Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Лабораторная работа №3

Метод штрафных функций

по дисциплине

«Методы оптимизации»

Факультет прикладной математики и информатики

Группа ПМ-63

Студенты Майер В.А.

Преподаватели Чимитова Е. В.

Вариант 6

Новосибирск 2019

1. Цель работы

Ознакомится 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

*Функция*

*Минимум данной функции (6,6)*

*Ограничения*

*С учетом ограничение минимум функции (-1, 1)*

1. Результаты
2. *Метод штрафных функций*
3. *Неравенство*

* *Зависимость от стратегии изменения штрафа*

*EPS = 1e-5*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(-3: 3)* | *(7, 3)* | *(1000, -3000)* |
| *Прибавление 1*  *Начальная величина = 1* | *Iter: 28*  *Func iter: 22977*  *Res: (3,99999984 , 5,99999983)*  *Func: 27,9999996* | *Iter: 28*  *Func iter: 23397*  *Res: (3,99999982 , 5,99999981)*  *Func: 27,9999997* | *Iter: 28*  *Func iter: 29836*  *Res: (3,99999981 , 5,99999981)*  *Func: 27,99999996* |
| *Прибавление 1*  *Начальная величина = 0* | *Iter: 29*  *Func iter: 24039*  *Res: (3,99999980, 5,999999797)*  *Func: 27,99999987* | *Iter: 29*  *Func iter: 21927*  *Res: (3,999999819, 5,99999981)*  *Func: 27,9999999* | *Iter: 29*  *Func iter: 32386*  *Res: (3,999999826 , 5,999999816)*  *Func: 27,99999972* |
| *Прибавление 10*  *Начальная величина = 10* | *Iter: 3*  *Func iter: 14003*  *Res: (3,999999691, 5,99999969)*  *Func: 27,99999999* | *Iter: 3*  *Func iter: 14745*  *Res: (3,999999787, 5,999999787)*  *Func: 28,00000000* | *Iter: 3*  *Func iter: 20736*  *Res: (3,999999701, 5,999999701)*  *Func: 27,99999999* |
| *Прибавление 10*  *Начальная величина = 0* | *Iter: 4*  *Func iter: 15593*  *Res: (3,999999824, 5,999999824)*  *Func: 27,99999999* | *Iter: 4*  *Func iter: 15169*  *Res: (3,999999884, 5,999999884)*  *Func: 27,99999999* | *Iter: 4*  *Func iter: 21186*  *Res: (3,999999870, 5,999999870)*  *Func: 27,999999999* |
| *Прибавление 100*  *Начальная величина = 100* | *Iter: 1*  *Func iter: 209*  *Res: (0,9999999999, 2,999999999)*  *Func: 36,9999999* | *Iter: 1*  *Func iter: 211*  *Res: (0,9999999999, 2,999999999)*  *Func: 37,0000000001* | *Iter: 1*  *Func iter: 19929*  *Res: (-10,00000017, -8,000000163)*  *Func: 224,0000048* |
| *Прибавление 100*  *Начальная величина = 0* | *Iter: 2*  *Func iter: 14426*  *Res: (3,999999955, 5,99999995)*  *Func: 27,99999999* | *Iter: 2*  *Func iter: 14426*  *Res: (3,999999955, 5,999999955)*  *Func: 27,9999999* | *Iter: 2*  *Func iter: 20337*  *Res: (3,999999956, 5,999999956)*  *Func: 27,99999999* |
| *Прибавление 1000*  *Начальная величина = 1000* | *Iter: 1*  *Func iter: 209*  *Res: (0,9999999999, 3,00000000002)*  *Func: 37,0000000* | *Iter: 1*  *Func iter: 212*  *Res: (0,9999999999, 2,999999999)*  *Func: 37,00000000* | *Iter: 1*  *Func iter: 19308*  *Res:(-10,000000404, -8,000000404)*  *Func: 224,000011326* |
| *Прибавление 1000*  *Начальная величина = 0* | *Iter: 2*  *Func iter: 14428*  *Res: (3,999999955 , 5,999999955)*  *Func: 28,00000000* | *Iter: 2*  *Func iter: 14428*  *Res: (3,9999999553, 5,9999999553)*  *Func: 28,000000000* | *Iter: 2*  *Func iter: 20339*  *Res: (3,9999999569, 5,999999956)*  *Func: 28,000000000* |
| *Умножение в 2 раза*  *Начальная величина 1* | *Iter: 6*  *Func iter: 16335*  *Res: (3,999999814, 5,999999814)*  *Func: 27,999999999* | *Iter: 6*  *Func iter: 16017*  *Res: (3,999999813, 5,9999998138)*  *Func: 28,000000000* | *Iter: 6*  *Func iter: 21504*  *Res: (3,9999998509, 5,9999998509)*  *Func: 27,999999999* |
| *Умножение в 5 раз*  *Начальная величина 1* | *Iter: 4*  *Func iter: 15372*  *Res: (3,999999869, 5,999999869)*  *Func: 28,000000000* | *Iter: 4*  *Func iter: 14954*  *Res: (3,9999998825, 5,999999882)*  *Func: 27,999999999* | *Iter: 4*  *Func iter: 22115*  *Res: (3,999999833, 5,999999833)*  *Func: 27,999999999* |
| *Умножение в 10 раз*  *Начальная величина 1* | *Iter: 3*  *Func iter: 14956*  *Res: (3,999999881, 5,999999881)*  *Func: 27,999999999* | *Iter: 3*  *Func iter: 14956*  *Res: (3,9999998686, 5,999999868)*  *Func: 28,0000000000* | *Iter: 3*  *Func iter: 20655*  *Res: (3,9999998948, 5,9999998948)*  *Func: 28,000000000* |
| *Умножение в 100 раз*  *Начальная величина 1* | *Iter: 2*  *Func iter: 14638*  *Res: (3,999999910, 5,999999910)*  *Func: 27,999999999* | *Iter: 2*  *Func iter: 14532*  *Res: (3,9999999023, 5,9999999023)*  *Func: 27,999999999* | *Iter: 2*  *Func iter: 20443*  *Res: (3,9999999056, 5,9999999057)*  *Func: 28,000000000* |

