Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Лабораторная работа №2

Методы спуска (0-го, 1-го и 2-го порядка и переменной метрики)

по дисциплине

«Методы оптимизации»

Факультет прикладной математики и информатики

Группа ПМ-63

Студенты Майер В.А.

Сыпко Н.Е.

Преподаватели Чимитова Е. В.

Вариант Метод Флетчера-Риверса

Метод Девидона-Флетчера-Пауэлла

Новосибирск 2019

1. Цель рабо

Ознакомится с методами поиска минимума функции nпеременных в оптимизационных задачах без ограничения

1. Задание

*Функция Квадратичная - Min (1, 1)*

*Функция Розенброка - Min (1, 1)*

*Функция из варианта -max(2.79914, 1.95318)*

1. Результаты
2. *Метод Флетчера-Ривса*
3. *Квадратичная Функция*

|  |  |  |  |
| --- | --- | --- | --- |
| EPS \ X0 | (3, 3) | (0, 1000) | (-1000,1) |
| 1e-3 | Iter: 4  Function iter: 92  Res: (1,0002488149,  1,0002499876)  Func: 6,2E-08 | Iter: 24  Function iter: 552  Res: (0,99967290062 , 0,9996709468)  Func: 1,07E-07 | Iter: 16  Function iter: 368  Res: (0,999987644716403 , 0,999986907601326)  Func: 2,07E-10 |
| 1e-5 | Iter: 5  Function iter: 165  Res: (1,0000000022,  1,0000000023)  Func: 6,6E-18 | Iter: 9  Function iter: 297  Res: (1,00000002417074 , 1,0000000235821)  Func: 6,19-16 | Iter: 9  Function iter: 297  Res: (0,999999934754357 , 0,999999928908884)  Func: 7,67E-15 |
| 1e-7 | Iter: 3  Function iter: 128  Res: (0,99999994230,  0,99999994201)  Func: 3,33-15 | Iter: 6  Function iter: 257  Res: (0,99999999414,  0,999999994112667)  Func: 3,44E-17 | Iter: 6  Function iter: 257  Res: (1,0000000368433 , 1,00000003702564)  Func: 1,36E-15 |

1. *ФункцияРозенброка*

|  |  |  |  |
| --- | --- | --- | --- |
| EPS \ X0 | (3, 3) | (0, 1000) | (-5,7) |
| 1e-3 | Iter: 21  Function iter: 1594  Res: (1,00009103361038 , 1,000182496545)  Func: 8,304845442E-09 | Iter: 100  Function iter: 5703  Res: (1,000322018697 , 1,00064566978009)  Func: 1,03929730544425E-07 | Iter: 67  Function iter: 3817  Res: (0,999435045280219 , 0,998867356401358)  Func: 3,20106119592119E-07 |
| 1e-5 | Iter: 46  Function iter: 2622  Res: (0,9999987884909 , 0,999997586087881)  Func: 1,47604366977E-12 | Iter: 32  Function iter: 1826  Res: (1,00000147603295 , 1,00000295643308)  Func: 2,1805785985E-12 | Iter: 179  Function iter: 10203  Res: (0,999996284380999 , 0,999992546462304)  Func: 1,385561378696E-11 |
| 1e-7 | Iter: 64  Function iter: 3708  Res: (0,9999999491029 , 0,999999897893928)  Func: 2,6002434639E-15 | Iter: 96  Function iter: 5564  Res: (1,00000006675342 , 1,0000001337969)  Func: 4,46443226536E-15 | Iter: 200  Function iter: 11600  Res: (1,00000001868375 , 1,00000003747864)  Func: 3,503176312717E-16 |

1. *Функция из варианта*

|  |  |  |  |
| --- | --- | --- | --- |
| EPS \ X0 | (3, 3) | (7, -10) | (0,1000) |
| 1e-3 | Iter: 2  Function iter: 65  Res: (2,84062020624924 , 1,97858363438753)  Func: 2,7371565 | Iter: 2  Function iter: 90  Res: (2,21765534761586 , 1,97719891682408)  Func: 2,5705154 | Iter: 0  Function iter: 0  Res: (0 , 1000)  Func: 1,1025235E-05 |
| 1e-5 | Iter: 2  Function iter: 94  Res: (2,84059776142494 , 1,97863273586079)  Func: 2,7371534 | Iter: 7634  Function iter: 251974  Res: (2,73870410421837 , 1,97801541204824)  Func: 2,7361964 | Iter: 0  Function iter: 0  Res: (0 , 1000)  Func: 1,1025235E-05 |
| 1e-7 | Iter: 2  Function iter: 124  Res: (2,84059752896738 , 1,97863290247253)  Func: 2,7371534 | Iter: 10948  Function iter: 470826  Res: (2,74753566647765 , 1,98112775414055)  Func: 2,7366417 | Iter: 0  Function iter: 0  Res: (0 , 1000)  Func: 1,1025235E-05 |

1. *Метод Девидона-Флетчера-Поуэлла*
2. *Квадратичная Функция*

|  |  |  |  |
| --- | --- | --- | --- |
| EPS \ X0 | (3, 3) | (0, 1000) | (-1000,1) |
| 1e-3 | Iter: 15  Func iter: 395  Res: (0,999953133954393 , 0,999955895589584)  Func: 2,96E-09 | Iter: 19  Func iter: 504  Res: (1,00002112197689 , 1,00002214057357)  Func: 5,5E-10 | Iter: 19  Func iter: 504  Res: (0,999977964880595 , 0,999978760052151)  Func: 5,49E-10 |
| 1e-5 | Iter: 7  Func iter: 245  Res: (1,00000002097771 , 1,00000002115141)  Func: 4,43E-16 | Iter: 9  Func iter: 319  Res: (1,00000135581879 , 1,0000013794713)  Func: 1,89E-12 | Iter: 9  Func iter: 319  Res: (0,999998630339318 , 0,999998640431)  Func: 1,89E-12 |
| 1e-7 | Iter: 7  Func iter: 264  Res: (0,999999974067183 , 1,00000002571941)  Func: 2,675E-13 | Iter: 7  Func iter: 263  Res: (1,00000000058907 , 0,999999999414317)  Func: 1,38E-16 | Iter: 7  Func iter: 263  Res: (1,00000000038521 , 0,999999999618575)  Func: 5,89E-17 |

