Лабораторная работа 4	ФИО студента	Титов А.К.
	Группа	ИВТ - 360
	Методы	Метод штрафных функций
	Задачи	1.3 + 3 учебных задачи на выбор
Методы штрафных функций	Дата отчета	
	Оценка	
	Подпись преподавателя	

Задание

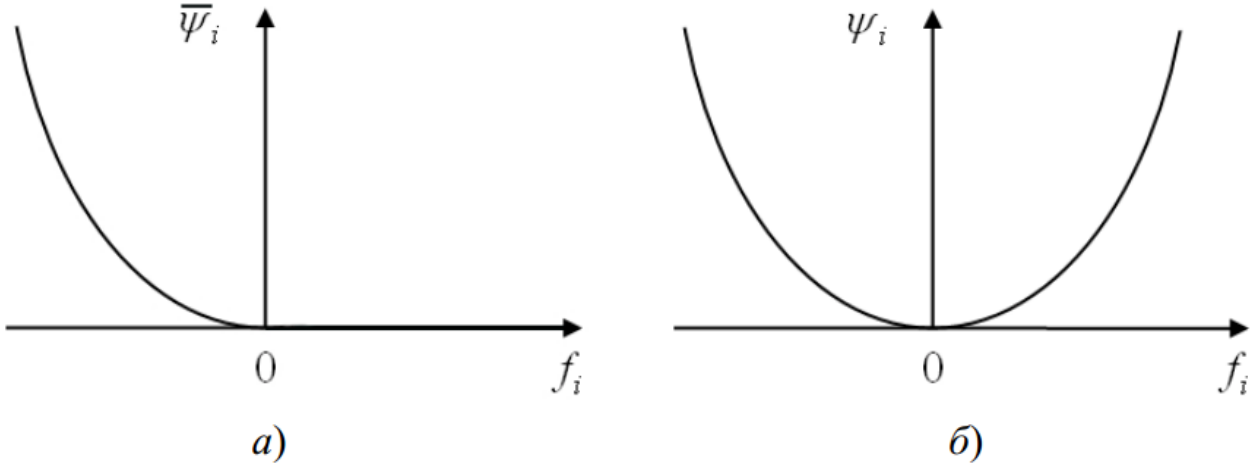
Реализовать программно следующие методы последовательной безусловной оптимизации

- Метод штрафных функций

Описание решения

В методическом пособии к лабораторной работе алгоритм был дан в форме, удобной для написания программы. Был использован алгоритм Хука - Дживса из 2 лабораторной.

Функции для ограничения-неравенств и ограничений-равенств



а) *б)*

Рис. 1

$\overline{\psi}_i(f_i(x)) = -\min(0, f_i(x))$ (рис. 2, *а*)) вместе с $\psi_i(f_i(x)) = |f_i(x)|$ (рис. 2, *б*)).

Примечание: Весь вывод программы не приводится в виду его объемности. Приводится лишь крайняя его часть с результатом работы алгоритма.

Результаты работы программы (задача 1.3)

3. Методом штрафных или барьерных функций минимизировать функцию

$f(x) = x_1^2 + x_2^2$

при ограничениях

$f_1(x) = x_1 - 1 \geq 0$;

$f_2(x) = 2 - x_1 - x_2 \geq 0$.

Начальные точки:

$x_0 = (0,1)^T$, $x_0 = (2,2)^T$

Точное решение: $x^* = (1, 0)^T$, $f^* = 1$ (рис. 6). Ожидаемый результат: приближенное решение

