|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| 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** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-72 |
| Вариант: | 1 |
| Студент: | Сычев Егор |
|  |  |
|  |  |
| Преподаватель: | Постовалов Сергей Николаевич |
|  |  |
|  | | |
| Новосибирск | | |
| 2020 | | |

1. **Цель работы**

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. **Задания**

|  |  |
| --- | --- |
| № | Вид работы |
|  | 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. |
| 3\* | Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелиней­ного программирования с ограничением типа неравенства (**только пункт а**) с использованием **метода барьерных функций**. |
| 4\* | Исследовать сходимость **метода барьерных функций** (**только пункт а**) в зависимости от   * выбора барьерных функций, * начальной величины коэффициента штрафа, * стратегии изменения коэффициента штрафа, * начального приближения, * задаваемой точности .   Сформулировать выводы. |



при ограничении:

а) 

б) 

1. **Ход работы**
   1. **Метод штрафных функций**
      1. **Ограничение неравенством**
         1. **Стандартные параметры**

G(g(x)) = 0.5 \* (g(x) – |g(x)|)

r0 = 1

ri = R(i) = 2i, i – номер итерации

x0 = (0, 0)

ε = 0.001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 269 | 2 | 0,567 | 0,434 | 2,143 |

* + - 1. **Выбор штрафной функции**

G(g(x)) = (0.5 \* (g(x) - |g(x)|)2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | X | y | f(x, y) |
| 216 | 11 | 0,564 | 0,437 | 2,143 |

G(g(x)) = (0.5 \* (g(x) - |g(x)|)100

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | X | y | f(x, y) |
| 430 | 64 | 0,859 | 0,757 | 1,355 |

* + - 1. **Выбор начальной величины коэффициента штрафа**

r0 = 2-32

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 269 | 2 | 0,567 | 0,434 | 2,143 |

r0 = 232

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 269 | 2 | 0,567 | 0,434 | 2,143 |

* + - 1. **Выбор стратегии изменения коэффициента штрафа**

R(i) = i \* (i + 1) / 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| function calls | iterations | x | y | | f(x, y) |
| 246 | 3 | 0,523 | 0,477 | 2,191 | | |

R(i) = i \* (i + 1) / 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 246 | 3 | 0,619 | 0,38 | 2,192 |

* + - 1. **Выбор начального приближения**

x0 = (0.08, -0.08)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 289 | 2 | 0,57 | 0,431 | 2,142 |

x0 = (100, -100)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 937 | 2 | 0,568 | 0,432 | 2,142 |

* + - 1. **Выбор точности**

ε = 1e-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 1011 | 2 | 0,56344 | 0,43656 | 2,1442 |

ε = 1e-7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 2232 | 2 | 0,563426 | 0,436574 | 2,144202 |

* + 1. **Ограничение равенством**
       1. **Стандартные параметры**
       2. **Выбор штрафной функции**
       3. **Выбор начальной величины коэффициента штрафа**
       4. **Выбор стратегии изменения коэффициента штрафа**
       5. **Выбор начального приближения**
       6. **Выбор точности**
  1. **Метод барьерных функций**
     1. **Ограничение неравенством**
        1. **Стандартные параметры**

G(g(x)) = 0.5 \* (g(x) – |g(x)|)

r0 = 1

ri = R(i) = 2i, i – номер итерации

x0 = (0, 0)

ε = 0.001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 22 | 1 | 0,215 | 0,036 | 3,347 |

* + - 1. **Выбор штрафной функции**

G(g(x)) = (0.5 \* (g(x) - |g(x)|)2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | X | y | f(x, y) |
| 18 | 1 | 0,041 | -0,154 | 4,028 |

G(g(x)) = (0.5 \* (g(x) - |g(x)|)100

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | X | y | f(x, y) |
| 72 | 1 | 0,068 | -0,108 | 3,888 |

* + - 1. **Выбор начальной величины коэффициента штрафа**

r0 = 2-32

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 22 | 1 | 0,215 | 0,036 | 3,347 |

r0 = 232

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 22 | 1 | 0,215 | 0,036 | 3,347 |

