матрицы

double A[N][N] = {{4,1,1},{1,7.8,-1},{1,-1,9.8}};

double b[N] = {1,-2,3}, e1[N]={1,0,0},e2[N]={0,0,1},e3[N]={0,0,1};

double mu, x0[N],x1[N],x2[N],x3[N], q[N], Aq[N], Xeps[N], Xdelta[3];

double tochno[3];

double f(double x[N])

{

double S=0;

for (int i=0; i<N; i++)

{

for (int j=0; j<N; j++)

S+=A[i][j]\*x[i]\*x[j];

}

S=0.5\*S+9;

for (int i=0; i<N; i++) S+=x[i]\*b[i];

return S;

}

//бесконечная норма вектора

double NormInfVect(double b[N])

{

double maxb=0;

for (int j=0; j<N; j++)

{

if (maxb<abs(b[j])) maxb=abs(b[j]);

}

return maxb;

}

//умножение матрицы A на вектор q

void MultAq(double A1[N][N], double q1[N])

{

double S;

for (int i=0; i<N; i++)

{

S = 0;

for (int j=0; j<N; j++)

{

S+=A1[i][j]\*q1[j];

}

Aq[i]=S;

}

}

//формирование вектора q

void Getq(double x[N])

{

MultAq(A,x);

for (int i=0; i<N; i++)

{

q[i]=Aq[i]+b[i];

}

}

//скалярное умножение векторов

double Multq(double q1[N], double q2[N])

{

double S=0;

for (int i=0; i<N; i++)

{

S+=q1[i]\*q2[i];

}

return S;

}

//одна итерация метода наискорейшего градиентного спуска

double OneIterationBest(int n)

{

int i;

double mu, Xnew[N];

Getq(x1);//формируем направление градиента

cout<<"\nГрадиент: "<<q[0]<<'\t'<<q[1]<<'\t'<<q[2]<<'\t';

MultAq(A,q);//формируем произведение A\*градиент

mu = - Multq(q,q)/Multq(q,Aq);

cout<<"Шаг метода: "<<mu<<endl;

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

{

Xnew[i]=x1[i]+mu\*q[i];

//разница между итерациями

Xeps[i]=x1[i]-Xnew[i];

}

//вывод текущего приближения

cout<<"X: ";

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

{

x1[i]=Xnew[i];

cout<<x1[i]<<'\t';

}

cout<<"f(X) = "<<f(x1)<<endl;

return NormInfVect(Xeps);

}

//метод наискорейшего градиентного спуска

void Bestspusk(double Eps0)

{

int i,n=N;

cout << "-------Метод наискорейшего градиентного спуска-------"<<endl;

for(i=0;i<n;i++) x1[i]=x0[i];

int k=1;

double eps;

do

{

cout << "Итерация " << k << " : ";

eps = OneIterationBest(n);

cout << "Оценка точности: " << eps <<endl<<endl;

k++;

}

while(eps>Eps0);

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

{

Xdelta[i]=fabs(x1[i]-tochno[i]);

}

cout << "Реально достигнутая точность: " << NormInfVect(Xdelta) <<endl << endl;

}

//одна итерация метода наискорейшего покоординатного спуска

double OneIterationKoord(int n)

{

int i;

double mu1, mu2, mu3, Xnew[N], Xnew1[N], Xnew2[N], Xnew3[N],f1,f2,f3;

Getq(x1);//формируем направление градиента

cout<<"\nГрадиент: "<<q[0]<<'\t'<<q[1]<<'\t'<<q[2]<<'\t';

mu1 = -q[0]/A[0][0];//коэффициент в направлении 1-го орта

mu2 = -q[1]/A[1][1];//коэффициент в направлении 2-го орта

mu3 = -q[2]/A[2][2];//коэффициент в направлении 3-го орта

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

{

Xnew1[i]=x1[i];Xnew2[i]=x1[i];Xnew3[i]=x1[i];

}

Xnew1[0]=Xnew1[0]+mu1;f1=f(Xnew1);

Xnew2[1]=Xnew2[1]+mu2;f2=f(Xnew2);

Xnew3[2]=Xnew3[2]+mu3;f3=f(Xnew3);

//выбираем лучшее направлении

if ((f1<f2)&(f1<f3))

{

cout<<"Лучшее направление (1;0;0), шаг метода "<<mu1<<endl;

for(i=0;i<n;i++) Xnew[i]=Xnew1[i];

}

if ((f2<f1)&(f2<f3))

{

cout<<"Лучшее направление (0;1;0), шаг метода "<<mu2<<endl;

for(i=0;i<n;i++) Xnew[i]=Xnew2[i];

}

if ((f3<f1)&(f3<f2))

{

cout<<"Лучшее направление (0;0;1), шаг метода "<<mu3<<endl;

for(i=0;i<n;i++) Xnew[i]=Xnew3[i];

}

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

{

//разница между итерациями

Xeps[i]=x1[i]-Xnew[i];

}

//вывод текущего приближения

cout<<"X: ";

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

{

x1[i]=Xnew[i];

cout<<x1[i]<<'\t';

}

cout<<"f(X) = "<<f(x1)<<endl;

return NormInfVect(Xeps);

}

//метод наискорейшего покоординатного спуска

void Bestkoord(double Eps0)

{

int i,n=N;

cout << "-------Метод наискорейшего покоординатного спуска-------"<<endl;

for(i=0;i<n;i++) x1[i]=x0[i];

int k=1;

double eps;

do

{

cout << "Итерация " << k << " : ";

eps = OneIterationKoord(n);

cout << "Оценка точности: " << eps <<endl<<endl;

k++;

}

while(eps>Eps0);

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

{

Xdelta[i]=fabs(x1[i]-tochno[i]);

}

cout << "Реально достигнутая точность: " << NormInfVect(Xdelta) <<endl << endl;

}

int main()

{

double eps=0.000001;

tochno[0]=-883.0/3527.0;tochno[1]=1805.0/7054.0;tochno[2]=-1795.0/7054.0;

setlocale(LC\_ALL, "russian");

cout<<"Введите начальное приближение x0"<<endl;

cin>>x0[0];

cout<<"Введите начальное приближение y0"<<endl;

cin>>x0[1];

cout<<"Введите начальное приближение z0"<<endl;

cin>>x0[2];

Bestspusk(eps);

Bestkoord(eps);

return 0;

}