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 или матричного уравнения прямым методом»

метод:

«**Метод Гаусса с частичным выбором ведущего элемента - 1.1.1г(в)**»

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_____

Оценка _____ Дата: _____

Подпись: _____

Санкт-Петербург

2021

Оглавление

1. НЕОБХОДИМЫЕ ФОРМУЛЫ

Этапы метода Гаусса

$(A|F) \Rightarrow (R|G) \Rightarrow (I|X)$, где $|A| \neq 0$

$A - n \times n$

$F - n \times l$

$X - n \times l$

Прямой ход

$$R_{k^*} = a_{kk}^{(k-1)^{-1}} A_{k^*}^{(k-1)}; G_{k^*} = a_{kk}^{(k-1)^{-1}} F_{k^*}^{(k-1)}, k = 1,2,\dots,n$$

$$A_{i^*}^{(k)} = A_{i^*}^{(k-1)} - a_{ik}^{(k-1)} R_{k^*}; F_{i^*}^{(k)} = F_{i^*}^{(k-1)} - a_{ik}^{(k-1)} G_{k^*}, i = k+1,\dots,n$$

Обратный ход

$$X_{n^*} = G_{n^*}$$

$$X_{k^*} = G_{k^*} - \sum_{j=k+1}^{n} r_{kj} X_{j^*}, k = n-1,\dots,1$$

2. ВХОДНЫЕ ДАННЫЕ

Матрица A

$$A = \begin{pmatrix} 1.00 & 0.17 & -0.25 & 0.54 \\ 0.47 & 1.00 & 0.67 & -0.32 \\ -0.11 & 0.35 & 1.00 & -0.74 \\ 0.55 & 0.43 & 0.36 & 1.00 \end{pmatrix}$$

Матрица f

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

3. СКРИНШОТЫ ПРОГРАММЫ

Импорт модулей

```python
import numpy as np          # для работы с матрицами и веторами
from typing import Union    # для работы с типизацией
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

```python
A = 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))
```

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

Тестовые наборы данных

```python
test_A1 = np.matrix([[ 2. , -1.4,  0. ],
                     [-0.6,  0.4,  1.2],
                     [ 1. , -0.2,  1. ]],
                    dtype=np.dtype(np.float64))

test_f1 = np.array([1.4, 0.8, 1.2],
                   dtype=np.dtype(np.float64))
```

```python
test_A2 = np.matrix([[0.25, 0.5 ],
                     [0.75, 1.  ]],
                     dtype=np.dtype(np.float64))

test_f2 = np.matrix([[0.75, 1.25],
                     [1.25, 2.25]],
                     dtype=np.dtype(np.float64))
```

```python
test_A3 = np.matrix([[0.1, 0.2, 0.3],
                     [0.4, 0.5, 0.6],
                     [0.1, 0. , 0.1]],
                     dtype=np.dtype(np.float64))

test_f3 = np.array([0.1, 0.1, 0.1],
                   dtype=np.dtype(np.float64))
```

```python
test_A4 = np.matrix([[ 0.5,  0.5,  1. ,  1.5],
                     [ 0.5,  1. ,  1.5, -0.5],
                     [ 1.5, -0.5, -0.5, -1. ],
                     [ 1. ,  1.5, -0.5, -0.5]],
                     dtype=np.dtype(np.float64))

test_f4 = np.array([0.5, -2., -2., -3.],
                   dtype=np.dtype(np.float64))
```

```python
test_A5 = np.matrix([[ 0.5, -1. , -0.5],
                     [-1.5,  1. ,  1. ],
                     [ 1.5, -0.5, -1. ]],
                     dtype=np.dtype(np.float64))

test_f5 = np.matrix([[ 0.5,  0. ],
                     [ 1. , -1. ],
                     [-1.5,  0.5]],
                     dtype=np.dtype(np.float64))
```

```python
test_A_Err = np.matrix([[1, 1, 3], [1, 1, 3], [2, -1, 4]],
                       dtype=np.dtype(np.float64))
test_f_Err = np.array([0., 1., 2.],
                      dtype=np.dtype(np.float64))
```

Алгоритм

