|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 3 | | |
| по дисциплине «Методы оптимизации» | | |
| МЕТОД ШТРАФНЫХ ФУНКЦИЙ | | |
|  | | |
|  | Бригада 1 | Исакин Даниил |
| Группа ПМ-13 | Вострецова Екатерина |
| Вариант 1 |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | Лемешко борис юрьевич |
|  |  |
| Новосибирск, 2024 | | |

1. **Цель**

Ознакомиться с методами штрафных функций при решении задач нелинейного программирования. Изучить типы штрафных и барьерных функций, их особенности, способы и области применения, влияние штрафных функций на сходимость алгоритмов, зависимость точности решения задачи нелинейного программирования от величины коэффициента штрафа.

1. **Задание**

Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с использованием метода штрафных функций

Исследовать сходимость метода штрафных функций в зависимости:

* от выбора штрафных функций,
* начальной величины коэффициента штрафа,
* стратегии изменения коэффициента штрафа,
* начальной точки,
* задаваемой точности

Сформулировать выводы.

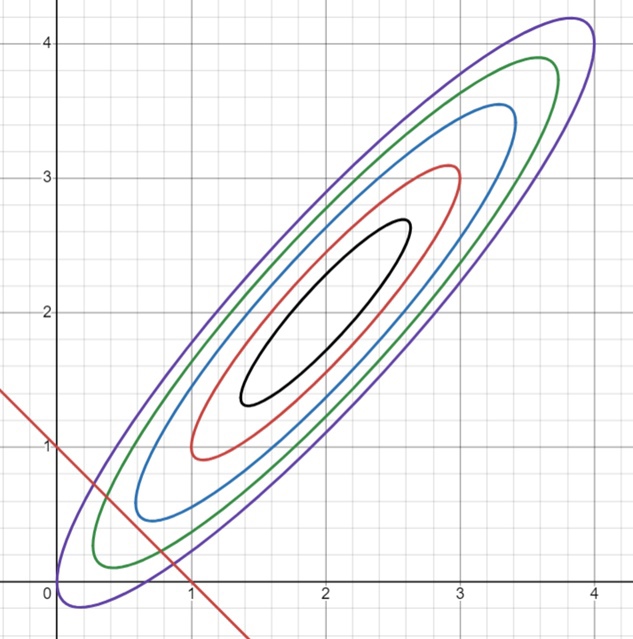
Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с ограничением типа неравенства (только задача а) с использованием метода барьерных функций.

Исследовать сходимость метода барьерных функций (только задача а) в зависимости:

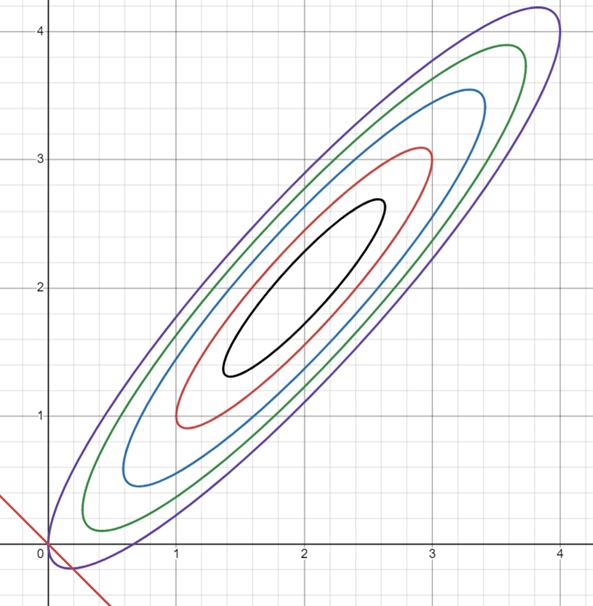
* от выбора барьерных функций,
* начальной величины коэффициента штрафа,
* стратегии изменения коэффициента штрафа,
* начального приближения,
* задаваемой точности.