1. *ФункцияРозенброка*

|  |  |  |  |
| --- | --- | --- | --- |
| EPS \ X0 | (3, 1) | (0, 1000) | (-5,7) |
| 1e-3 | Iter: 6905  Func iter: 381055  Res: (1,02169457 , 1,02956092)  Func: 0,0205837925482 | Iter: 6863  Func iter: 381055  Res: (1,00000189221039 , 1,00000155734521)  Func: 4,99568611205E-10 | Iter: 62659  Function iter: 3981055  Res: (0,998945045280219 , 0,999756356401358)  Func: 3,201061195921E-07 |
| 1e-5 | Iter: 7382  Func iter: 398105  Res: (1,02170559407696 , 1,02970041004822)  Func: 0,0205837925482 | Iter: 6728  Func iter: 373537  Res: (1,00000007787168 , 1,00000015370236)  Func: 6,4805725867E-15 | Iter: 7541  Function iter: 1386  Res: (1,0000009387556 , 1,00000187519843)  Func: 8,817973827E-13 |
| 1e-7 | Iter: 10918  Func iter: 348342  Res: (1,02127420528357 , 1,02884222540548)  Func: 0,02049968834500 | Iter: 2778  Func iter: 153388  Res: (1,00000118827859 , 1,00000134592742)  Func: 1,07632065062E-10 | Iter: 5555  Func iter: 226703  Res: (-1,98071064952695 , 3,70696915591925)  Func: 13,5608485214167 |

1. *Функция из задания*

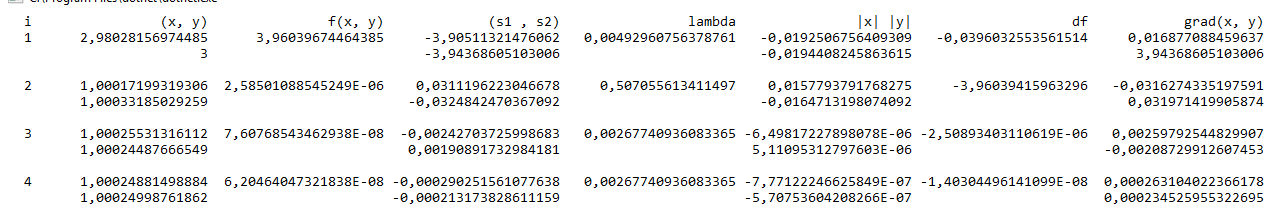
|  |  |  |  |
| --- | --- | --- | --- |
| EPS \ X0 | (3, 3) | (7, -10) | (0,1000) |
| 1e-3 | Iter: 9  Func iter: 137  Res: (2,79241440531783 , 1,95351801251382)  Func: 2,73983335755011 | Iter: 21  Func iter: 220  Res: (2,53009765191702 , -8,7209345028367)  Func: 0,10256701482228 | Iter: 0  Func iter: 0  Res: (0 , 1000)  Func: 1,10252350E-05 |
| 1e-5 | Iter: 13  Func iter: 248  Res: (2,79243982613854 , 1,95351263384466)  Func: 2,73983362839952 | Iter: 277  Func iter: 3059  Res: (0,944768998946 , -7,1669178332894)  Func: 0,10263198499 | Iter: 0  Iter: 0  Func iter: 0  Res: (0 , 1000)  Func: 1,10252350E-05 |
| 1e-7 | Iter: 11  Func iter: 312  Res: (2,79243964171326 , 1,95351270158543)  Func: 2,73983362643707 | Iter: 345  Func iter: 6081  Res: (0,69649450346496 , -6,75809090273154)  Func: 0,102636695910293 | Iter: 0  Func iter: 0  Res: (0 , 1000)  Func: 1,10252350E-05 |