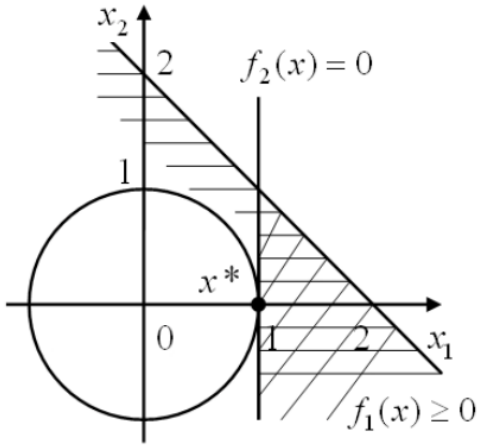


Рис. 6

Тестирующая функция

```
FunctionND funcND = (PointN point) => { return System.Math.Pow(point.At(0), 2) +
System.Math.Pow(point.At(1), 2); }; // z = x^2 + y^2
PointN x0   = new PointN(0.0, 1.0);
PointN x0_1 = new PointN(2.0, 2.0); // доп. вариант базовой точки
double eps = 0.01;

// Функции ограничений
List<FunctionND> limitFunctions = new List<FunctionND>
{
  new FunctionND((PointN point) => { return point.At(0) - 1; }),
  new FunctionND((PointN point) => { return 2 - point.At(0) - point.At(1); }),
};
int m = 2; // Число функций ограничений типа Fi(x) >= 0
int p = 2; // Общее число функций ограничений

TestPenaltyFunctionMethod(funcND, limitFunctions, x0_1, m, p, eps);
```

Часть вывода программы

```
==================Next Iteration====================
>>> ++++++++++++++++++++HookJivs started++++++++++++++++++++
> Exploring search ( xk{ 0,992000 0,000000 } ) = b2{ 0,992000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,992000 0,000000 }
> Exploring search ( xk{ 0,992000 0,000000 } ) = x{ 0,992000 0,000000 }
> Exploring search ( xk{ 0,992000 0,000000 } ) = b2{ 0,992000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,992000 0,000000 }
> Exploring search ( xk{ 0,992000 0,000000 } ) = x{ 0,992000 0,000000 }
> Exploring search ( xk{ 0,992000 0,000000 } ) = b2{ 1,002000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,992000 0,000000 }
> Exploring search ( xk{ 1,012000 0,000000 } ) = x{ 1,002000 0,000000 }
> Exploring search ( xk{ 1,002000 0,000000 } ) = b2{ 1,001000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 1,002000 0,000000 }
> Exploring search ( xk{ 1,000000 0,000000 } ) = x{ 0,999000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 1,000000 0,000000 }
> Exploring search ( xk{ 0,997000 0,000000 } ) = x{ 0,998000 0,000000 }
> Exploring search ( xk{ 0,999000 0,000000 } ) = b2{ 0,999000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,999000 0,000000 }
> Exploring search ( xk{ 0,999000 0,000000 } ) = x{ 0,999000 0,000000 }
>>> ++++++++++++++++++++HookJivs finished++++++++++++++++++++
P(xrk, rk) = 0,000777600000006563
==================Next Iteration====================


-----------------------------------------
Variable metric method: x_min = { 0,999000 0,000000 }
F(x_min) = 0,998000999999992
=========================================
```

Результаты работы программы (задача 2.1)

2.1. Минимизировать функцию

$$f(x) = \frac{1}{3}(x_1 + 1)^3 + x_2$$

при ограничениях

$x_1 - 1 \geq 0$, $x_2 \geq 0$.

Решение: $x^* = (1, 0)^T$, $f(x^*) = 2\frac{2}{3}$.

Тестирующая функция

```
FunctionND funcND = (PointN point) => { return (1.0 / 3.0) * System.Math.Pow(point.At(0) + 1, 3); };
// z = (1/3)*(x+1)^3 + y
PointN x0 = new PointN(0.0, 0.0);
double eps = 0.01;

// Функции ограничений
List<FunctionND> limitFunctions = new List<FunctionND>
{
  new FunctionND((PointN point) => { return point.At(0) - 1; }),
  new FunctionND((PointN point) => { return point.At(1); }),
};
int m = 2; // Число функций ограничений типа Fi(x) >= 0
int p = 2; // Общее число функций ограничений

TestPenaltyFunctionMethod(funcND, limitFunctions, x0, m, p, eps);
```