Сформулировать выводы

Вид функции сверху при первом ограничении:



Вид функции сверху при втором граничном условии:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Первая задача (а) | | | Вторая задача (б) | | |
|  | | |  | | |
|  |  |  |  |  |  |
| 0.571429 | 0.428571 | 2.14286 | 0.095238 | -0.095238 | 3.80952 |

1. **Результаты исследования для штрафных функций**

**Зависимость от выбора штрафных функций**

– штрафные функции

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– начальное приближение

– точность

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Количество итераций | Число вычислений целевой функции |  |  |  |
| 1 | 1 | 2 | 8738 | 0.0914077 | -0.0976010 | 3.82135 |
| 2 | 1 | 4 | 25228 | 0.0993768 | -0.0920074 | 3.79551 |
| 4 | 1 | 4 | 25720 | 0.0988980 | -0.0913616 | 3.79518 |
| 1 | 2 | 19 | 92818 | 0.0958619 | -0.0945512 | 3.80703 |
| 2 | 2 | 20 | 99520 | 0.0958619 | -0.0945512 | 3.80703 |
| 4 | 2 | 20 | 99520 | 0.0958619 | -0.0945512 | 3.80703 |

Выбор штрафной функции влияет на количество совершенных итераций и вычислений целевой функции: чем больше степень функции штрафа – тем больше вычислений. При этом для функции штрафа для ограничения равенство результат более точный при .

**Зависимость от начальной величины коэффициента штрафа**

– штрафные функции

– стратегия изменения коэффициента штрафа

– начальное приближение

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
| 0.001 | 30 | 155966 | 0.0958620 | -0.0945518 | 3.80703 |
| 0.1 | 22 | 105992 | 0.0958617 | -0.0945513 | 3.80703 |
| 1 | 19 | 92818 | 0.0958619 | -0.0945512 | 3.80703 |
| 10 | 16 | 90888 | 0.0958617 | -0.0945504 | 3.80703 |
| 50 | 14 | 88934 | 0.0958619 | -0.0945512 | 3.80703 |
| 100 | 12 | 77374 | 0.0958604 | -0.0945478 | 3.80703 |

Чем больше значение начального коэффициента штрафа, тем быстрее сходится решение (меньше итераций и меньше вычислений целевой функции).

**Зависимость от стратегии изменения коэффициента штрафа**

– штрафные функции

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– начальное приближение

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 19 | 92818 | 0.0958619 | -0.0945512 | 3.80703 |
|  | 9 | 47438 | 0.0969385 | -0.093366 | 3.80272 |
|  | 6 | 32954 | 0.0980591 | -0.092135 | 3.79825 |
|  | 5 | 21688 | 0.103866 | -0.0857463 | 3.77509 |
|  | 3 | 12408 | 0.122977 | -0.064724 | 3.69937 |
| 0 | 48 | 77656 | 0.0961623 | -0.0941637 | 3.80572 |
|  | 32 | 165038 | 0.0953464 | -0.0951106 | 3.80907 |

Прибавление к коэффициентам константы дает результат хуже, чем линейная или степенная зависимость. Линейная зависимость даёт результат хуже, чем степенная. Чем больше коэффициент при коэффициенте штрафа тем меньше количество итераций и количество вычислений целевой функции. В степенной зависимости то же самое касается степени.

**Зависимость от начальной точки**

– штрафные функции

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 18 | 87602 | 0.0958633 | -0.0945524 | 3.80703 |
|  | 19 | 92818 | 0.0958619 | -0.0945512 | 3.80703 |
|  | 19 | 93696 | 0.0958624 | -0.0945519 | 3.80703 |
|  | 19 | 94870 | 0.0958632 | -0.0945540 | 3.80703 |
|  | 19 | 95242 | 0.0958619 | -0.0945510 | 3.80703 |
|  | 19 | 96910 | 0.0958633 | -0.0945540 | 3.80703 |

Чем ближе начальное приближение к искомому значению, тем меньше количество вычислений целевой функции.

