МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»

(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ЛАБОРАТОРНАЯ РАБОТА №2

По основной образовательной программе подготовки бакалавров направлению 01.03.02 Прикладная математика и информатика профиль «Системное программирование»

Студент группы Б9121-02.03.01сцт

Москера Креспо Адриан Хосуэ

«24» декабря 2023 г.

Преподаватель кандидат физико-математических наук

_____ (подпись)

Яковлев Анатолий Александрович

«___» _____ 2023 г.

г. Владивосток

2023

Постановка задачи

Найти минимум функции $\mathbb{R}^n$

$$f(x) = \frac{1}{2}x^T Ax + bx$$

с условием $\|x - x_0\| \leq r$.

Исходные данные

A — произвольная симметрическая, невырожденная матрица, $A \in \mathbb{R}^{4\times4}$

b — произвольный ненулевой вектор, $b \in \mathbb{R}^4$

x_0 — произвольный начальный ненулевой вектор, $x \in \mathbb{R}^4$

r — радиус сферы

$$A = \begin{pmatrix} 8.917471161017044 & -5.881235445212047 & -3.748113781121384 & 3.0726928964843987 \\ -5.881235445212047 & -9.040202112715473 & -6.772212260246581 & 2.309845893033086 \\ -3.748113781121384 & -6.772212260246581 & 7.126513709691746 & 4.664488504098735 \\ 3.0726928964843987 & 2.309845893033086 & 4.664488504098735 & 5.724519936137147 \end{pmatrix}$$

Чтобы сгенерировать симметричную матрицу A, генерируется случайная временная матрица A_{temp} так, что $A = \frac{1}{2}(A_{\text{temp}} + A_{\text{temp}}^T)$.

$$b = \begin{pmatrix} -1.0034508506027269 \\ 8.494875723462954 \\ 3.3273150949612385 \\ 1.977512900042199 \end{pmatrix} \qquad x_0 = \begin{pmatrix} 9.902751075372013 \\ -9.113925983068228 \\ -1.288098758917016 \\ -5.869546741009071 \end{pmatrix}$$

$$r = 5.0$$

Решение

Найдём функцию Лагранжа:

$$L(x, y) = \frac{1}{2}x^T Ax + bx + y(\|x - x_0\|^2 - r^2)$$

Найдём точки минимума. Для этого возьмём частную производную по и приравняем её к нулю:

$$\frac{\partial L}{\partial x} = Ax + b + 2y(x - x_0) = 0$$

Рассмотрим два случая:

1. *Пусть* $y = 0$.

$Ax + b = 0$, тогда $x_* = -A^{-1}b$, где x_* — «подозрительная» на минимум точка.

$$x_* = \begin{pmatrix} 0.6147706419264252 \\ -6.822929360179975e-2 \\ 0.4621409715178318 \\ -1.024464312324732 \end{pmatrix}$$

$$f(x_*) = -0.8423471280686472$$

Проверим, подходит ли данная точка под условие $\|x - x_0\| \le r$.

$$\left\| \begin{pmatrix} 0.6147706419264252 \\ -6.822929360179975e-2 \\ 0.4621409715178318 \\ -1.024464312324732 \end{pmatrix} - \begin{pmatrix} 9.902751075372013 \\ -9.113925983068228 \\ -1.288098758917016 \\ -5.869546741009071 \end{pmatrix} \right\| = 13.95096312032364$$

$$\|x - x_0\| = 13.95096312032364 \le r = 5.0$$

Условие не выполняется. Таким образом, найденная точка не подходит под ограничения и не будет рассматриваться при выборе итогового ответа.

2. *Пусть* $y > 0$

Преобразуем L'_x и получим следующую систему уравнений из пяти уравнений:

$$\begin{cases} (A + 2Iy)x + (b - 2yx_0) = 0 \\ \|x - x_0\|^2 - r^2 = 0 \end{cases}$$

Для нахождения точек, подозрительных на оптимум, воспользуемся методом Ньютона:

$$x_{k+1} = x_k - f'^{-1}(x_k) - f(x_k)$$

где x_k — пятимерный вектор неизвестных, составленный из элементов вектора x и y.