```python
def gaussian_elimination(A_arg: np.matrix, f_arg: Union[np.matrix, np.array]) -> Union[np.matrix, 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))]))
    for i in range(len(A)):
        column = np.abs(A[i:, i])      # берём i-ую колонку по модулю
        leading_elem = np.max(column)  # методом частичного выбора находим ведущий элемент
        if leading_elem == 0.:  # проверяем определитель (if ведущий элемент == 0, то det(A) = 0 => решений нет)
            warnings.warn("Определитель равен 0")  # печатаем ошибку
            return  # заканчиваем выполнение программы
        if np.where(column == leading_elem)[0][0] != 0:  # нужно ли нам менять строки (?)
            pos_max = column.argmax() + i      # узнаём номер строки ведущего элемента
            A[[i, pos_max]] = A[[pos_max, i]]  # меняем строки местами в матрице A
            f[[i, pos_max]] = f[[pos_max, i]]  # меняем строки местами в матрице f
        for j in range(i+1, len(A)):   # делаем верхний треугольник
            coef = -(A[j, i]/A[i, i])  # считаем коэффициент
            A[j] = coef * A[i] + A[j]  # домножаем `i` строку и прибавляем `j`
            f[j] = coef * f[i] + f[j]
        display(Markdown(f'<text style=font-weight:bold;font-size:16px;font-family:serif>{i+1} итерация<text>'),
                sp.BlockMatrix([sp.Matrix(A.round(decimals=10)), sp.Matrix(f.round(decimals=10))]))  # выводим промежуточный результат
    n = f.shape[0]  # размерность нашего ответа
    X = np.zeros(shape=f.shape)   # заполняем наше будущее решение нулями
    X[n-1] = f[n-1]/A[n-1, n-1]   # решаем последнее уравнение
    for i in range(n-2, -1, -1):  # рассчитывает значения начиная с конца
        sum_elem = sum(A[i, j] * X[j] for j in range(i+1, n))  # для известных `x` суммируем коэффициенты
        X[i] = (f[i] - sum_elem)/A[i, i]  # находим `x`
    display(Markdown('<text style=font-weight:bold;font-size:16px;font-family:serif>Ответ<text>'),
            sp.Matrix(X.round(decimals=10)))  # выводим ответ
    return X  # возвращаем ответ для проверки результата
```

4. РЕЗУЛЬТАТЫ И ТЕСТЫ

Тесты

```
1  gaussian_elimination(test_A_Err, test_f_Err)
```

Исходные данные

$$\left[\begin{bmatrix} 1.0 & 1.0 & 3.0 \\ 1.0 & 1.0 & 3.0 \\ 2.0 & -1.0 & 4.0 \end{bmatrix} \begin{bmatrix} 0.0 \\ 1.0 \\ 2.0 \end{bmatrix}\right]$$

1 итерация

$$\left[\begin{bmatrix} 2.0 & -1.0 & 4.0 \\ 0.0 & 1.5 & 1.0 \\ 0.0 & 1.5 & 1.0 \end{bmatrix} \begin{bmatrix} 2.0 \\ 0.0 \\ -1.0 \end{bmatrix}\right]$$

2 итерация

$$\left[\begin{bmatrix} 2.0 & -1.0 & 4.0 \\ 0.0 & 1.5 & 1.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} 2.0 \\ 0.0 \\ -1.0 \end{bmatrix}\right]$$

```
<ipython-input-11-d38b6c41687f>:9: UserWarning: Определитель равен 0
  warnings.warn("Определитель равен 0")  # печатаем ошибку
```

```
1  np.testing.assert_allclose(np.linalg.solve(A, f),
2                             gaussian_elimination(A, f))
```

Исходные данные

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

1 итерация

$$\left[\begin{bmatrix} 1.0 & 0.17 & -0.25 & 0.54 \\ 0.0 & 0.9201 & 0.7875 & -0.5738 \\ 0.0 & 0.3687 & 0.9725 & -0.6806 \\ 0.0 & 0.3365 & 0.4975 & 0.703 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.359 \\ 0.733 \\ 0.735 \end{bmatrix}\right]$$

2 итерация

$$\left[\begin{bmatrix} 1.0 & 0.17 & -0.25 & 0.54 \\ 0.0 & 0.9201 & 0.7875 & -0.5738 \\ 0.0 & 0.0 & 0.6569351157 & -0.4506684056 \\ 0.0 & 0.0 & 0.2094946201 & 0.9128507771 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.359 \\ 0.5891424845 \\ 0.6037061189 \end{bmatrix}\right]$$

3 итерация

$$\left[\begin{bmatrix} 1.0 & 0.17 & -0.25 & 0.54 \\ 0.0 & 0.9201 & 0.7875 & -0.5738 \\ 0.0 & 0.0 & 0.6569351157 & -0.4506684056 \\ 0.0 & 0.0 & 0.0 & 1.0565675677 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.359 \\ 0.5891424845 \\ 0.4158303637 \end{bmatrix}\right]$$

4 итерация

$$\left[\begin{bmatrix} 1.0 & 0.17 & -0.25 & 0.54 \\ 0.0 & 0.9201 & 0.7875 & -0.5738 \\ 0.0 & 0.0 & 0.6569351157 & -0.4506684056 \\ 0.0 & 0.0 & 0.0 & 1.0565675677 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.359 \\ 0.5891424845 \\ 0.4158303637 \end{bmatrix}\right]$$

Ответ

$$\begin{bmatrix} 0.4408885509 \\ -0.3630309901 \\ 1.1667983323 \\ 0.3935672231 \end{bmatrix}$$