|  |  |  |  |
| --- | --- | --- | --- |
| *Добавление знач. Ф. штрафа*  *Начальная величина 0* | *Iter: 166*  *Func iter: 93311*  *Res: (4,000009618, 5,999999831)*  *Func: 27,999725940* | *Iter: 166*  *Func iter: 91947*  *Res: (4,0000095937, 5,9999998164)*  *Func: 27,999726235* | *Iter: 166*  *Func iter: 96712*  *Res: (4,0000096311, 5,9999998478)*  *Func: 27,999726068* |
| *Добавление знач. Ф. штрафа \* 2*  *Начальная величина 0* | *Iter: 81*  *Func iter: 59759*  *Res: (4,000008682, 5,999999860)*  *Func: 27,999752989* | *Iter: 81*  *Func iter: 61841*  *Res: (4,00000862253, 5,99999979789)*  *Func: 27,999752910* | *Iter: 81*  *Func iter: 57758*  *Res: (4,0000086589, 5,9999998385)*  *Func: 27,999753028* |
| *Добавление знач. Ф. штрафа \* 5*  *Начальная величина 0* | *Iter: 29*  *Func iter: 28613*  *Res: (4,000008322, 5,999999848)*  *Func: 27,999762730* | *Iter: 29*  *Func iter: 25583*  *Res: (4,0000083338, 5,9999998602)*  *Func: 27,999762741* | *Iter: 29*  *Func iter: 31186*  *Res: (4,0000082981, 5,99999981647)*  *Func: 27,999762514* |
| *Добавление знач. Ф. штр. \* 10*  *Начальная величина 0* | *Iter: 11*  *Func iter: 18700*  *Res: (4,0000071, 5,9999998)*  *Func: 27,999797089* | *Iter: 11*  *Func iter: 19632*  *Res: (4,00000715181, 5,9999998908)*  *Func: 27,9997966931* | *Iter: 11*  *Func iter: 25139*  *Res: (4,0000070976, 5,99999984)*  *Func: 27,999797061* |
| *Добавление знач. Ф. штр. \* 100*  *Начальная величина 0* | *Iter: 2*  *Func iter: 14426*  *Res: (3,999999955, 5,999999955)*  *Func: 27,999999999* | *Iter: 2*  *Func iter: 14426*  *Res: (3,9999999554, 5,9999999554)*  *Func: 28,000000000* | *Iter: 2*  *Func iter: 20337*  *Res: (3,9999999569, 5,9999999569)*  *Func: 27,999999999* |
| *Добавление знач. Ф. штр. \* 1000*  *Начальная величина 0* | *Iter: 2*  *Func iter: 14428*  *Res: (3,999999955, 5,999999955)*  *Func: 28,000000000* | *Iter: 2*  *Func iter: 14428*  *Res: (3,9999999553, 5,9999999553)*  *Func: 28,000000000* | *Iter: 2*  *Func iter: 20339*  *Res: (3,9999999569, 5,9999999569)*  *Func: 28,0000000009* |