$f(x_k)$ — левая часть данной системы, $f'(x_k)$ — матрица Якоби данной системы уравнений.

$$f'(x) = J = \begin{pmatrix} A + 2Iy & 2(x - x_0) \\ 2(x - x_0)^T & 0 \end{pmatrix}$$

Метод Ньютона будем запускать на нескольких начальных приближениях, т.к. функция может иметь несколько оптимальных точек. За начальное приближение берётся восемь точек:

$$x_1 = \begin{pmatrix} -6.5898083421499445 \\ 0.7592215884645075 \\ -4.226682137443012 \\ -6.119390680278865 \\ 8.627489224405448 \end{pmatrix} \qquad x_5 = \begin{pmatrix} -7.598796223848999 \\ 6.843339272245674 \\ -9.182634429570456 \\ -3.1893765108104084 \\ 9.206199154867335 \end{pmatrix}$$

$$x_2 = \begin{pmatrix} -2.6100865251413534 \\ 5.99242760986122 \\ -8.74927058223387 \\ -5.95634448776172 \\ 3.0078855539858225 \end{pmatrix} \qquad x_6 = \begin{pmatrix} 6.178983466733086 \\ 2.1890780465770114 \\ 4.1797457469462955 \\ -0.2392934769547459 \\ 1.816814793169577 \end{pmatrix}$$

$$x_3 = \begin{pmatrix} 1.3960738579100074 \\ 6.462717315110776 \\ -5.392336641442533 \\ 9.741069773234269 \\ 9.188512682496391 \end{pmatrix} \qquad x_7 = \begin{pmatrix} -2.292605667486429 \\ -6.746023009212916 \\ -4.696646180855078 \\ 7.4200281278856375 \\ 9.377417885952877 \end{pmatrix}$$

$$x_4 = \begin{pmatrix} 7.2108095713692295 \\ 6.869197562738942 \\ -9.983988083646999 \\ 4.03945099928122 \\ 4.600615303013031 \end{pmatrix} \qquad x_8 = \begin{pmatrix} 4.611794059659715 \\ 6.079254454194263 \\ 7.7980462506833135 \\ 7.846995291838499 \\ 2.9922584062467177 \end{pmatrix}$$

Условие для выхода из цикла:

$$\|x_{k+1} - x_k\| \leq \varepsilon,$$

где $\varepsilon = 10^{-6}$.

В результате получаем несколько точек x_i, подозрительных на оптимум:

i	**Начальное приближение**	x_i	y_i	$f(x_i)$
1	$\begin{pmatrix} -6.5898083 \\ 0.7592215 \\ -4.2266821 \\ -6.1193906 \\ 8.6274892 \end{pmatrix}$	$\begin{pmatrix} 6.061015807963965 \\ -11.876589956638057 \\ -2.111750020911596 \\ -4.480174039784763 \end{pmatrix}$	15.2331403	−138.21816218494826
2	$\begin{pmatrix} -2.6100865 \\ 5.9924276 \\ -8.7492705 \\ -5.9563444 \\ 3.0078855 \end{pmatrix}$	$\begin{pmatrix} 14.7481665631942 \\ -8.998465898153777 \\ -2.3565551605286443 \\ -6.475367374443787 \end{pmatrix}$	−17.7870006	1311.2159918285604
3	$\begin{pmatrix} 1.3960738 \\ 6.4627173 \\ -5.3923366 \\ 9.7410697 \\ 9.1885126 \end{pmatrix}$	$\begin{pmatrix} 6.758203006065774 \\ -13.175068797262252 \\ 11.216595144120248 \\ -21.872915990532427 \end{pmatrix}$	−1.0726899	1419.4616874561702

4	$\begin{pmatrix} 7.2108095 \\ 6.8691975 \\ -9.9839880 \\ 4.0394509 \\ 4.6006153 \end{pmatrix}$	$\begin{pmatrix} -12.799489605395173 \\ 3.9115356629638285 \\ 0.763414232449291 \\ 4.53999352717738 \end{pmatrix}$	-4.2064018	969.446868112623
5	$\begin{pmatrix} -7.5987962 \\ 6.8433392 \\ -9.1826344 \\ -3.1893765 \\ 9.2061991 \end{pmatrix}$	$\begin{pmatrix} 6.791822024079396 \\ -13.204440097576258 \\ 11.234563811710583 \\ -21.91831889661696 \end{pmatrix}$	-1.0673031	1426.0002410148386
6	$\begin{pmatrix} 6.1789834 \\ 2.1890780 \\ 4.1797457 \\ -0.2392934 \\ 1.8168147 \end{pmatrix}$	$\begin{pmatrix} -3.091764331166946 \\ -4.569631582300696 \\ 5.951991972588306 \\ -8.570411385348269 \end{pmatrix}$	-2.6509415	355.862001464922
7	$\begin{pmatrix} -2.2926056 \\ -6.7460230 \\ -4.6966461 \\ 7.4200281 \\ 9.3774178 \end{pmatrix}$	$\begin{pmatrix} 6.061015807964309 \\ -11.876589956638353 \\ -2.1117500209120843 \\ -4.4801740397843695 \end{pmatrix}$	15.2331403	-138.21816218496198
8	$\begin{pmatrix} 4.6117940 \\ 6.0792544 \\ 7.7980462 \\ 7.8469952 \\ 2.9922584 \end{pmatrix}$	$\begin{pmatrix} -36.51834885388397 \\ 24.63354895348917 \\ -11.913823909519436 \\ 36.57261085272549 \end{pmatrix}$	-8.0068537	9408.243073065663