1. **Путь решения**

*Метод Флетчера-Ривса*

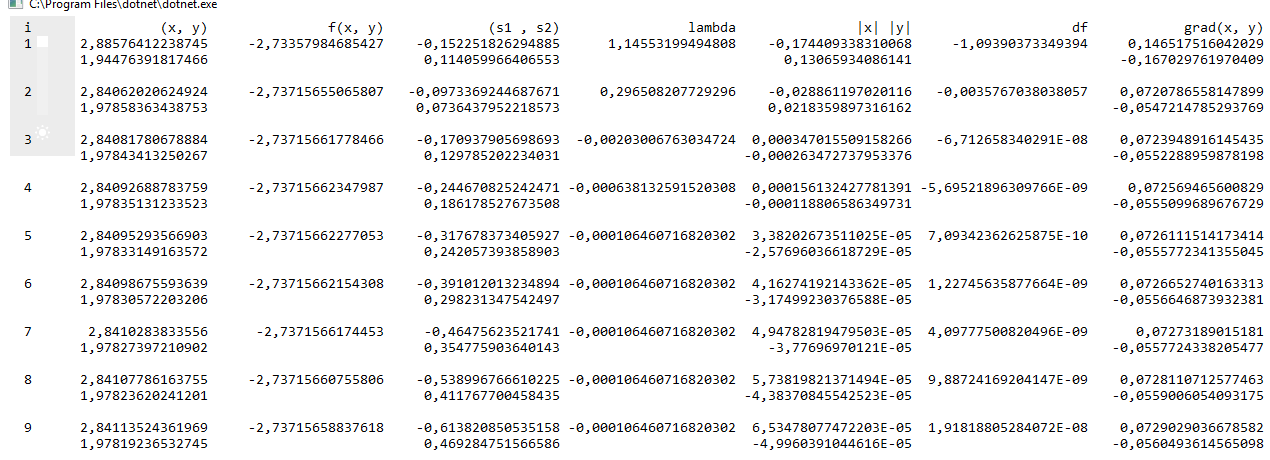
*Квадратичная функция*

*X0 = (3, 3)*

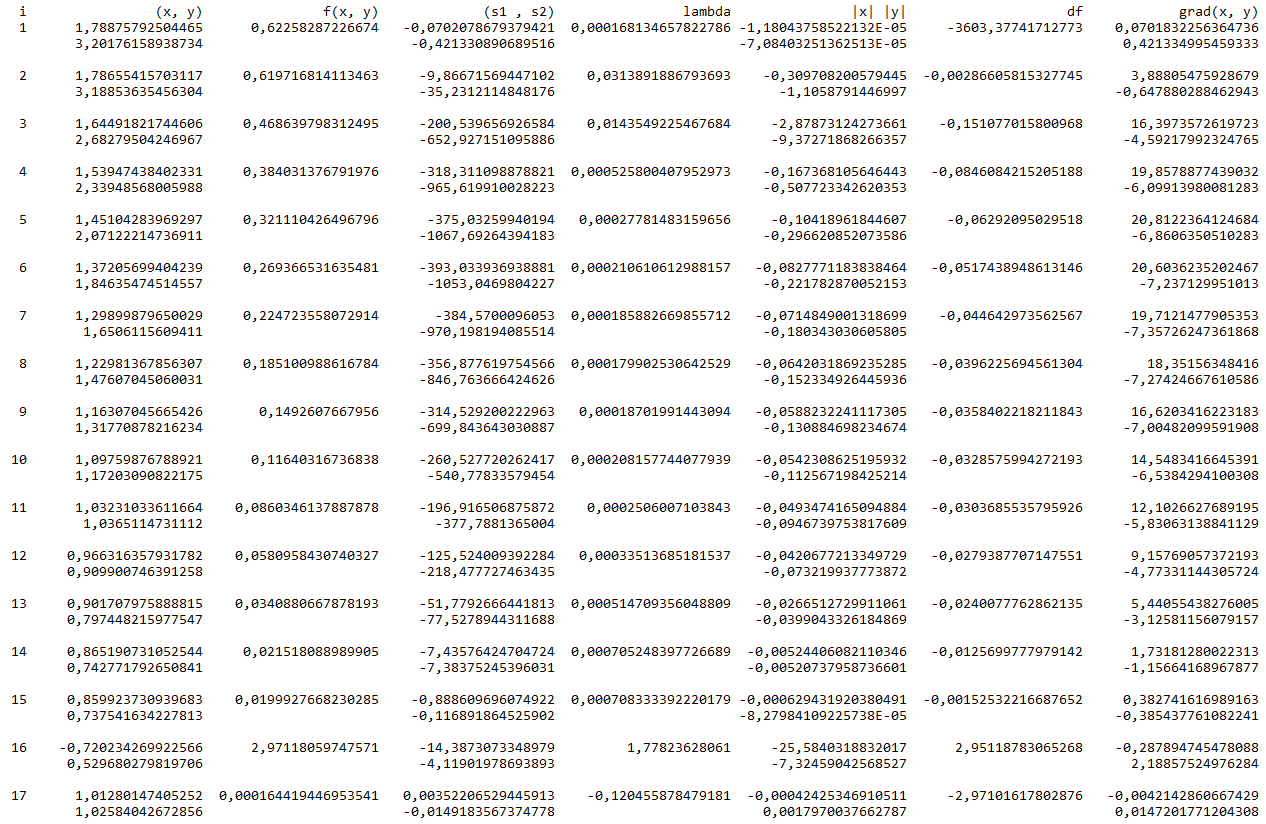


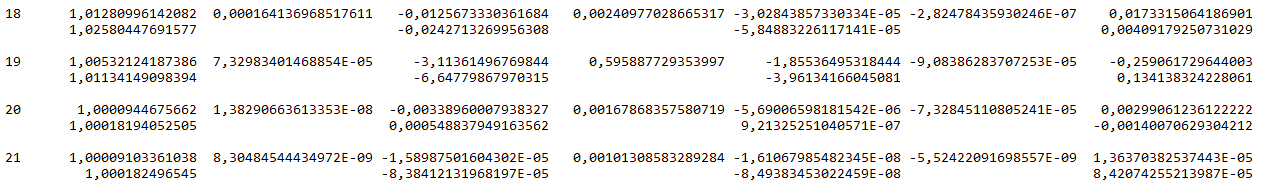