* *Зависимость от представление функции*

*EPS = 1e-5*

*Начальное значение коэффициента штрафа 10*

*Коэффициент штрафа умножается на 10*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(-3: 3)* | *(7, 3)* | *(1000, -3000)* |
| *Max(x+2-y, 0)* | *Iter: 2*  *Func iter: 13684*  *Res: (3,999999671, 5,999999671)*  *Func: 28,000000000* | *Iter: 2*  *Func iter: 14002*  *Res: (3,9999996942, 5,9999996942)*  *Func: 27,999999999* | *Iter: 2*  *Func iter: 20523*  *Res: (3,9999997078, 5,9999997078)*  *Func: 28,000000000* |
| *Max(x+2-y, 0) ^ 2* | *Iter: 4*  *Func iter: 31523*  *Res: (4,001398530, 5,999999509)*  *Func: 27,960841113* | *Iter: 4*  *Func iter: 31525*  *Res: (4,0013985415, 5,9999995205)*  *Func: 27,960841114* | *Iter: 4*  *Func iter: 39338*  *Res: (4,0013985089, 5,9999994885)*  *Func: 27,960841129* |
| *Max(x+2-y, 0) ^ 100* | *Iter: 5*  *Func iter: 1018993*  *Res: (4,873751726, 5,999993040)*  *Func: 8,8789364716* | *Iter: 5*  *Func iter: 1026588*  *Res: (4,8737522879, 5,99999359)*  *Func: 8,8789364481* | *\*\** |

1. *Уравнение*

*EPS = 1e-5*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(-3: 3)* | *(7, 3)* | *(1000, -3000)* |
| *Прибавление 1*  *Начальная величина = 1* | *Iter: 6*  *Func iter: 6987*  *Res:(-0,2068965408, 0,2068965408)*  *Func: 34,758620689* | *Iter: 6*  *Func iter: 10269*  *Res:(-0,20689653460, 0,20689653459)*  *Func: 34,75862068* | *Iter: 6*  *Func iter: 8845*  *Res:(-0,2068965465, 0,20689654663)*  *Func: 34,758620689* |
| *Прибавление 1*  *Начальная величина = 0* | *Iter: 7*  *Func iter: 6881*  *Res:(-0,2068965401, 0,2068965401)*  *Func: 34,758620689* | *Iter: 7*  *Func iter: 11751*  *Res:(-0,20689653794, 0,20689653802)*  *Func: 34,758620689* | *Iter: 7*  *Func iter: 27180*  *Res:(-0,20689654115, 0,2068965412)*  *Func: 34,758620689* |
| *Прибавление 10*  *Начальная величина = 10* | *Iter: 1*  *Func iter: 313*  *Res:(-0,2068965387, 0,2068965387)*  *Func: 34,758620689* | *Iter: 1*  *Func iter: 213*  *Res:(-0,20689655044, 0,20689655046)*  *Func: 34,758620689* | *Iter: 1*  *Func iter: 481*  *Res:(-0,20689654405, 0,20689654407)*  *Func: 34,758620689* |
| *Прибавление 10*  *Начальная величина = 0* | *Iter: 2*  *Func iter: 1059*  *Res:(-0,2068965412, 0,2068965413)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1161*  *Res:(-0,20689654140, 0,2068965414)*  *Func: 34,7586206893* | *Iter: 2*  *Func iter: 1330*  *Res:(-0,20689653829, 0,20689653828)*  *Func: 34,758620689* |
| *Прибавление 100*  *Начальная величина = 100* | *Iter: 1*  *Func iter: 315*  *Res:(-0,2068965454, 0,2068965454)*  *Func: 34,758620689* | *Iter: 1*  *Func iter: 421*  *Res:(-0,2068965499, 0,20689654997)*  *Func: 34,758620689* | *Iter: 1*  *Func iter: 576*  *Res:(-0,20689654150, 0,20689654153)*  *Func: 34,758620689* |
| *Прибавление 100*  *Начальная величина = 0* | *Iter: 2*  *Func iter: 1059*  *Res:(-0,2068965310, 0,2068965311)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1057*  *Res:(-0,20689654128, 0,20689654126)*  *Func: 34,7586206897* | *Iter: 2*  *Func iter: 1330*  *Res:(-0,20689653829, 0,20689653828)*  *Func: 34,7586206897* |