Выясним, в какой из данных точек функция принимает минимальное значение. Отбросим результаты, полученные при $y < 0$, и получим, что минимальное значение функции $f(x)$ при заданных ограничениях достигается в точке:

$$x_{\min} = \begin{pmatrix} 6.061015807963965 \\ -11.876589956638057 \\ -2.111750020911596 \\ -4.480174039784763 \end{pmatrix}$$

Минимальное значение функции:

$$f_{\min}(x) = -138.21816218494826$$

Приложения (Я.П.: Haskell)

Приложение для этой работы было разделено на две части. Сначала была создана библиотека общего назначения, содержащая модуль под названием «lab1», содержащий все функции алгоритма. Затем основной файл бинарного файла

содержит вызовы функций для генерации матрицы, расчета и регистрации необходимых значений, а также создания графика рассчитанных значений.

Весь исходный код этого приложения можно найти по адресу https://github.com/AJMC2002/opt-methods/tree/main.

Зависимости

- chrono = "0.4.31" — для регистрации в логгере времени каждого запуска.
- log2 = "0.1.10" — библиотека ведения логов.
- nalgebra = "0.32.3" — библиотека линейной алгебры.
- plotters = "0.3.5" — библиотека построения графиков.
- utils = { path = "../utils" } — пользовательская библиотека с алгоритмами, используемыми для этой работы.

Библиотека

Библиотека использует три пользовательских враппер-типа: `type FloatingType`, `type __GenericSquareMatrix<const N: usize>` и `type __GenericVector<const N: usize>`. Они определяют общую степень точности, которой будут обладать наши значения, и быстрый способ записи универсальных типов матриц постоянного размера и векторов соответственно.

Первая функция `fn new_positive_definite_matrix` генерирует случайную обратимую матрицу M, для которой все ее элементы ограничены аргументами `min` и `max`, затем возвращает положительно определенную матрицу $M^T M$.

Далее `fn gradient_method` принимает наш начальный вектор x_0, параметры итерации и функцию первой производной от $f(x)$, чтобы вернуть вектор всех векторов x_i, полученных в нашем итерационном процессе.

```rust
// utils/src/lib.rs
pub type FloatingType = f64;

pub mod lab1 {
    use nalgebra::{ArrayStorage, Const, DimMin, SquareMatrix, Vector};

    use crate::FloatingType;

    pub type __GenericSquareMatrix<const N: usize> =
       SquareMatrix<FloatingType, Const<N>, ArrayStorage<FloatingType, N, N>>;
    pub type __GenericVector<const N: usize> =
        Vector<FloatingType, Const<N>, ArrayStorage<FloatingType, N, 1>>;

    pub fn new_positive_definite_matrix<const N: usize>(
        min: FloatingType,
        max: FloatingType,
    ) -> __GenericSquareMatrix<N>
    where
        Const<N>: DimMin<Const<N>, Output = Const<N>>,
    {
        loop {
```

```
            let m = (max - min) * __GenericSquareMatrix::<N>::new_random()
                + __GenericSquareMatrix::<N>::from_element(min);
            if m.determinant() > 0 as FloatingType {
                return m.transpose() * m;
            }
        }
    }

    pub fn gradient_method<const N: usize, F>(
        x0: &__GenericVector<N>,
        lambda: FloatingType,
        epsilon: FloatingType,
        f_prime: F,
    ) -> Vec<__GenericVector<N>>
    where
        F: Fn(&__GenericVector<N>) -> __GenericVector<N>,
    {
        let mut x_log = vec![*x0];
        loop {
            let x = x_log.last().unwrap();
            let x_next = x - lambda * f_prime(x);
            if (x_next - x).norm() < epsilon {
                break;
            } else {
                x_log.push(x_next)
            }
        }
        x_log
    }
}
```