Часть вывода программы

```
==================Next Iteration====================
>>> +++++++++++++++++++HookJivs started++++++++++++++++++++
> Exploring search ( xk{ 0,997000 0,000000 } ) = b2{ 0,997000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,997000 0,000000 }
> Exploring search ( xk{ 0,997000 0,000000 } ) = x{ 0,997000 0,000000 }
> Exploring search ( xk{ 0,997000 0,000000 } ) = b2{ 0,997000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,997000 0,000000 }
> Exploring search ( xk{ 0,997000 0,000000 } ) = x{ 0,997000 0,000000 }
> Exploring search ( xk{ 0,997000 0,000000 } ) = b2{ 1,007000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,997000 0,000000 }
> Exploring search ( xk{ 1,017000 0,000000 } ) = x{ 1,007000 0,000000 }
> Exploring search ( xk{ 1,007000 0,000000 } ) = b2{ 1,006000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 1,007000 0,000000 }
> Exploring search ( xk{ 1,005000 0,000000 } ) = x{ 1,004000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 1,005000 0,000000 }
> Exploring search ( xk{ 1,002000 0,000000 } ) = x{ 1,001000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 1,002000 0,000000 }
> Exploring search ( xk{ 0,998000 0,000000 } ) = x{ 0,999000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,998000 0,000000 }
> Exploring search ( xk{ 0,997000 0,000000 } ) = x{ 0,998000 0,000000 }
> Exploring search ( xk{ 0,999000 0,000000 } ) = b2{ 1,000000 0,000000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,999000 0,000000 }
> Exploring search ( xk{ 1,001000 0,000000 } ) = x{ 1,000000 0,000000 }
>>> +++++++++++++++++++HookJivs finished++++++++++++++++++++
P(xrk, rk) = 0
==================Next Iteration====================


-------------------------------------------
Variable metric method: x_min = { 1,000000 0,000000 }
F(x_min) = 2,66666666666667
===========================================
```

Результаты работы программы (задача 2.5)

2.5. Минимизировать функцию

$$f(x) = 4x_1^2 + 8x_1 - x_2 - 3$$

при ограничении

$$x_1 + x_2 = -2.$$

Решение: $x^* = (-1.125, -0.875)^T$, $f(x^*) = -6.0625$.

Тестирующая функция

```
FunctionND funcND = (PointN point) => { return 4 * point.At(0) * point.At(0) + 8 * point.At(0) -
point.At(1) - 3; }; // z = x^2 + y^2
PointN x0 = new PointN(0.0, 0.0);
double eps = 0.01;

// Функции ограничений
List<FunctionND> limitFunctions = new List<FunctionND>
{
  new FunctionND((PointN point) => { return point.At(0) + point.At(1) + 2; })
};
int m = 0; // Число функций ограничений типа Fi(x) >= 0
int p = 1; // Общее число функций ограничений

TestPenaltyFunctionMethod(funcND, limitFunctions, x0, m, p, eps);
```