**Зависимость от задаваемой точности**

– штрафные функции

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– начальное приближение

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 19 | 4398 | 0.0949402 | -0.0939519 | 3.80765 |
|  | 19 | 82760 | 0.0957876 | -0.0946329 | 3.80732 |
|  | 19 | 92818 | 0.0958619 | -0.0945512 | 3.80703 |
|  | 20 | 145256 | 0.0958616 | -0.0945522 | 3.80703 |

С увеличением задаваемой точности увеличивается точность решения, однако растет количество итераций и вычислений целевой функции.

1. **Результаты исследования для барьерных функций (задача а)**

**Зависимость от выбора барьерной функций**

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– начальное приближение

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 29 | 131596 | 0.571409 | 0.428503 | 2.14298 |
|  | 10 | 30794 | 0.571673 | 0.428671 | 2.14237 |

Логарифмическая функция быстрее сошлась и потребовала меньшее количество вычислений целевой функции.

**Зависимость от начальной величины коэффициента штрафа**

– барьерная функция

– стратегия изменения коэффициента штрафа

– начальное приближение

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
| 0.001 | 3 | 13506 | 0.587078 | 0.413046 | 2.14778 |
| 0.1 | 3 | 14710 | 0.571638 | 0.428574 | 2.14255 |
| 1 | 10 | 30794 | 0.571673 | 0.428671 | 2.14237 |
| 10 | 13 | 42268 | 0.571625 | 0.428597 | 2.14254 |
| 50 | 15 | 56078 | 0.571585 | 0.428523 | 2.14270 |
| 100 | 16 | 64222 | 0.571585 | 0.428523 | 2.14270 |

Чем больше начальная величина коэффициента штрафа, тем больше итераций и вычислений целевой функции.

**Зависимость от стратегии изменения коэффициента штрафа**

– барьерная функция

– начальный коэффициент штрафа

– начальное приближение

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 5 | 21234 | 0.594636 | 0.449907 | 2.07978 |
|  | 6 | 25276 | 0.583474 | 0.440018 | 2.10945 |
|  | 10 | 30794 | 0.571673 | 0.428671 | 2.14237 |
|  | 11 | 62738 | 0.597051 | 0.451866 | 2.07366 |
|  | 101 | 378198 | 0.577087 | 0.434796 | 2.12592 |

Вычитание от коэффициента константы дает результат хуже, чем линейная зависимость, также чем меньше вычитаемая константа, тем больше количество итераций и количество вычислений целевой функции. Линейная зависимость дает лучший результат, чем больше константа, на которую делим, тем меньше итераций и количество вычислений целевой функции.

**Зависимость от начальной точки**

– барьерная функция

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– точность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 10 | 30794 | 0.571673 | 0.428671 | 2.14237 |
|  | 16 | 65142 | 0.571585 | 0.428523 | 2.14270 |
|  | 17 | 66256 | 0.571585 | 0.428523 | 2.14270 |

Чем ближе начальное приближение к искомому значению, тем меньше количество вычислений целевой функции и количество итераций.

**Зависимость от задаваемой точности**

– барьерная функция

– начальный коэффициент штрафа

– стратегия изменения коэффициента штрафа

– начальное приближение

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Количество итераций | Число вычислений целевой функции |  |  |  |
|  | 19 | 20668 | 0.571920 | 0.428474 | 2.14230 |
|  | 18 | 49882 | 0.571731 | 0.428743 | 2.14232 |
|  | 10 | 30794 | 0.571673 | 0.428671 | 2.14237 |
|  | 16 | 86686 | 0.571586 | 0.428525 | 2.14270 |

С увеличением задаваемой точности растет и точность полученного решения.