Бинарный

Для наших вычислений нам нужно получить $f(x), f'(x), (f')^{-1}(x)$. Была определена структура для хранения этих различных определений функций с учетом их параметров A и b.

```
// minimization/src/main.rs
struct Function<const N: usize> {
    a: __GenericSquareMatrix<N>,
    b: __GenericVector<N>,
}

impl<const N: usize> Function<N> {
    fn new(a: __GenericSquareMatrix<N>, b: __GenericVector<N>) -> Self {
        Self { a, b }
    }

    pub fn f(&self, x: &__GenericVector<N>) -> FloatingType {
       (x.transpose() * self.a * x)[(0, 0)] / 2 as FloatingType + self.b.dotc(x)
    }
    pub fn f_prime(&self, x: &__GenericVector<N>) -> __GenericVector<N> {
```

```
        (self.a + self.a.transpose()) * x / 2 as FloatingType + self.b
    }

          pub   fn   f_prime_inv(&self,   f_prime_val:   __GenericVector<N>)   ->
__GenericVector<N> {
        2 as FloatingType
            * (self.a.transpose() + self.a).try_inverse().unwrap()
            * (f_prime_val - self.b)
    }
}
```

Следующая основная функция в конечном итоге оказывается довольно простой. Мы получаем нашу случайную матрицу A и векторы b и x_0, получаем нужные нам результаты, регистрируем их и генерируем наш график *.

* *Была написана пользовательская функция* `fn plot_steps`*, но поскольку ее реализация на самом деле не входит в рамки данной работы, она не будет обсуждаться подробно.*

```
// minimization/src/main.rs
use chrono::Local;
use log2::{debug, info};
use nalgebra::{matrix, vector, Vector6};
use plotters::prelude::*;
use std::error::Error;
use utils::{
           lab1::{__GenericSquareMatrix,   __GenericVector,   gradient_method,
new_positive_definite_matrix},
    FloatingType,
};

const MIN: FloatingType = -10.0;
const MAX: FloatingType = 10.0;
const LAMBDA: FloatingType = 0.0001;
const EPSILON: FloatingType = 0.000001;

fn main() -> Result<(), Box<dyn Error>> {
    let _log2 = log2::open("output.log").start();

    let a = new_positive_definite_matrix::<6>(MIN, MAX);
    let b = (MAX - MIN) * Vector6::new_random() + Vector6::from_element(MIN);
    let x0 = (MAX - MIN) * Vector6::new_random() + Vector6::from_element(MIN);

    let function = Function::new(a, b);

   let x_steps = gradient_method(&x0, LAMBDA, EPSILON, |x| function.f_prime(x));
    let x_exact = function.f_prime_inv(Vector6::zeros());

    let steps = x_steps.len() - 1;
    let intermediate_results = [
        x_steps[0],
        x_steps[steps / 4],
        x_steps[steps / 2],
```

```
        x_steps[3 * steps / 4],
        x_steps[steps],
    ];

    info!("a {}", a);
    info!("b {}", b);
    info!("x0 {}", x0);
    info!("tochnoe {}", x_exact);
    info!("xm {}", x_steps.last().unwrap());
    info!("m (steps) {}", steps);
    info!(
        "intermediate steps\n{}",
        intermediate_results
            .iter()
            .enumerate()
                                .map(|(i,  it)|  format!("x_{{{}m/{}}}\n{}",  i,
intermediate_results.len() - 1, it))
            .collect::<Vec<String>>()
            .join("")
    );
    info!("x (exact solution) {}", x_exact);
    info!(
        "f(intermediate steps)\n{}",
        intermediate_results
            .iter()
            .enumerate()
            .map(|(i, it)| format!(
                "f(x_{{{}m/{}}}) = {}\n",
                i,
                intermediate_results.len() - 1,
                function.f(it)
            ))
            .collect::<Vec<String>>()
            .join("")
    );
    info!("f(x) (exact solution) = {}", function.f(&x_exact));
    info!(
        "abs diff x vector = {}",
        (x_steps.last().unwrap() - x_exact).map(FloatingType::abs)
    );
    info!(
        "abs diff f(x) = {}",
        (function.f(x_steps.last().unwrap()) - function.f(&x_exact)).abs()
    );

    plot_steps(
        x_steps
            .iter()
            .map(|x| function.f(x))
            .collect::<Vec<FloatingType>>(),
    )?;
```

```
    Ok(())
}
```