Часть вывода программы

```
=================Next Iteration===================
>>> +++++++++++++++++++HookJivs started+++++++++++++++++++
> Exploring search ( xk{ -1,132000 -0,864000 } ) = b2{ -1,132000 -0,864000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,132000 -0,864000 } ) = x{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,132000 -0,864000 } ) = b2{ -1,132000 -0,864000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,132000 -0,864000 } ) = x{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,132000 -0,864000 } ) = b2{ -1,132000 -0,864000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,132000 -0,864000 } ) = x{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,132000 -0,864000 } ) = b2{ -1,133000 -0,865000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,132000 -0,864000 }
> Exploring search ( xk{ -1,134000 -0,866000 } ) = x{ -1,133000 -0,866000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,134000 -0,866000 }
> Exploring search ( xk{ -1,133000 -0,867000 } ) = x{ -1,132000 -0,867000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,133000 -0,867000 }
> Exploring search ( xk{ -1,131000 -0,868000 } ) = x{ -1,131000 -0,868000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,131000 -0,868000 }
> Exploring search ( xk{ -1,130000 -0,869000 } ) = x{ -1,130000 -0,869000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,130000 -0,869000 }
> Exploring search ( xk{ -1,129000 -0,870000 } ) = x{ -1,129000 -0,870000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,129000 -0,870000 }
> Exploring search ( xk{ -1,128000 -0,871000 } ) = x{ -1,128000 -0,871000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,128000 -0,871000 }
> Exploring search ( xk{ -1,127000 -0,872000 } ) = x{ -1,127000 -0,872000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,127000 -0,872000 }
> Exploring search ( xk{ -1,126000 -0,873000 } ) = x{ -1,126000 -0,873000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,126000 -0,873000 }
> Exploring search ( xk{ -1,125000 -0,874000 } ) = x{ -1,125000 -0,874000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,125000 -0,874000 }
> Exploring search ( xk{ -1,124000 -0,875000 } ) = x{ -1,124000 -0,875000 }
> Exploring search ( xk{ -1,125000 -0,874000 } ) = b2{ -1,125000 -0,874000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ -1,125000 -0,874000 }
> Exploring search ( xk{ -1,125000 -0,874000 } ) = x{ -1,125000 -0,874000 }
>>> +++++++++++++++++++HookJivs finished+++++++++++++++++++
P(xrk, rk) = 0,000777599999999829
=================Next Iteration===================

---------------------------------------------
Variable metric method: x_min = { -1,125000 -0,874000 }
F(x_min) = -6,0635
=============================================
```

Результаты работы программы (задача 2.17)

2.17. Задача Розена-Сузуки

Минимизировать функцию

$$f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

при ограничениях

$$8 - x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4 > 0,$$

$$10 - x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 - x_4 > 0,\ 5 - 2x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4 > 0.$$

Решение: $x^* = (0, 1, 2, -1)^T$, $f(x^*) = -44$.

Тестирующая функция

```
FunctionND funcND = (PointN p) => { return SMath.Pow(p.At(0), 2)
                                    + SMath.Pow(p.At(1), 2)
                                    + 2 * SMath.Pow(p.At(2), 2)
                                    + SMath.Pow(p.At(3), 2)
                                    - 5 * p.At(0)
                                    - 5 * p.At(1)
                                    - 21 * p.At(2)
                                    + 7 * p.At(3); };
PointN x0 = new PointN(0.0, 0.0, 0.0, 0.0);
double eps = 0.01;

// Функции ограничений
List<FunctionND> limitFunctions = new List<FunctionND>
{
  new FunctionND((PointN p) => { return 8 - SMath.Pow(p.At(0),2)
                                       - SMath.Pow(p.At(1), 2)
                                       - SMath.Pow(p.At(2), 2)
                                       - SMath.Pow(p.At(3), 2)
                                       - p.At(0)
                                       + p.At(1)
                                       - p.At(2)
                                       + p.At(3); }),
  new FunctionND((PointN p) => { return 10 - SMath.Pow(p.At(0),2)
                                       - 2 * SMath.Pow(p.At(1),2)
                                       - SMath.Pow(p.At(2), 2)
                                       - 2 * SMath.Pow(p.At(3), 2)
                                       + p.At(0)
                                       - p.At(3); }),
  new FunctionND((PointN p) => { return 5 - 2 * SMath.Pow(p.At(0),2)
                                       - SMath.Pow(p.At(1),2)
                                       - SMath.Pow(p.At(2),2)
                                       - 2 * p.At(0)
                                       + p.At(1)
                                       + p.At(3); })

};
int m = 3; // Число функций ограничений типа Fi(x) >= 0
int _p = 3; // Общее число функций ограничений

TestPenaltyFunctionMethod(funcND, limitFunctions, x0, m, _p, eps);
```