* + - 1. **Выбор стратегии изменения коэффициента штрафа**

R(i) = i \* (i + 1) / 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| function calls | iterations | x | y | | f(x, y) |
| 10207 | 10000 | 0,617 | 0,361 | 2,239 | | |

R(i) = i \* (i + 1) / 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 53 | 3 | 0,638 | 0,362 | 2,235 |

* + - 1. **Выбор начального приближения**

x0 = (0.08, -0.08)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 34 | 1 | 0,215 | 0,036 | 3,347 |

x0 = (100, -100)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 698 | 1 | 0,214 | 0,036 | 3,348 |

* + - 1. **Выбор точности**

ε = 1e-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 87 | 1 | 0,21501 | 0,03651 | 3,34552 |

ε = 1e-7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function calls | iterations | x | y | f(x, y) |
| 184 | 1 | 0,215008 | 0,036509 | 3,345506 |

1. **Текст программы**

**descent\_methods.rs**

use nalgebra::allocator::Allocator;

use nalgebra::DefaultAllocator;

use nalgebra::DimName;

use nalgebra::VectorN;

use super::one\_dimension\_searchers::minimize;

pub fn conjugate\_gradients<D>(

    f: &dyn Fn(&VectorN<f64, D>) -> f64,

    df: &dyn Fn(&VectorN<f64, D>) -> VectorN<f64, D>,

    mut x: VectorN<f64, D>,

    eps: f64,

    rev: bool,

) -> (VectorN<f64, D>, i32, i32, String)

where

    D: DimName,

    DefaultAllocator: Allocator<f64, D>,

{

    let mut iter = 0;

    let mut func\_calls = 1;

    let mut result = String::new();

    let precision = -eps.log10().round() as usize;

    result.push\_str(&format!("\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\n",

"i", "xi", "yi", "f(x, y)", "si1", "si2", "lambdai", "|yi-y(i-1)|", "|yi-y(i-1)|", "|fi-f(i-1)|",

"angle((xi, yi), si)", "gi1", "gi2"));

    let mut g = df(&x);

    #[allow(non\_snake\_case)]

    let mut S = g.clone();

    result.push\_str(&format!("\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\n",

    iter,

    x[0],

    x[1],

    if rev { 1./ f(&x) } else {f(&x)},

    S[0],

    S[1],

    0,

    0,

    0,

    0,

    x.angle(&S),

    g[0],

    g[1]));

    loop {

        let (lambda, search\_func\_calls) =

            minimize(&|lambda: f64| -> f64 { f(&(&x + lambda \* &S)) }, 0., eps);

        let dx = &(lambda \* &S);

        x += dx;

        iter += 1;

        func\_calls += search\_func\_calls;

        let \_g = g;

        g = df(&x);

        func\_calls += 1;

        if iter % D::dim() as i32 == 0 {

            S = g.clone();

        } else {

            S = g.clone() + (g.norm() / \_g.norm()).powi(2) \* S;

        }

        result.push\_str(&format!("\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\n",

        iter,

        x[0],

        x[1],

        if rev { 1./ f(&x) } else {f(&x)},

        S[0],

        S[1],

        lambda,

        dx[0].abs(),

        dx[1].abs(),

        (f(&x) - f(&(&x - dx))).abs(),

        x.angle(&S),

        g[0],

        g[1]));

        if S.norm() < eps {

            return (x, func\_calls, iter, result);

        }

    }

}

**variable\_metric\_methods.rs**

use nalgebra::allocator::Allocator;

use nalgebra::DefaultAllocator;

use nalgebra::DimName;

use nalgebra::MatrixN;

use nalgebra::VectorN;

use super::one\_dimension\_searchers::minimize;

pub fn broyden<D>(

    f: &dyn Fn(&VectorN<f64, D>) -> f64,

    df: &dyn Fn(&VectorN<f64, D>) -> VectorN<f64, D>,

    mut x: VectorN<f64, D>,

    eps: f64,

    rev: bool,

) -> (VectorN<f64, D>, i32, i32, String)

where

    D: DimName,

    DefaultAllocator: Allocator<f64, D> + Allocator<f64, D, D> + Allocator<f64, nalgebra::U1, D>,