*Функция из задания*

*X0 = (3, 3)*



*Функция Розенброка*

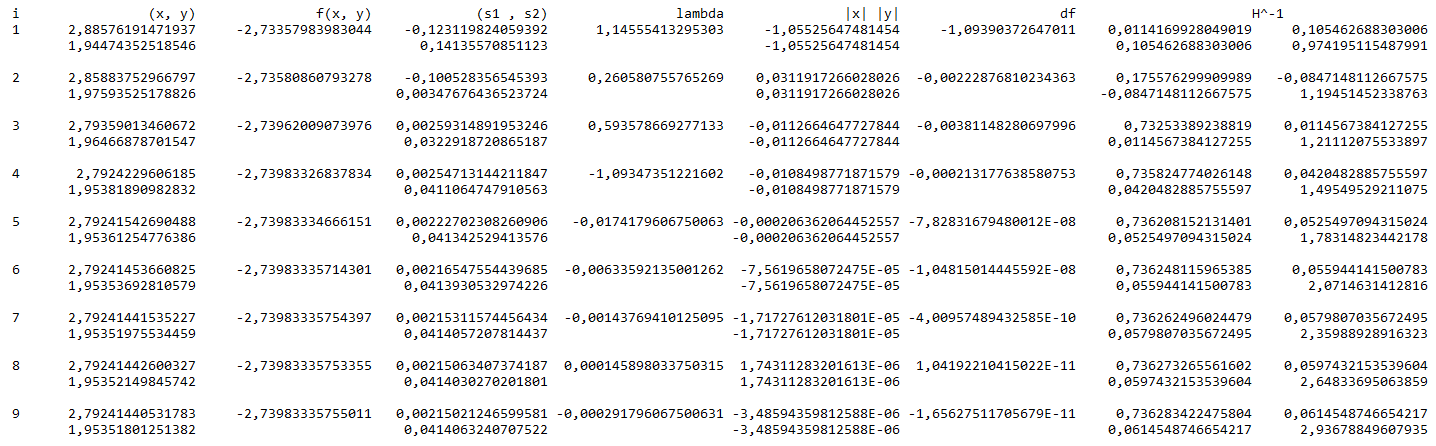




*Метод Девидона-Флетчера-Поуэлла*

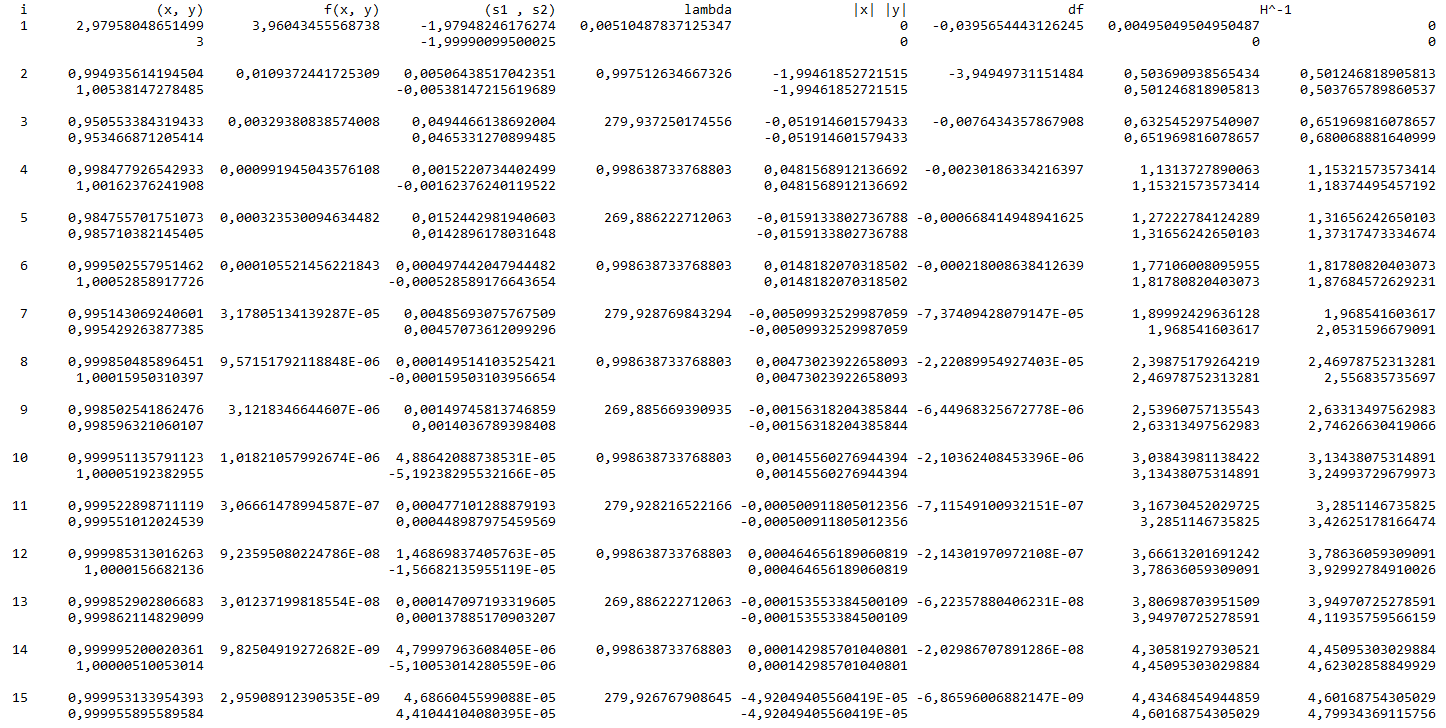
*Функция из задания*

X0 = (3, 3)



