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 проблемы собственных значений для для матриц»

метод:

«QR-алгоритм со сдвигом- 3.3.6б»

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 ДЛЯ QR-АЛГОРИТМА.....3
2. МОДИФИЦИРОВАННЫЙ АЛГОРИТМ ГРАМА-ШМИДТА.....4
2. ВХОДНЫЕ ДАННЫЕ.....5
3. СКРИНШОТЫ ПРОГРАММЫ.....6
4. РЕЗУЛЬТАТЫ И ТЕСТЫ.....8

1. НЕОБХОДИМЫЕ ФОРМУЛЫ ДЛЯ QR-АЛГОРИТМА

Данные:

A – *матрица* $(n \times n)$

K_{max} – *критерий прекращения итерационного процесса по числу итераций*

δ – *критерий прекращения итерационного процесса по малости двух соседних приближений*

Шаги QR-алгоритма со сдвигом:

$A_k - t_k = Q_k R_k$, *где* Q_k – *унитарная матрица* , R_k – *верхняя треугольная матрица*

$A_{k+1} = R_k Q_k + t_k I$

$t_{k+1} = a^{(k)}_{nn}$

$k = 1, \ldots, K_{max}$

2. МОДИФИЦИРОВАННЫЙ АЛГОРИТМ ГРАМА-ШМИДТА

Данные:

$A - матрица(n \times n)$

Шаги алгоритма:

$s=0$

$s=s+a_{jk}^2,\ j=1,\ldots,n$

$r_{kk}=\sqrt{s}$

$q_{jk}=\frac{a_{jk}}{r_{kk}},\ j=1,\ldots,n$

$s=0;\ s=s+a_{ji}*q_{jk},\ j=1,\ldots,n;\ r_{ki}=s;\ a_{ji}=a_{ji}-r_{ki}*q_{jk},\ i=k+1,\ldots,n$

$k=1,\ldots,n$

Этот алгоритм будет нужен для нахождения QR-разложения

2. ВХОДНЫЕ ДАННЫЕ

Матрица A_1

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

Матрица A_2

$$A = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}$$

$K_{max} = 20$

$\delta = 10^{-10}$

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

```
A1 = 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))
```

```
A2 = np.matrix([[1.00, 0.42, 0.54, 0.66],
                [0.42, 1.00, 0.32, 0.44],
                [0.54, 0.32, 1.00, 0.22],
                [0.66, 0.44, 0.22, 1.00]],
                dtype=np.dtype(np.float64))
```

Модифицированный алгоритм Грама-Шмидта для нахождения QR-разложения

```
def qr_mod_gram_schmidt(A_arg: np.matrix):
    A = np.copy(A_arg)
    n = A.shape[0]
    R, Q = np.zeros(A.shape), np.zeros(A.shape)
    for k in range(n):
        s = 0
        for j in range(n):
            s += A[j, k]**2
            R[k, k] = np.sqrt(s)
        for j in range(n): Q[j, k] = A[j, k]/R[k, k]
        for i in range(k, n):
            s = 0
            for j in range(n):
                s += A[j, i] * Q[j, k]
                R[k, i] = s
            for j in range(n): A[j, i] = A[j, i] - R[k, i] * Q[j, k]
    return np.asmatrix(Q), np.asmatrix(R)
```

QR-алгоритм

```
def qr_mod_algorithm(A: np.matrix, Kmax: int, delta: float) -> np.array:
    if Kmax < 1:
        warnings.warn("Количество итераций должно быть положительным числом")
        return
    Ak = A
    t = 0
    I = np.identity(A.shape[0])
    eigvals = []
    d = delta
    k = 0
    while k < Kmax and d <= delta:
        Q, R = qr_mod_gram_schmidt(A - t * I)
        Ak = np.matmul(R,Q) + t * I if k else np.matmul(R, Q)
        t = A[-1, -1]
        eigvals.append(np.diagonal(Ak))
        d = np.linalg.norm(eigvals[-1] - eigvals[-2]) if k > 2 else delta
        k += 1
    return eigvals[-1]
```

4. РЕЗУЛЬТАТЫ И ТЕСТЫ

Результаты

```
real_res = qr_mod_algorithm(A1, 20, 10**-10)
np_res = np.linalg.eigvals(A1)
display(Markdown('<text style=font-weight:bold;font-size:16px;font-family:serif>Полученный ответ<text>'),
            sp.Matrix(real_res.round(decimals=10)))
display(Markdown('<text style=font-weight:bold;font-size:16px;font-family:serif>Встроенная функция<text>'),
            sp.Matrix(np_res.round(decimals=10)))
```

Полученный ответ

$$\begin{bmatrix} 5.9990004868 \\ 5.6427240738 \\ 5.3813841267 \\ 4.2768913128 \end{bmatrix}$$

Встроенная функция

$$\begin{bmatrix} 10.3267786405 \\ 5.1025199601 \\ 3.3389380551 \\ 2.5317633444 \end{bmatrix}$$

```
real_res = qr_mod_algorithm(A2, 20, 10**-10)
np_res = np.linalg.eigvals(A2)
r = np_res - real_res
display(Markdown('<text style=font-weight:bold;font-size:16px;font-family:serif>Полученный ответ<text>'),
            sp.Matrix(real_res.round(decimals=10)))
display(Markdown('<text style=font-weight:bold;font-size:16px;font-family:serif>Встроенная функция<text>'),
            sp.Matrix(np_res.round(decimals=10)))
```

Полученный ответ

$$\begin{bmatrix} 1.6041434263 \\ 0.9009086984 \\ 0.7265559211 \\ 0.7683919542 \end{bmatrix}$$

Встроенная функция

$$\begin{bmatrix} 2.3227488001 \\ 0.2422607083 \\ 0.6382838028 \\ 0.7967066889 \end{bmatrix}$$