```
np.testing.assert_allclose(np.linalg.solve(test_A1, test_f1),
                           gaussian_elimination(test_A1, test_f1))
```

Исходные данные

$$\left[\left[\begin{array}{ccc} 2.0 & -1.4 & 0.0 \\ -0.6 & 0.4 & 1.2 \\ 1.0 & -0.2 & 1.0 \end{array}\right] \quad \left[\begin{array}{c} 1.4 \\ 0.8 \\ 1.2 \end{array}\right]\right]$$

1 итерация

$$\left[\left[\begin{array}{ccc} 2.0 & -1.4 & 0.0 \\ 0.0 & -0.02 & 1.2 \\ 0.0 & 0.5 & 1.0 \end{array}\right] \quad \left[\begin{array}{c} 1.4 \\ 1.22 \\ 0.5 \end{array}\right]\right]$$

2 итерация

$$\left[\left[\begin{array}{ccc} 2.0 & -1.4 & 0.0 \\ 0.0 & 0.5 & 1.0 \\ 0.0 & 0.0 & 1.24 \end{array}\right] \quad \left[\begin{array}{c} 1.4 \\ 0.5 \\ 1.24 \end{array}\right]\right]$$

3 итерация

$$\left[\left[\begin{array}{ccc} 2.0 & -1.4 & 0.0 \\ 0.0 & 0.5 & 1.0 \\ 0.0 & 0.0 & 1.24 \end{array}\right] \quad \left[\begin{array}{c} 1.4 \\ 0.5 \\ 1.24 \end{array}\right]\right]$$

Ответ

$$\left[\begin{array}{c} 0.0 \\ -1.0 \\ 1.0 \end{array}\right]$$

```
np.testing.assert_allclose(np.linalg.solve(test_A2, test_f2),
                           gaussian_elimination(test_A2, test_f2))
```

Исходные данные

$$\left[\left[\begin{array}{cc} 0.25 & 0.5 \\ 0.75 & 1.0 \end{array}\right] \quad \left[\begin{array}{cc} 0.75 & 1.25 \\ 1.25 & 2.25 \end{array}\right]\right]$$

1 итерация

$$\left[\left[\begin{array}{cc} 0.75 & 1.0 \\ 0.0 & 0.1666666667 \end{array}\right] \quad \left[\begin{array}{cc} 1.25 & 2.25 \\ 0.3333333333 & 0.5 \end{array}\right]\right]$$

2 итерация

$$\left[\left[\begin{array}{cc} 0.75 & 1.0 \\ 0.0 & 0.1666666667 \end{array}\right] \quad \left[\begin{array}{cc} 1.25 & 2.25 \\ 0.3333333333 & 0.5 \end{array}\right]\right]$$

Ответ

$$\left[\begin{array}{cc} -1.0 & -1.0 \\ 2.0 & 3.0 \end{array}\right]$$