|  |  |  |  |
| --- | --- | --- | --- |
| *Умножение в 2 раза*  *Начальная величина 1* | *Iter: 4*  *Func iter: 4020*  *Res:(-0,2068965400, 0,20689654003)*  *Func: 34,758620689* | *Iter: 4*  *Func iter: 5398*  *Res:(-0,20689654215, 0,20689654216)*  *Func: 34,758620689* | *Iter: 4*  *Func iter: 3022*  *Res:(-0,20689654128, 0,20689654132)*  *Func: 34,758620689* |
| *Умножение в 5 раз*  *Начальная величина 1* | *Iter: 3*  *Func iter: 2110*  *Res:(-0,2068965381, 0,20689653819)*  *Func: 34,758620689* | *Iter: 3*  *Func iter: 6354*  *Res:(-0,20689654159, 0,20689654160)*  *Func: 34,758620689* | *Iter: 3*  *Func iter: 4282*  *Res:(-0,20689654256, 0,20689654255)*  *Func: 34,7586206896* |
| *Умножение в 10 раз*  *Начальная величина 1* | *Iter: 2*  *Func iter: 1058*  *Res:(-0,2068965444, 0,2068965444)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1056*  *Res:(-0,20689654123, 0,20689654122)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1330*  *Res:(-0,20689653701, 0,20689653699)*  *Func: 34,758620689* |
| *Умножение в 100 раз*  *Начальная величина 1* | *Iter: 2*  *Func iter: 1058*  *Res:(-0,2068965444, 0,2068965444)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1368*  *Res:(-0,20689654258, 0,206896542595262)*  *Func: 34,7586206896* | *Iter: 2*  *Func iter: 1330*  *Res:(-0,20689653701, 0,20689653699)*  *Func: 34,75862068* |
| *Добавление знач. Ф. штрафа*  *Начальная величина 0* | *Iter: 2*  *Func iter: 1475*  *Res: (-0,206896550345259 , 0,206896550382369)*  *Func: 34,7586206894402* | *Iter: 2*  *Func iter: 1057*  *Res: (-0,206896541285402 , 0,206896541263901)*  *Func: 34,7586206897797* | *Iter: 2*  *Func iter: 1330*  *Res: (-0,206896538293941 , 0,206896538280585)*  *Func: 34,7586206897326* |
| *Добавление знач. Ф. штрафа \* 2*  *Начальная величина 0* | *Iter: 2*  *Func iter: 1371*  *Res: (-0,206896534660522 , 0,206896534616991)*  *Func: 34,7586206899074* | *Iter: 2*  *Func iter: 1057*  *Res: (-0,206896541285402 , 0,206896541263901)*  *Func: 34,7586206897797* | *Iter: 2*  *Func iter: 1330*  *Res: (-0,206896538293941 , 0,206896538280585)*  *Func: 34,7586206897326* |
| *Добавление знач. Ф. штрафа \* 5*  *Начальная величина 0* | *Iter: 2*  *Func iter: 1371*  *Res:(-0,2068965346, 0,2068965346)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1057*  *Res:(-0,20689654128, 0,20689654126)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1330*  *Res:(-0,20689653829, 0,20689653828)*  *Func: 34,758620689* |
| *Добавление знач. Ф. штр. \* 10*  *Начальная величина 0* | *Iter: 2*  *Func iter: 1167*  *Res:(-0,2068965469, 0,2068965469)*  *Func: 34,758620689* | *Iter: 2*  *Func iter: 1271*  *Res:(-0,20689655267, 0,20689655263)*  *Func: 34,7586206898* | *Iter: 2*  *Func iter: 1332*  *Res:(-0,20689653859, 0,20689653857)*  *Func: 34,758620689* |

