МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики

Кафедра прикладной математики и экономико-математических методов

ЛАБОРАТОРНАЯ РАБОТА

на тему:

«Решение СЛАУ итерационным методом»

метод:

«Градиентные методы. Метод наискорейшего спуска - 1.2.4а»

Направление (специальность)_____01.03.02_____
(код, наименование)

Направленность (специализация)_____

Обучающийся _____Бронников Егор Игоревич_____
(Ф.И.О. полностью)

Группа___ПМ-1901__
(номер группы)

Проверил _____Хазанов Владимир Борисович_____
(*Ф.И.О. преподавателя)*

Должность_____профессор_____

Оценка _____ Дата: _____

Подпись: _____

Санкт-Петербург

2021

Оглавление

1. НЕОБХОДИМЫЕ ФОРМУЛЫ

Данные:

$Ax = f$

$A - положительно\ определённая\ и\ симметричная\ матрица$

$K_{max} - критерий\ прекращения\ итерационного\ процесса\ по\ числу\ итераций$

$x_0 - нулевой\ вектор\ (начальное\ приближение)$

Шаги метода наискорейшего спуска:

$r_k = f - Ax_k$

$min\, H(x_k) \Rightarrow \frac{d}{d\,\alpha_k} H(x_k) = 0 \Rightarrow \alpha_k = \frac{(r_k, r_k)}{(Ar_k, r_k)}$, $где\ x_k - текущее\ приближение$

$x_{k+1} = x_k + \alpha_k r_k$, $k = 1, \ldots, K_{max}$

2. ВХОДНЫЕ ДАННЫЕ

Матрица A

$$A = \begin{pmatrix} 4.33 & -1.12 & -1.08 & 1.14 \\ -1.12 & 4.33 & 0.24 & -1.22 \\ -1.08 & 0.24 & 7.21 & -3.22 \\ 1.14 & -1.22 & -3.22 & 5.43 \end{pmatrix}$$

Матрица f

$$\mathbf{f} = \begin{pmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{pmatrix}$$

$K_{max} = 10$

3. СКРИНШОТЫ ПРОГРАММЫ

Импорт модулей

```
import numpy as np         # для работы с матрицами и веторами
import warnings            # для работы с ошибками
import sympy as sp         # для красивого вывода промежуточных результатов
from IPython.display import Markdown, display  # для красивого вывода текста
```

Входные данные

```
A = np.matrix([[4.33, -1.12, -1.08, 1.14],
               [-1.12, 4.33, 0.24, -1.22],
               [-1.08, 0.24, 7.21, -3.22],
               [1.14, -1.22, -3.22, 5.43]],
               dtype=np.dtype(np.float64))
```

```
f = np.array([0.3, 0.5, 0.7, 0.9],
             dtype=np.dtype(np.float64))
```

Тестовые наборы данных

```
test_A1_Err = np.matrix([[1.00, 0.17, -0.25, 0.54],
                         [0.47, 1.00, 0.67, -0.32],
                         [-0.11, 0.35, 1.00, -0.74],
                         [0.55, 0.43, 0.36, 1.00]],
                         dtype=np.dtype(np.float64))
```

```
test_A2_Err = np.matrix([[-11.00, 6.00],
                         [6.00, -11.00]],
                         dtype=np.dtype(np.float64))
test_f2_Err = np.array([0.3, 0.5],
            dtype=np.dtype(np.float64))
```

Проверяет положительно определённая ли матрица

```
def is_positive_definite(A: np.matrix) -> bool:
    return np.all(np.linalg.eigvals(A) > 0)  # считаем собственные числа и проверяет, что все они больше 0
```

```
is_positive_definite(A)
```

```
True
```

Проверяет симметричная ли матрица

```
def is_symmetric(A: np.matrix) -> bool:
    return np.allclose(A, A.T)  # сравниваем обычную матрицу и транспонированную
                                # если они совпадают, то матрица симметричная
```

```
is_symmetric(A)
```

```
True
```

Алгоритм

```
def steepest_descent_method(A_arg: np.matrix, f_arg: np.array, K_max: int) -> np.array:
    A, f = np.copy(A_arg), np.copy(f_arg)  # копируем аргументы, чтобы их не 'пачкать'
    display(Markdown('<text style=font-weight:bold;font-size:16px;font-family:serif>Исходные данные<text>'),
        sp.BlockMatrix([sp.Matrix(A.round(decimals=10)), sp.Matrix(f.round(decimals=10))]))
    if not is_positive_definite(A):  # проверяем положительна ли определена матрица `A`
        warnings.warn("Матрица не является положительно определённой")  # печатаем ошибку
        return
    elif not is_symmetric(A):  # проверям симметричная ли матрица `A`
        warnings.warn("Матрица не является симметричной")  # печатаем ошибку
        return
    elif K_max < 0:  # проверяем `K_max`
        warnings.warn("Количество итераций не может быть отрицательным числом")  # печатаем ошибку
        return
    x = np.zeros(f.shape, dtype=np.dtype(np.float64))  # начальное приближение
    for k in range(K_max):  # итерируемся до `K_max`
        display(Markdown(f'<text style=font-weight:bold;font-size:16px;font-family:serif>{k+1} итерация<text>'),
                sp.Matrix(x.round(decimals=10)))
        r = np.squeeze(np.asarray(f - np.matmul(A, x)))  # находим вектор невязки
        alpha = (np.dot(r, r)/np.dot(np.matmul(A, r), r)).item(0)  # находим alpha
        x = x + alpha*r  # находим `k`-ое приближённое решение
    display(Markdown(f'<text style=font-weight:bold;font-size:16px;font-family:serif>Ответ<text>'),
            sp.Matrix(x.round(decimals=10)))
    return x
```