{

    let mut iter = 0;

    let mut func\_calls = 1;

    let mut result = String::new();

    let precision = -eps.log10().round() as usize;

    result.push\_str(&format!("\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\n",

        "i", "xi", "yi", "f(x, y)", "si1", "si2", "lambdai", "|yi-y(i-1)|", "|yi-y(i-1)|", "|fi-f(i-1)|",

"angle((xi, yi), si)", "gi1", "gi2", "etai11", "etai12", "etai21", "etai22"));

    let mut g = df(&x);

    let mut eta = MatrixN::<f64, D>::from\_diagonal\_element(1.);

    result.push\_str(&format!("\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\n",

    iter,

    x[0],

    x[1],

    if rev { 1./ f(&x) } else {f(&x)},

    (&eta \* &g)[0],

    (&eta \* &g)[1],

    0,

    0,

    0,

    0,

    x.angle(&(&eta \* &g)),

    g[0],

    g[1],

    eta[(0, 0)],

    eta[(0, 1)],

    eta[(1, 0)],

    eta[(1, 1)]));

    loop {

        #[allow(non\_snake\_case)]

        let S = &eta \* &g;

        let (lambda, search\_func\_calls) =

            minimize(&|lambda: f64| -> f64 { f(&(&x + lambda \* &S)) }, 0., eps);

        let dx = &(lambda \* &S);

        x += dx;

        iter += 1;

        func\_calls += search\_func\_calls;

        let dg = &(df(&x) - &g);

        g += dg;

        func\_calls += 1;

        if iter % D::dim() as i32 == 0 {

            eta = MatrixN::<f64, D>::from\_diagonal\_element(1.);

        } else {

            let dx\_eta\_dg = &(dx - &eta \* dg);

            eta += dx\_eta\_dg \* dx\_eta\_dg.transpose() / dx\_eta\_dg.dot(dg);

        }

        result.push\_str(&format!("\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\"{}\";\n",

        iter,

        x[0],

        x[1],

        if rev { 1./ f(&x) } else {f(&x)},

        S[0],

        S[1],

        lambda,

        dx[0].abs(),

        dx[1].abs(),

        (f(&x) - f(&(&x - dx))).abs(),

        x.angle(&S),

        g[0],

        g[1],

        eta[(0, 0)],

        eta[(0, 1)],

        eta[(1, 0)],

        eta[(1, 1)]));

        if g.norm() < eps {

            return (x, func\_calls, iter, result);

        }

    }

}

1. **Выводы**

В случае квадратичной функции разница между методами сопряжённых градиентов и переменной метрики не сильно различна, а выбор начального приближения слабо влияет на сходимость. Сходимость быстрая, но из-за решения задачи минимизации, количество вызовов функции достаточно велико. На сходимость может сильно повлиять точность: например, при использовании сопряжённых градиентов в модификации Флетчера-Ривса с точностью   
10-5 было больше итераций, чем при других точностях. Причиной этого может быть неоптимальное решение задачи одномерной минимизации при заданной точности.

В случае функции Розенброка сходимость гораздо хуже, а выбор начально точки гораздо из-за наличия нескольких точек локального экстремума. При использовании сопряжённых градиентов в модификации Флетчера-Ривса точность сильно влияет на сходимость: бόльшая обеспечивает более оптимальное решение задачи одномерной минимизации и, как следствие, меньшее количество итераций. Метод Бройдена показал более хорошую сходимость.

В случае функции из варианта методы показали почти одинаковую, хорошую сходимость при любой точности. Оба метода нашли одинаковый экстремум, но при этом выбор начального приближения оказался критичным: для разных начальных приближений были получены разные результаты. Это связано с наличием нескольких экстремумов у функции.