* *Зависимость от представление функции*

*EPS = 1e-5*

*Начальное значение коэффициента штрафа 10*

*Коэффициент штрафа умножается на 10*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(-3: 3)* | *(7, 3)* | *(1000, -3000)* |
| *|x+y|* | *Iter: 1*  *Func iter: 313*  *Res:(-0,2068965387, 0,2068965387)*  *Func: 34,758620689* | *Iter: 1*  *Func iter: 213*  *Res:(-0,20689655044, 0,20689655046)*  *Func: 34,7586206895* | *Iter: 1*  *Func iter: 481*  *Res:(-0,20689654405, 0,20689654407)*  *Func: 34,758620689* |
| *(x+y)^2* | *Iter: 3*  *Func iter: 1481*  *Res:(-0,2053986853, 0,2082945356)*  *Func: 34,741846753* | *Iter: 3*  *Func iter: 1275*  *Res:(-0,20539868765, 0,20829453933)*  *Func: 34,7418467454* | *Iter: 3*  *Func iter: 1517*  *Res:(-0,20539868361, 0,20829453520)*  *Func: 34,74184674* |
| *(x+y)^100* | *Iter: 4*  *Func iter: 1377*  *Res: (0,2506733489, 0,6339617744)*  *Func: 29,822736357* | *Iter: 4*  *Func iter: 1381*  *Res:(0,25067335009, 0,63396177310)*  *Func: 29,8227363589* | *Iter: 4*  *Func iter: 23915*  *Res:(0,2506733464, 0,63396177600)*  *Func: 29,822736362* |

**Вывод:**

Количество итераций не зависит от начального положение, это связно с тем, что на первой же итерации все начальные точки сведутся в одну после первой же итерации.

При большом значении начального значения коэффициента штрафа, значение основной функции теряется.