4. РЕЗУЛЬТАТЫ И ТЕСТЫ

Тесты

```
x = steepest_descent_method(A, f, 10)
```

Исходные данные

$$\left[\begin{bmatrix} 4.33 & -1.12 & -1.08 & 1.14 \\ -1.12 & 4.33 & 0.24 & -1.22 \\ -1.08 & 0.24 & 7.21 & -3.22 \\ 1.14 & -1.22 & -3.22 & 5.43 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{bmatrix}\right]$$

1 итерация

$$\begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

2 итерация

$$\begin{bmatrix} 0.1159775588 \\ 0.1932959314 \\ 0.2706143039 \\ 0.3479326764 \end{bmatrix}$$

3 итерация

$$\begin{bmatrix} 0.0985107399 \\ 0.2228601566 \\ 0.2605456268 \\ 0.3451615732 \end{bmatrix}$$

4 итерация

$$\begin{bmatrix} 0.099772049 \\ 0.2233106841 \\ 0.2589100115 \\ 0.347960791 \end{bmatrix}$$

5 итерация

$$\begin{bmatrix} 0.1000197017 \\ 0.2250170915 \\ 0.2607752527 \\ 0.34866444 \end{bmatrix}$$

6 итерация

$$\begin{bmatrix} 0.1003772689 \\ 0.2250728728 \\ 0.260373851 \\ 0.3494673588 \end{bmatrix}$$

7 итерация

$$\begin{bmatrix} 0.1004387563 \\ 0.2254805811 \\ 0.2609386966 \\ 0.3496940339 \end{bmatrix}$$

8 итерация

$$\begin{bmatrix} 0.1005313895 \\ 0.2254993065 \\ 0.2608236673 \\ 0.3499218645 \end{bmatrix}$$

9 итерация

$$\begin{bmatrix} 0.1005401597 \\ 0.2256156715 \\ 0.2609812639 \\ 0.3499883034 \end{bmatrix}$$

10 итерация

$$\begin{bmatrix} 0.1005660379 \\ 0.2256202222 \\ 0.2609492325 \\ 0.3500528971 \end{bmatrix}$$

Ответ

$$\begin{bmatrix} 0.1005680733 \\ 0.2256523012 \\ 0.2609940632 \\ 0.3500720528 \end{bmatrix}$$

```
steepest_descent_method(test_A1_Err, f, 10)
```

Исходные данные

$$\left[\begin{bmatrix} 1.0 & 0.17 & -0.25 & 0.54 \\ 0.47 & 1.0 & 0.67 & -0.32 \\ -0.11 & 0.35 & 1.0 & -0.74 \\ 0.55 & 0.43 & 0.36 & 1.0 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.5 \\ 0.7 \\ 0.9 \end{bmatrix}\right]$$

```
<ipython-input-119-f9ccc6ac74a8>:9: UserWarning: Матрица не является симметричной
  warnings.warn("Матрица не является симметричной")  # печатаем ошибку
```

```
steepest_descent_method(test_A2_Err, test_f2_Err, 10)
```

Исходные данные

$$\left[\begin{bmatrix} -11.0 & 6.0 \\ 6.0 & -11.0 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}\right]$$

```
<ipython-input-119-f9ccc6ac74a8>:6: UserWarning: Матрица не является положительно определённой
  warnings.warn("Матрица не является положительно определённой")  # печатаем ошибку
```

5. ОЦЕНКА ТОЧНОСТИ ПОЛУЧЕННОГО РЕШЕНИЯ

Оценка точности полученного решения

```
r = f - np.matmul(A, x)
display(Markdown(f'<text style=font-weight:bold;font-size:16px;font-family:serif>Вектор невязки<text>'),
        sp.Matrix(r.T))
```

Вектор невязки

$$\begin{bmatrix} 6.22681101437594 \cdot 10^{-5} \\ 1.11072297926951 \cdot 10^{-5} \\ -7.82185327168339 \cdot 10^{-5} \\ 0.000157840633184247 \end{bmatrix}$$