Часть вывода программы

```
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,002000 1,002000 1,998000 -1,002000 }
> Exploring search ( xk{ 0,005000 0,998000 1,998000 -1,002000 } ) = x{ 0,005000 0,998000 1,998000 -1
,002000 }
> Exploring search ( xk{ 0,003000 1,001000 1,998000 -1,002000 } ) = b2{ 0,003000 1,001000 1,998000 -
1,002000 }
> Doing step xk = b1 + (b2 - b1) * 2, will give xk{ 0,003000 1,001000 1,998000 -1,002000 }
> Exploring search ( xk{ 0,003000 1,001000 1,998000 -1,002000 } ) = x{ 0,003000 1,001000 1,998000 -1
,002000 }
>>> ++++++++++++++++++++HookJivs finished++++++++++++++++++++
P(xrk, rk) = 0,00081403289286763
==================Next Iteration====================


------------------------------------------
Variable metric method: x_min = { 0,003000 1,001000 1,998000 -1,002000 }
F(x_min) = -44,0019780000001
==========================================
```

Код программы

```
namespace ConditionalOptimization.Math
{
  using FunctionND = System.Func<PointN, double>;
  public class PenaltyFunctionMethod
  {
    // Input data
    private FunctionND f;
    private List<FunctionND> limitFunctions;
    private PointN x0;
    private double eps;
    private int m;
    private int p;
    private double r0;
    private double z;

    public List<string> Log;
    public PenaltyFunctionMethod(FunctionND f, List<FunctionND> limitFunctions, PointN x0, int m,
int p, double r0 = 0.1, double z = 6)
    {
      this.f = f;
      this.limitFunctions = limitFunctions;
      this.x0 = x0;
      this.r0 = r0;
      this.z = z;
      this.m = m;
      this.p = p;

      Log = new List<string>(); // TODO: здесь концепция Log не работает
    }

    public PointN FindMin(double eps)
    {
      int k = 0;
      int dimensionsCount = x0.Coordinates.Count;

      // Промежуточные данные
      PointN xk = new PointN(x0);
      PointN x_rk;
      double rk = r0;

      do
      {
        // Поиск минимума F(x, r)
        F Fxr = new F((PointN point) => { return F(point, rk); });
        Console.WriteLine(">>> ++++++++++++++++++++HookJivs started++++++++++++++++++++");
        x_rk = HookJivs.MethodHookJivs(Fxr, xk, new VectorN(dimensionsCount, 1.0), eps);
        Console.WriteLine(">>> ++++++++++++++++++++HookJivs finished++++++++++++++++++++");

        // Значение штрафной функции в xrk
        double P_x_rk = P(x_rk, rk); // TODO в данных условиях не имеет смысла, т.к. нигде далее не
используется и пересчитывается заново
        Console.WriteLine("P(xrk, rk) = {0}", P_x_rk);  //Log.Add(String.Format("P(xrk, rk) = {0}",
P_x_rk));

        // Инкремент цикла
        rk = rk * z;
        xk = x_rk;
        k++;
        Console.WriteLine("==================Next Iteration====================");
//Log.Add("==================Next Iteration====================");
      } while (P(x_rk, rk) >= eps);

      return x_rk;
    }
```

```
    // Вспомогательная функция
    private double F(PointN x, double rk) {  return P(x, rk) + f(x);  }


    // Штрафная функция (Penalty function)
    private double P(PointN x, double rk)
    {
      double sumFi  = 0;
      double sum_Fi = 0;

      // Цикл по всем функциям органичений типа F(x) >= 0
      for (int i = 0; i < m; i++)
        sumFi += _PFi(x, i);

      // Цикл по всем функциям органичений типа F(x) = 0
      for (int i = m; i < p; i++)
        sum_Fi += PFi(x, i);

      return rk * (sumFi + sum_Fi);
    }


    // Схема с параболами
    private double _PFi(PointN x, int index)
    {
      return System.Math.Pow(System.Math.Min(0, limitFunctions[index](x)), 2);
    }

    private double PFi(PointN x, int index)
    {
      return System.Math.Pow(limitFunctions[index](x), 2);
    }
  }
}
```