При увеличении степени функции штрафа теряется точность решения

1. *Метод барьерных функций*
2. *Неравенство*

* *Зависимость от стратегии изменения штрафа*

*EPS = 1e-5*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(-3: 3)* | *(7, 3)* | *(1000, -3000)* |
| *Деление на 2*  *Начальная величина = 0,5* | *Iter: 26*  *Func iter: 349857*  *Res: (3,999973221, 5,999996290)*  *Func: 28,000645935* | *Iter: 26*  *Func iter: 349652*  *Res: (3,999973191, 5,999996260)*  *Func: 28,000645935* | *Iter: 26*  *Func iter: 378711*  *Res: (3,999972865, 5,999995934)*  *Func: 28,000645933* |
| *Деление на 2*  *Начальная величина = 1* | *Iter: 27*  *Func iter: 260521*  *Res: (3,9999734617, 5,9999965307)*  *Func: 28,000645934* | *Iter: 27*  *Func iter: 261668*  *Res:(3,9999735286, 5,9999965977)*  *Func: 28,000645936* | *Iter: 27*  *Func iter: 290297*  *Res:(3,9999738698, 5,9999969389)*  *Func: 28,000645936* |
| *Деление на 2*  *Начальная величина = 0.01* | *Iter: 21*  *Func iter: 1907673*  *Res: (3,9999656065, 5,9999840616)*  *Func: 28,00051674* | *Iter: 21*  *Func iter: 1915684*  *Res: (3,9999662739, 5,9999847292)*  *Func: 28,000516750* | *Iter: 21*  *Func iter: 2070702*  *Res: (3,9999665665, 5,9999850217)*  *Func: 28,000516748* |
| *Деление на 5*  *Начальная величина = 0,2* | *Iter: 13*  *Func iter: 511161*  *Res: (3,9999894973, 5,9999949063)*  *Func: 28,000151451* | *Iter: 13*  *Func iter: 510124*  *Res:(3,9999893710, 5,9999947800)*  *Func: 28,000151451* | *Iter: 13*  *Func iter: 548965*  *Res:(3,9999889797, 5,9999943887)*  *Func: 28,000151451* |
| *Деление на 5*  *Начальная величина = 1* | *Iter: 14*  *Func iter: 257505*  *Res:(3,9999910554, 5,9999964643)*  *Func: 28,000151450* | *Iter: 14*  *Func iter: 258236*  *Res: (3,9999910874, 5,9999964964)*  *Func: 28,000151452* | *Iter: 14*  *Func iter: 287385*  *Res: (3,9999914911, 5,99999690)*  *Func: 28,000151451* |
| *Деление на 5*  *Начальная величина = 0.01* | *Iter: 11*  *Func iter: 1905593*  *Res: (3,9999779884, 5,999984035)*  *Func: 28,000169327* | *Iter: 11*  *Func iter: 1913604*  *Res: (3,9999786532, 5,9999847006)*  *Func: 28,000169328* | *Iter: 11*  *Func iter: 2068622*  *Res: (3,9999789532, 5,9999850006)*  *Func: 28,000169327* |
| *Деление на 10*  *Начальная величина = 0.1* | *Iter: 10*  *Func iter: 679537*  *Res:(3,9999917248, 5,9999936146)*  *Func: 28,000052914* | *Iter: 10*  *Func iter: 679540*  *Res: (3,9999916757, 5,9999935656)*  *Func: 28,000052915* | *Iter: 10*  *Func iter: 735335*  *Res: (3,9999916295, 5,9999935193)*  *Func: 28,000052915* |

|  |  |  |  |
| --- | --- | --- | --- |
| *Деление на 10*  *Начальная величина = 1* | *Iter: 11*  *Func iter: 256881*  *Res: (3,9999945711, 5,9999964609)*  *Func: 28,000052914* | *Iter: 11*  *Func iter: 257612*  *Res: (3,9999946012, 5,9999964911)*  *Func: 28,000052915* | *Iter: 11*  *Func iter: 286761*  *Res: (3,9999950018, 5,9999968916)*  *Func: 28,000052914* |
| *Прибавление 10*  *Начальная величина = 0.01* | *Iter: 9*  *Func iter: 1905177*  *Res: (3,9999821410, 5,9999840308)*  *Func: 28,00005291* | *Iter: 9*  *Func iter: 1913188*  *Res:(3,9999828099, 5,9999846997)*  *Func: 28,000052915* | *Iter: 9*  *Func iter: 2068206*  *Res:(3,9999831051, 5,9999849949)*  *Func: 28,000052915* |
| *Деление на 1000*  *Начальная величина = 1* | *Iter: 5*  *Func iter: 255633*  *Res: (3,9999962595, 5,9999964485)*  *Func: 28,000005292* | *Iter: 5*  *Func iter: 256364*  *Res:(3,9999962951, 5,9999964841)*  *Func: 28,000005292* | *Iter: 5*  *Func iter: 285513*  *Res: (3,9999966993, 5,999996888)*  *Func: 28,000005291* |
| *Деление на 1000*  *Начальная величина = 0.001* | *Iter: 4*  *Func iter: 5458857*  *Res: (3,9999589169, 5,9999591059)*  *Func: 28,000005293* | *Func iter: 5431404*  *Res: (3,99995687483, 5,9999570638)*  *Func: 28,000005293* | *Iter: 4*  *Func iter: 5903206*  *Res: (3,9999584647, 5,9999586537)*  *Func: 28,00000529* |
| *Деление на 1000*  *Начальная величина = 0.000001* | *Iter: 3*  *Func iter: 119527201*  *Res: (3,9988887758, 5,9988889648)*  *Func: 28,000006525* | *Iter: 3*  *Func iter: 119617580*  *Res: (3,9988951909, 5,998895379)*  *Func: 28,000006511* | *Iter: 3*  *Func iter: 133392958*  *Res: (3,9988919415, 5,9988921305)*  *Func: 28,000006518* |