1. **Текст программы**

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <Windows.h>

#include <fstream>

#include <vector>

#include <algorithm>

#include <iostream>

#include <iomanip>

using namespace std;

double eps = 1e-7; // заданная точность

double epsQ = 1e-17; // заданная точность

double r = 1; // изначальный коэффициент штрафа

int maxiter = 50; // максимальное количество итераций

int f\_calc = 0; // число вычислений целевой функции

double sigma = 1e-8;

double a = 0, b = 0; // координаты отрезка

vector<double> x0(2); // начальное приближение

vector<double> xk(2); // значение на k-ой итерации

// целевая функция

double function(double x, double y)

{

f\_calc++;

return 5 \* pow(x - y, 2) + pow(x - 2, 2);

}

// функция ограничений (неравенство)

double g(double x, double y)

{

return x + y - 1;

}

// функция ограничений (равенство)

double h(double x, double y)

{

return x + y;

}

// штрафная(барьерная) функция

double G(double x, double y)

{

return pow(0.5 \* (g(x, y) + abs(g(x, y))), 1);

}

// штрафная функция

double H(double x, double y)

{

return pow(abs(h(x, y)), 2);

}

// вспомогательная задача

double Q(double x, double y)

{

return function(x, y) + r \* G(x, y) + r \* H(x, y);

}

// изменения коэффициент штрафа

void new\_r()

{

r \*= 2;

}

// поиск интервала с минимумом

void interval\_search(double x1, double& a, double& b, int num)

{

ofstream output4("output4.txt");

double x2 = x1 + sigma;

double h = sigma;

int k = 1;

if (num == 0)

{

if (Q(x1, x0[1]) < Q(x2, x0[1]))

{

x2 = x1 - sigma;

h \*= -1;

}

bool flag = true;

while (flag)

{

h \*= 2;

x1 = x2;

x2 += h;

k++;

if (Q(x1, x0[1]) > Q(x2, x0[1]))

{

x1 = x2;

}

else

{

flag = false;

x1 -= h / 2;

}

}

}

else

{

if (Q(x0[0], x1) < Q(x0[0], x2))

{

x2 = x1 - sigma;

h \*= -1;

}

bool flag = true;

while (flag)

{

h \*= 2;

x1 = x2;

x2 += h;

k++;

if (Q(x0[0], x1) > Q(x0[0], x2))

{

x1 = x2;

}

else

{

flag = false;

x1 -= h / 2;

}

}

}

if (x1 < x2)

{

a = x1;

b = x2;

}

else

{

a = x2;

b = x1;

}

}

// метод дихотомии

double dichotomy\_method(double& a, double& b, int num)

{

vector<double> x1(2), x2(2);

for (int i = 0; i < 2; i++)

{

x1[i] = x2[i] = x0[i];

}

for (int i = 1; abs(b - a) >= eps; i++)

{

x1[num] = (a + b - sigma) / 2;

x2[num] = (a + b + sigma) / 2;

if (Q(x1[0], x1[1]) > Q(x2[0], x2[1]))

a = x1[num];

else

b = x2[num];

}

return (a + b) / 2;

}

// метод Гаусса для поиска минимума

void Gauss()

{

bool end = false;

for (int iter = 1; iter <= maxiter && !end; iter++)

{

// вычисляем значения

for (int i = 0; i < 2; i++)

{

interval\_search(x0[i], a, b, i);

xk[i] = dichotomy\_method(a, b, i);

}

// проверка на конец

if (abs(xk[0] - x0[0]) < eps && abs(xk[1] - x0[1]) < eps)

end = true;

if (abs(Q(xk[0], xk[1]) - Q(x0[0], x0[1])) < eps)

end = true;

x0[0] = xk[0];

x0[1] = xk[1];

}

}

int main()

{

// начальное приближение

x0[0] = 0;

x0[1] = 0;

fstream output;

output.open("штраф.xls", ios::out);

bool end = false;

// цикл для штрафных функций

for (int iter = 1; iter <= maxiter && !end; iter++)

{

// решаем вспомогательную задачу Гауссом

Gauss();

// проверка на конец (если соблюдены ограничения)

if (abs(H(xk[0], xk[1]) + G(xk[0], xk[1])) < epsQ)

end = true;

else

// меняем коэффициент штрафа

new\_r();

output << iter << "\t" << f\_calc << "\t" << xk[0] << "\t" << xk[1] << "\t" << function(xk[0], xk[1]) << endl;

}

}