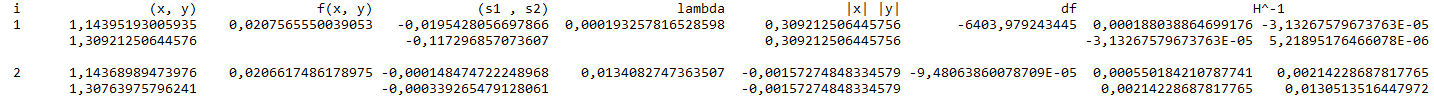
*Квадратичная функция*

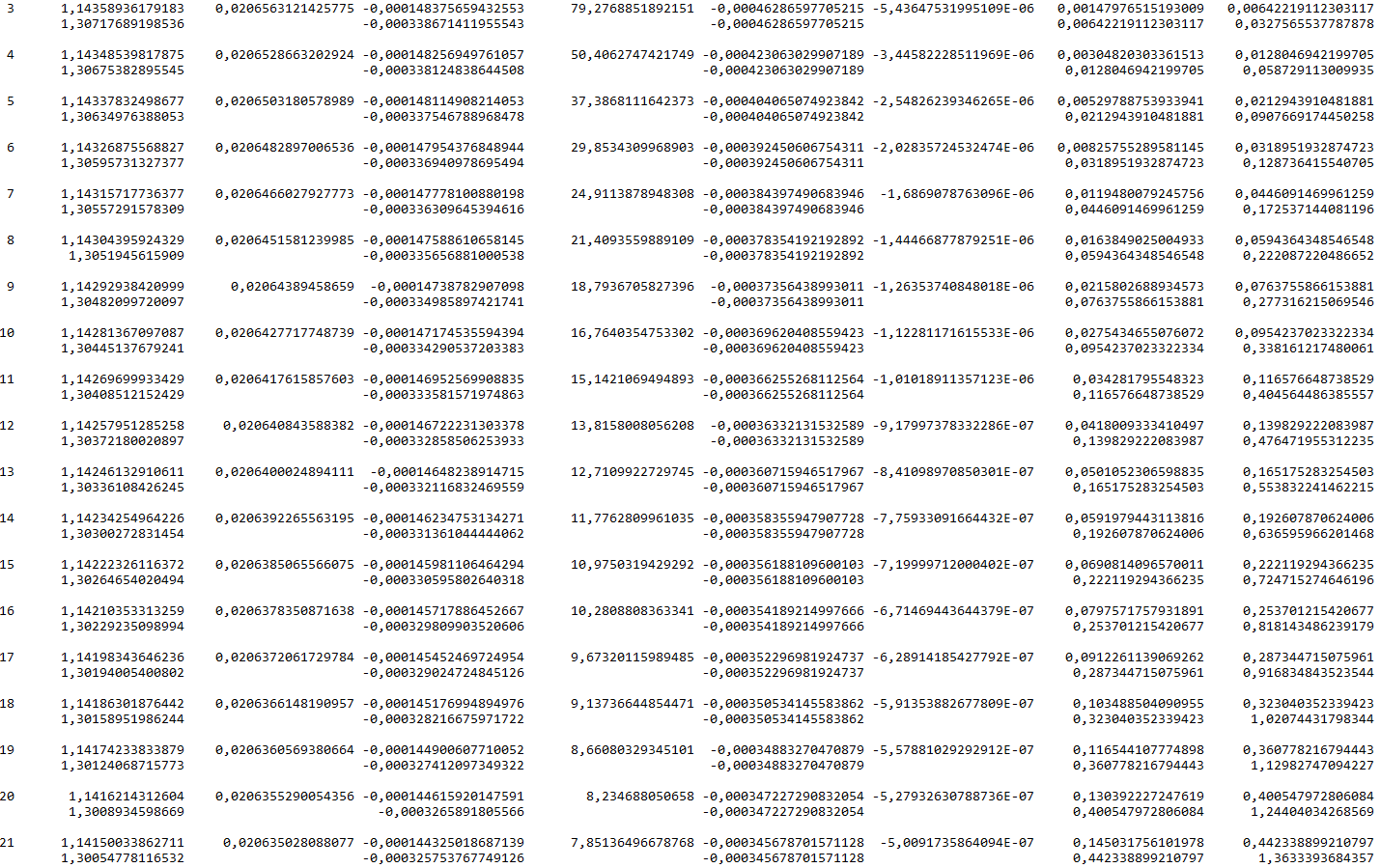
*X0 = (3, 3)*

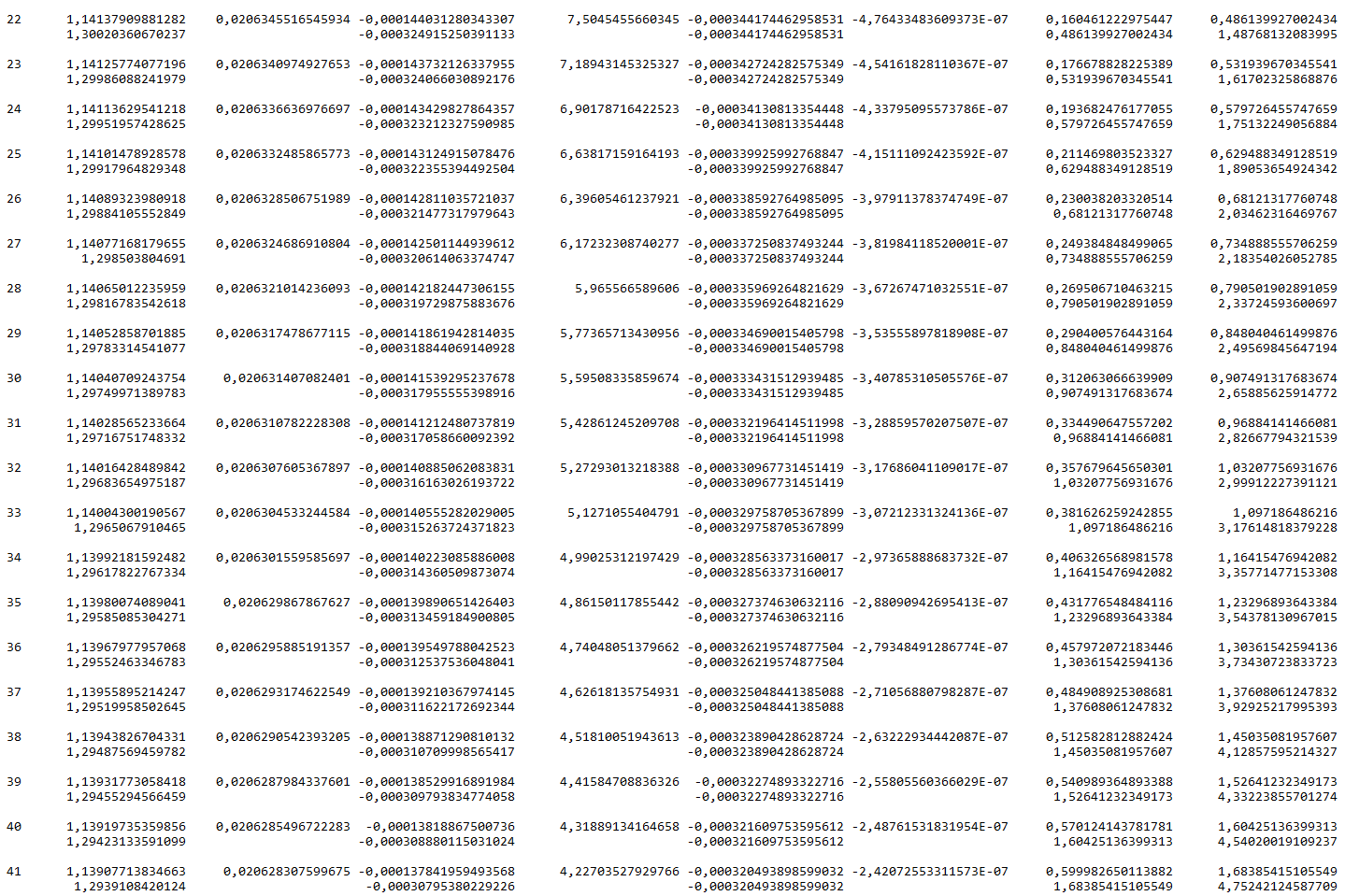


*Функция Розенброка*

*X0 = (3, 3)*





***…***

1. Вывод

Точность не прямо пропорциональна количеству итераций, иногда большая точность позволяет найти лучше минимум на прямой, из которого направление будет более точным.