* *Зависимость от представление функции*

*EPS = 1e-5*

*Начальное значение коэффициента штрафа 1*

*Коэффициент штрафа умножается на 100*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(-3: 3)* | *(7, 3)* | *(1000, -3000)* |
| *1/ (y – x - 2)* | *Iter: 7*  *Func iter: 256049*  *Res: (3,9999962611, 5,9999964501)*  *Func: 28,000005291* | *Iter: 7*  *Func iter: 256780*  *Res: (3,9999962962, 5,9999964852)*  *Func: 28,000005290* | *Iter: 7*  *Func iter: 285929*  *Res:(3,9999966995 , 5,9999968884)*  *Func: 28,000005291* |
| *-ln(y – x - 2)* | *Iter: 4*  *Func iter: 574289*  *Res: (3,9999940811, 5,9999941168)*  *Func: 28,000000999* | *Iter: 4*  *Func iter: 577204*  *Res:(3,9999942953, 5,9999943310)*  *Func: 28,000000999* | *Iter: 4*  *Func iter: 614276*  *Res: (3,9999938871, 5,999993922)*  *Func: 28,000000999* |

**Вывод:**

Количество итераций не зависит от начального положение, это связно с тем, что на первой же итерации все начальные точки сведутся в одну после первой же итерации.

Количество итераций алгоритма уменьшается при уменьшении начального значения и при увеличении делителя коэффициента штрафа на каждой итерации.

При уменьшении начального значения, количество итераций вычисления функции увеличивается.

1. **Текст программы**

using System;

using System.Collections.Generic;

namespace csharp\_lab1

{

class Program

{

static void Main(string[] args)

{

#region 3 lab

Solver.EPS = 1e-5;

List<Func<Solver.Point, double>> func = new List<Func<Solver.Point, double>>();

//Штрафные функции

//a)

//func.Add(p1 => Math.Max(-p1.y + p1.x + 2, 0));

//func.Add(p1 => Math.Pow(1.0 / 2.0 \* (-p1.y + p1.x + 2 + Math.Abs(-p1.y + p1.x + 2)), 2));

//func.Add(p1 => Math.Pow(1.0 / 2.0 \* (-p1.y + p1.x + 2 + Math.Abs(-p1.y + p1.x + 2)), 100));

//б)

//func.Add(p1 => Math.Abs(p1.x + p1.y));

//func.Add(p1 => Math.Pow(p1.x + p1.y, 2));

//func.Add(p1 => Math.Pow(p1.x + p1.y, 100));

//Барьерные функции

//func.Add(p1 =>

//{

// if (p1.y - p1.x - 2 > 0) return 1.0 / (p1.y - p1.x - 2);

// else return 1e+100 + 1e+98 \* (-p1.y + p1.x + 2);

//});

func.Add(p1 =>

{

if (p1.y - p1.x - 2 > 0) return -Math.Log(p1.y - p1.x - 2);

else return 1e+100 + 1e+98 \* (-p1.y + p1.x + 2);

});

//Solver.Point p = Solver.Solve\_Penalty\_function(p2 => 7 \* Math.Pow(p2.x - p2.y, 2) + Math.Pow(p2.y - 6, 2), func, new Solver.Point(-3, 3));

//Solver.Point p = Solver.Solve\_Penalty\_function(p2 => 7 \* Math.Pow(p2.x - p2.y, 2) + Math.Pow(p2.y - 6, 2), func, new Solver.Point(7, 3));

//Solver.Point p = Solver.Solve\_Penalty\_function(p2 => 7 \* Math.Pow(p2.x - p2.y, 2) + Math.Pow(p2.y - 6, 2), func, new Solver.Point(1000, -3000));

Solver.Point p = Solver.Solve\_Barrier\_function(p2 => 7 \* Math.Pow(p2.x - p2.y, 2) + Math.Pow(p2.y - 6, 2), func, new Solver.Point(-3, 3));

//Solver.Point p = Solver.Solve\_Barrier\_function(p2 => 7 \* Math.Pow(p2.x - p2.y, 2) + Math.Pow(p2.y - 6, 2), func, new Solver.Point(7, 3));