```
np.testing.assert_allclose(np.linalg.solve(test_A4, test_f4),
                           gaussian_elimination(test_A4, test_f4))
```

Исходные данные

$$\left[\begin{bmatrix} 0.5 & 0.5 & 1.0 & 1.5 \\ 0.5 & 1.0 & 1.5 & -0.5 \\ 1.5 & -0.5 & -0.5 & -1.0 \\ 1.0 & 1.5 & -0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 0.5 \\ -2.0 \\ -2.0 \\ -3.0 \end{bmatrix}\right]$$

1 итерация

$$\left[\begin{bmatrix} 1.5 & -0.5 & -0.5 & -1.0 \\ 0.0 & 1.1666666667 & 1.6666666667 & -0.1666666667 \\ 0.0 & 0.6666666667 & 1.1666666667 & 1.8333333333 \\ 0.0 & 1.8333333333 & -0.1666666667 & 0.1666666667 \end{bmatrix} \begin{bmatrix} -2.0 \\ -1.3333333333 \\ 1.1666666667 \\ -1.6666666667 \end{bmatrix}\right]$$

2 итерация

$$\left[\begin{bmatrix} 1.5 & -0.5 & -0.5 & -1.0 \\ 0.0 & 1.8333333333 & -0.1666666667 & 0.1666666667 \\ 0.0 & 0.0 & 1.2272727273 & 1.7727272727 \\ 0.0 & 0.0 & 1.7727272727 & -0.2727272727 \end{bmatrix} \begin{bmatrix} -2.0 \\ -1.6666666667 \\ 1.7727272727 \\ -0.2727272727 \end{bmatrix}\right]$$

3 итерация

$$\left[\begin{bmatrix} 1.5 & -0.5 & -0.5 & -1.0 \\ 0.0 & 1.8333333333 & -0.1666666667 & 0.1666666667 \\ 0.0 & 0.0 & 1.7727272727 & -0.2727272727 \\ 0.0 & 0.0 & 0.0 & 1.9615384615 \end{bmatrix} \begin{bmatrix} -2.0 \\ -1.6666666667 \\ -0.2727272727 \\ 1.9615384615 \end{bmatrix}\right]$$

4 итерация

$$\left[\begin{bmatrix} 1.5 & -0.5 & -0.5 & -1.0 \\ 0.0 & 1.8333333333 & -0.1666666667 & 0.1666666667 \\ 0.0 & 0.0 & 1.7727272727 & -0.2727272727 \\ 0.0 & 0.0 & 0.0 & 1.9615384615 \end{bmatrix} \begin{bmatrix} -2.0 \\ -1.6666666667 \\ -0.2727272727 \\ 1.9615384615 \end{bmatrix}\right]$$

Ответ

$$\begin{bmatrix} -1.0 \\ -1.0 \\ 0.0 \\ 1.0 \end{bmatrix}$$

```
np.testing.assert_allclose(np.linalg.solve(test_A5, test_f5),
                           gaussian_elimination(test_A5, test_f5))
```

Исходные данные

$$\left[\begin{bmatrix} 0.5 & -1.0 & -0.5 \\ -1.5 & 1.0 & 1.0 \\ 1.5 & -0.5 & -1.0 \end{bmatrix} \begin{bmatrix} 0.5 & 0.0 \\ 1.0 & -1.0 \\ -1.5 & 0.5 \end{bmatrix}\right]$$

1 итерация

$$\left[\begin{bmatrix} -1.5 & 1.0 & 1.0 \\ 0.0 & -0.6666666667 & -0.1666666667 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} 1.0 & -1.0 \\ 0.8333333333 & -0.3333333333 \\ -0.5 & -0.5 \end{bmatrix}\right]$$

2 итерация

$$\left[\begin{bmatrix} -1.5 & 1.0 & 1.0 \\ 0.0 & -0.6666666667 & -0.1666666667 \\ 0.0 & 0.0 & -0.125 \end{bmatrix} \begin{bmatrix} 1.0 & -1.0 \\ 0.8333333333 & -0.3333333333 \\ 0.125 & -0.75 \end{bmatrix}\right]$$

3 итерация

$$\left[\begin{bmatrix} -1.5 & 1.0 & 1.0 \\ 0.0 & -0.6666666667 & -0.1666666667 \\ 0.0 & 0.0 & -0.125 \end{bmatrix} \begin{bmatrix} 1.0 & -1.0 \\ 0.8333333333 & -0.3333333333 \\ 0.125 & -0.75 \end{bmatrix}\right]$$

Ответ

$$\begin{bmatrix} -2.0 & 4.0 \\ -1.0 & -1.0 \\ -1.0 & 6.0 \end{bmatrix}$$

5. ССЫЛКА НА ПРОГРАММУ

К сожалению, Google Colab работает на старой версии Python и на старой версии модул, поэтому если запускать код там, то он будет выдавать ошибки. Если же скачать файл и открыть его через Jupyter Lab, то всё должно быть хорошо.