Величина разности градиента напрямую зависит от скорости сходимости.

Количество итераций в методе Флтчера – Ривса почти не зависит от тосности

Количество итераций в методе Девидона-Флетчера-Паулла зависит от заданной функции(ухудшается матрица η)

Градиент в начальной точке не должен быть меньше eps

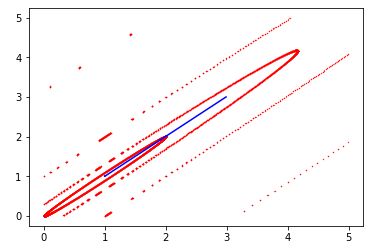
Начальные операции ищут больше 90% пути

Функция Розенброка плохая вещь

1. График поиска минимума

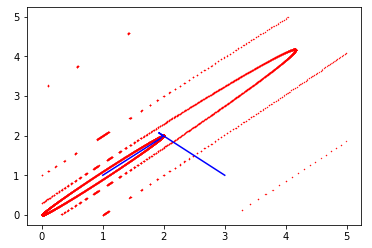
*МетодМетод Флетчера-Ривса и Девидона-Флетчера-Поуэлла*

*Квадратичная Функцияx0 = (3, 3)*

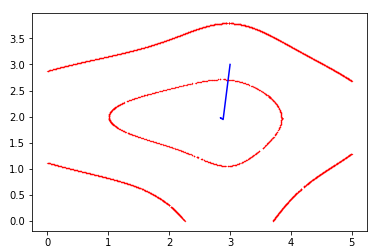


*Метод Флетчера-Ривса*

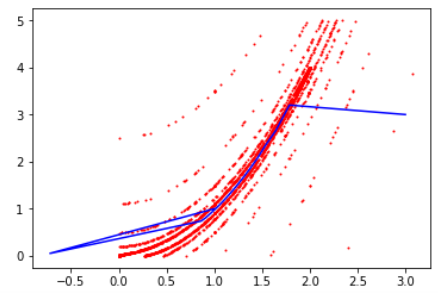
*КвадратичнаяФункцияx0 = (3, 1)*



*Функция из варианта x0 = (3, 3)*

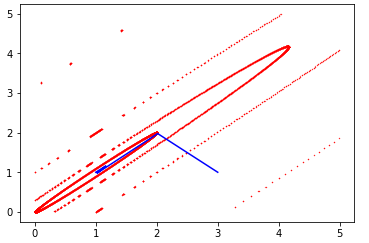


*Функция Розенброка x0 = (3, 3)*

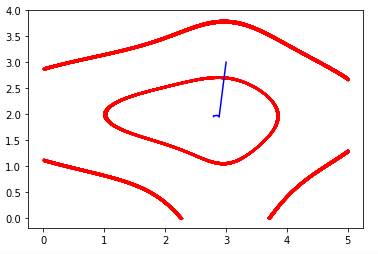


*Метод Девидона-Флетчера-Поуэлла*

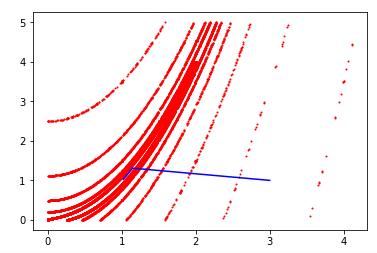
*Функция Квадратичнаяx0 = (3, 1)*



*Функция из варианта x0 = (3, 3)*

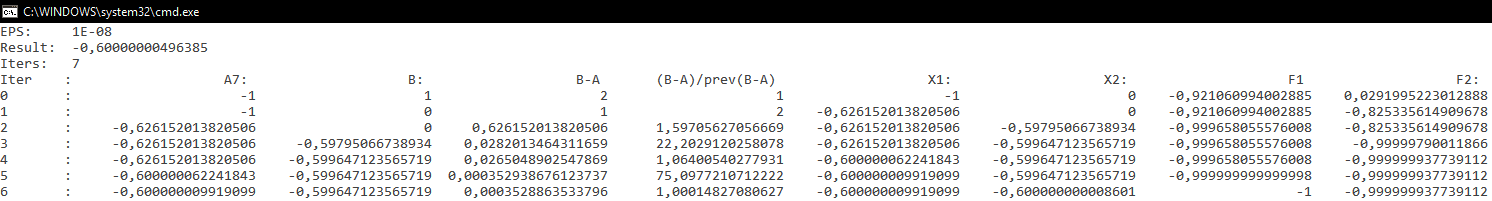


*Функция Розенброка x0 = (3, 1)*

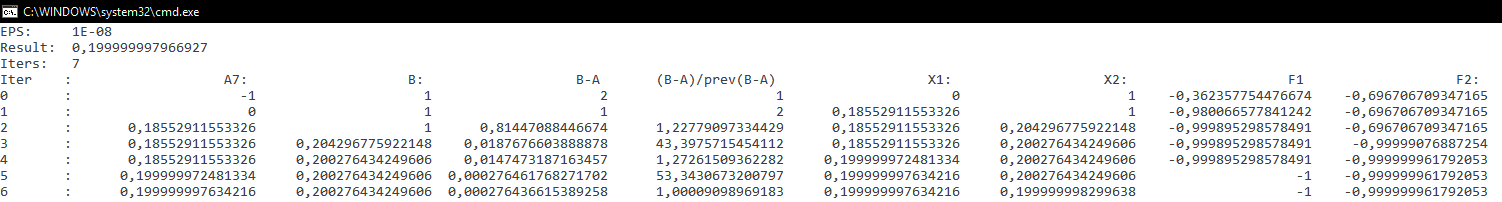


1. **Метод парабол для поиска минимума**

Функция x = -cos(x+0.6)