//Solver.Point p = Solver.Solve\_Barrier\_function(p2 => 7 \* Math.Pow(p2.x - p2.y, 2) + Math.Pow(p2.y - 6, 2), func, new Solver.Point(1000, -3000));

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

#endregion

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);

}

}

#region lab3

static public Point Solve\_Penalty\_function(Func<Point, double> func, List<Func<Point, double>> h, Point p0 = new Point())

{

Point x = p0;

FuncIters = 0;

LastIters = 0;

int ped = 22;

//Коэфициент функции штрафа

List<double> r = new List<double>();

//Добавка для коэффициента функции штрафа на каждом шаге

List<double> value = new List<double>();

//Коэфициент добавки

double w = 10;

for (int i = 0; i < h.Count; i++)

{

r.Add(1);

value.Add(0);

}

Func<Point, double> Q = p =>

{

double Penatly = 0;

for (int i = 0; i < h.Count; i++)

Penatly += r[i] \* h[i](p);

return func(p) + Penatly;

};

Console.WriteLine("i".PadLeft(4) + "".PadLeft(ped) + "(x,y)".PadRight(ped) + "f(x, y)".PadLeft(ped) + "r".PadLeft(ped));

int end;

var xlast = x;

do

{

end = 0;

for (int i = 0; i < h.Count; i++)

r[i] \*= 10;// w \* value[i];

x = Gause(Q, x);

for (int i = 0; i < h.Count; i++)

value[i] = h[i](x);

LastIters++;

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

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

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

Q(x).ToString().PadLeft(ped));

for (int i = 0; i < h.Count; i++)

Console.Write(value[i].ToString().PadLeft(ped));

Console.Write("\n");

for (int i = 0; i < h.Count; i++)

if (value[i] <= EPS)

end++;

} while (end < h.Count && (xlast - x).Norm() > EPS);

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 Solve\_Barrier\_function(Func<Point, double> func, List<Func<Point, double>> h, Point p0 = new Point())

{

Point x = p0;

FuncIters = 0;

LastIters = 0;

int ped = 22;

//Коэфициент функции штрафа

List<double> r = new List<double>();

//Добавка для коэффициента функции штрафа на каждом шаге

List<double> value = new List<double>();

//Коэфициент добавки

double w = 100;

for (int i = 0; i < h.Count; i++)

{

r.Add(100);

value.Add(0);

}

Func<Point, double> Q = p =>

{

double Penatly = 0;

for (int i = 0; i < h.Count; i++)

Penatly += r[i] \* h[i](p);

return func(p) + Penatly;

};

Console.WriteLine("i".PadLeft(4) + "".PadLeft(ped) + "(x,y)".PadRight(ped) + "f(x, y)".PadLeft(ped) + "r".PadLeft(ped));

int end;

var xlast = x;

do

{

end = 0;

for (int i = 0; i < h.Count; i++)

r[i] /= w; //\* value[i];

xlast = x;

x = Gause(Q, x);

for (int i = 0; i < h.Count; i++)

value[i] = h[i](x);

LastIters++;

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

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

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

Q(x).ToString().PadLeft(ped));

for (int i = 0; i < h.Count; i++)

Console.Write(value[i].ToString().PadLeft(ped));

Console.Write("\n");

for (int i = 0; i < h.Count; i++)

if (Math.Abs(value[i]) <= EPS)

end++;

} while ( (xlast - x).Norm() > EPS /\*end < h.Count\*/);

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 Gause(Func<Point, double> func, Point p0 = new Point())

{

Point x = p0;

Point xlast;

Point ex = new Point(1, 0);// 1 0

Point ey = new Point(0, 1);// 0 1

Point p;

double lambda;

Func<double, double> f;

do

{

lambda = 1;

xlast = x;

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

p = Find\_Line(lambda, f);

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

x += lambda \* ex;

lambda = 1;

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

p = Find\_Line(lambda, f);

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

x += lambda \* ey;

FuncIters+=2;

} while ((xlast - x).Norm() > 1e-8) ;

return x;

}

#endregion