Функция x = -cos(x-0.2)

****

1. **Тест программы для метода парабол**

static void Main(string[] args)

{

#region 1 lab

Solver.EPS = 1e-8;

Console.WriteLine("EPS: " + Solver.EPS);

var res = Solver.Solve\_Parabola(-1, 1,x =>-Math.Cos(x-0.2));

Console.WriteLine("Result: " + res);

Console.WriteLine("Iters: " + Solver.LastIters);

Console.WriteLine(StepInfo.Title());

foreach (var item in Solver.LastSteps)

Console.WriteLine(item);}}

public static double Solve\_Parabola(double x1, double x3, Func<double, double> f)

{

LastSteps.Clear();

LastIters = 0;

var flag = true;

var bsuba = x3 - x1;

double x = (x3 + x1) / 2;

var x2 = (x3 + x1) / 2;

do

{

double f1 = f(x1), f2=f(x2), f3=f(x3);

double a0 = f1,

a1 = (f2 - f1) / (x2 - x1),

a2 = 1 / (x3 - x2) \* ((f3 - f1) / (x3 - x1) - (f2 - f1) / (x2 - x1));

double nx = 0.5 \* (x1 + x2 - a1 / a2);

if (nx != nx) break;

x = nx;

//if (x < x1 || x > x3)

double a = x1, b = x3;

if (x < x2) x3 = x2;

else x1 = x2;

LastSteps.Add(new StepInfo(LastIters++, a, b, bsuba, x1, x3, f1, f3));

bsuba = b - a;

if(x<=a || x >= b)

{

x = Math.Clamp(x, a, b);

x1 = x3 = x;

break;

}

x2 = x;

} while (x3-x1>EPS);

return x;

}

1. **Тест программы**

using System;

namespace csharp\_lab1

{

class Program

{

static void Main(string[] args)

{

Solver.Point p = Solver.Devidona\_Fletchera\_Paualla\_Metod(p2 => 100 \* Math.Pow(p2.y - Math.Pow(p2.x, 2), 2) + Math.Pow(1 - p2.x, 2), p1 => new Solver.Point(2 \* (200 \* Math.Pow(p1.x,3) - 200 \* p1.x \* p1.y + p1.x - 1), 200 \* (p1.y - Math.Pow(p1.x,2))), new Solver.Point(3, 1));

Console.WriteLine( p.x + " " + p.y );

Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace csharp\_lab1

{

public class Solver

{

public static void Solv(double x0, Func<double, double> f)

{

}

public static double EPS = 1e-3;

public static int LastIters;

public static int FuncIters;

public static List<StepInfo> LastSteps = new List<StepInfo>();

public struct Point

{

public Point(double x0 = 0, double y0 = 0)

{

x = x0; y = y0;

}

public double x { set; get; }

public double y { set; get; }

public double Norm()

{

return Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2));

}

public PointT Transp()

{

return new PointT(x, y);

}

public static Point operator -(Point a)

{

return new Point(-a.x, -a.y);

}

public static Point operator -(Point a, Point b)

{

return new Point(a.x - b.x, a.y - b.y);

}

public static Point operator \*(double b, Point a)

{

return new Point(b \* a.x, b \* a.y);

}

public static Point operator +(Point a, Point b)

{

return new Point(a.x + b.x, a.y + b.y);

}

public static double operator \*(PointT a, Point b)

{

return a.x \* b.x + a.y \* b.y;

}

public static Matrix operator \*(Point a, PointT b)

{

return new Matrix( a.x \* b.x, a.y \* b.x, a.x\*b.y, a.y \* b.y);

}

}

public struct PointT

{

public PointT(double x0 = 0, double y0 = 0)

{

x = x0; y = y0;

}

public double x { set; get; }

public double y { set; get; }

}

public struct Matrix

{

// xx xy

// yx yy

public Matrix(double x0 = 1, double x1 = 0, double y0 = 0, double y1 = 1)

{

xx = x0; xy = x1; yx = y0; yy = y1;

}

public double xx, xy, yx, yy;

public Matrix Transp()

{

return new Matrix(xx, yx, xy, yy);

}

public static Point operator \*(Matrix a, Point b)

{

return new Point(a.xx \* b.x + a.xy \* b.y, a.yx \* b.x + a.yy \* b.y);

}

public static Point operator \*(Point a, Matrix b)

{

return new Point(b.xx \* a.x + b.xy \* a.y, b.yx \* a.x + b.yy \* a.y);

}

public static Matrix operator -(Matrix a, Matrix b)

{

return new Matrix(a.xx - b.xx, a.xy - b.xy, a.yx - b.yx, a.yy - b.yy);

}

public static Matrix operator +(Matrix a, Matrix b)

{

return new Matrix(a.xx + b.xx, a.xy + b.xy, a.yx + b.yx, a.yy + b.yy);

}

public static Matrix operator / (Matrix a, double b)

{

return new Matrix(a.xx / b, a.xy / b, a.yx / b, a.yy / b);

}

}

#region lab2

static public Point Devidona\_Fletchera\_Paualla\_Metod(Func<Point, double> func, Func<Point, Point> Grad, Point p0 = new Point())

{

Point x = p0, z;

Point dx, dg;

Matrix nu = new Matrix(1, 0, 0, 1);

Matrix H = new Matrix(0, 0, 0, 0);

double lamda;

Point S = -Grad(x);

Point gradt = -S;

Point gradlast = gradt;

Func<double, double> f;

LastIters = 0;

FuncIters = 0;

double ft = func(x);

double flast = ft;

int ped = 22;

Console.WriteLine("i".PadLeft(4) + "(x, y)".PadLeft(ped) + "f(x, y)".PadLeft(ped) + "(s1 , s2)".PadLeft(ped) + "lambda".PadLeft(ped)+ "|x| |y|".PadLeft(ped) + "df".PadLeft(ped) + "".PadLeft(ped) + "H^-1".PadRight(ped));

while (gradt.Norm() > EPS)

{

lamda = 0;

f = landa => func(x + landa \* (nu \* S));

//Поиск отриезка содержащего минимум взять из лабы 1

Point p = Find\_Line(lamda, f);

//Поиск lamda при котормом f(x + lamda \* S) минимальна

lamda = Solve\_GoldSech(p.x, p.y, f);

if (lamda == 0 || Double.IsNaN(nu.xx))

break;

try

{

dx = lamda \* (nu \* S);

x += dx;

gradlast = gradt;

gradt = Grad(x);

dg = gradt - gradlast;

FuncIters += 4; // 4 раза для опредиления градиентат в точке

z = nu \* dg;

nu += (dx \* dx.Transp()) / (dx.Transp() \* dg) - (nu \* dg \* z.Transp()) / (z.Transp() \* dg);

H += (dx \* dx.Transp()) / (dx.Transp() \* dg);

S = -1.0 \* (nu \* gradt);

}

catch (InvalidCastException e)

{

break;

}

LastIters++;

ft = func(x);

Console.WriteLine(LastIters.ToString().PadLeft(4) +

x.x.ToString().PadLeft(ped) +

ft.ToString().PadLeft(ped) +

S.x.ToString().PadLeft(ped) +

lamda.ToString().PadLeft(ped) +

dx.y.ToString().PadLeft(ped) +

(ft - flast).ToString().PadLeft(ped) +

H.xx.ToString().PadLeft(ped) +

H.xy.ToString().PadLeft(ped)

);

Console.WriteLine(" ".PadLeft(4) +

x.y.ToString().PadLeft(ped) +

" ".PadLeft(ped) +

S.y.ToString().PadLeft(ped) +

" ".PadLeft(ped) +

dx.y.ToString().PadLeft(ped) +

" ".PadLeft(ped) +

H.yx.ToString().PadLeft(ped) +

H.yy.ToString().PadLeft(ped)

);

Console.WriteLine();

flast = ft;

}

Console.WriteLine("\n\nEPS: " + EPS);

Console.WriteLine("x0: \t(" + p0.x + " , " + p0.y + ")");

Console.WriteLine("Iter: " + LastIters);

Console.WriteLine("Func iter: " + FuncIters);

Console.WriteLine("Res: \t(" + x.x + " , " + x.y + ")");

Console.WriteLine("Func: " + func(x));

return x;

}

static public Point MSG\_Fletchera\_Riversa(Func<Point, double> func, Func<Point, Point> Grad, Point p0 = new Point())

{

Point x = p0;

Point S = -Grad(x);

LastIters = 0;

FuncIters = 0;

double w;

double lamda;

Func<double, double> f;

Point gradt = -S;

Point gradlast = gradt;

int ped = 22;

Console.WriteLine("i".PadLeft(4) + "(x, y)".PadLeft(ped) + "f(x, y)".PadLeft(ped) + "(s1 , s2)".PadLeft(ped) + "lambda".PadLeft(ped) + "|x| |y|".PadLeft(ped) + "df".PadLeft(ped) + "grad(x, y)".PadLeft(ped));

double ft = func(x);

double flast = ft;

while (S.Norm() > EPS)

{

lamda = 3;

f = landa => func(x + landa \* S);

//Поиск отриезка содержащего минимум взять из лабы 1

Point p = Find\_Line(lamda, f);

//Поиск lamda при котормом f(x + lamda \* S) минимальна

lamda = Solve\_GoldSech(p.x, p.y, f);

if (lamda == 0 || Double.IsNaN(S.x) || Double.IsNaN(S.y))

break;

x += lamda \* S;

gradlast = gradt;

if (gradlast.Norm() == 0)

break;

gradt = Grad(x);

FuncIters += 4; // 4 раза для опредиления градиентат в точке

if (gradlast.Norm() != 0)

{

w = Math.Pow(gradt.Norm() / gradlast.Norm(), 2); ;

S = -gradt + w \* S;

}

LastIters++;

flast = ft;

ft = func(x);

Console.WriteLine(LastIters.ToString().PadLeft(4) +

x.x.ToString().PadLeft(ped) +

ft.ToString().PadLeft(ped) +

S.x.ToString().PadLeft(ped) +

lamda.ToString().PadLeft(ped) +

(lamda \* S).x.ToString().PadLeft(ped) +

(ft - flast).ToString().PadLeft(ped) +

gradt.x.ToString().PadLeft(ped)

);

Console.WriteLine(" ".PadLeft(4) +

x.y.ToString().PadLeft(ped) +

" ".PadLeft(ped) +

S.y.ToString().PadLeft(ped) +

" ".PadLeft(ped) +

(lamda \* S).y.ToString().PadLeft(ped) +

" ".PadLeft(ped) +

gradt.y.ToString().PadLeft(ped)

);

Console.WriteLine();

}

Console.WriteLine("EPS: " + EPS);

Console.WriteLine("x0: \t(" + p0.x + " , " + p0.y + ")");

Console.WriteLine("Iter: " + LastIters);

Console.WriteLine("Function iter: " + FuncIters);

Console.WriteLine("Res: \t(" + x.x + " , " + x.y + ")");

Console.WriteLine("Func: " + func(x));

return x;

}

#endregion

static private Point Gradient(Func<Point, double> func, Point p)

{

Point g;

double h = EPS / 20;

g = new Point( (func(new Point(p.x + h, p.y)) - func(new Point(p.x - h, p.y))) / (2\*h) , (func(new Point(p.x, p.y+h)) - func(new Point(p.x, p.y - h))) / (2 \* h));

return g;

}

}