МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ТЭВН

Лабораторная работа № 2

“Решение систем линейных алгебраических уравнений”

Вариант № 19

Факультет: ФЭН

Группа: ЭН2-31

Студент: Полозов А.А.

Преподаватель: Петрищев А.В.

Новосибирск 2024

# 1. Цель работы

Изучение операция по созданию и работе с массивами numpy, индексации произвольной группы элементов, операций над матрицами. Освоение методов решения систем линейных алгебраических уравнений (СЛАУ).

# 2. Задание № 1

Написать программу, создающую матрицу, заданную в соответствии с вариантом в приложении № 1. Программа не должна использовать вложенные циклы, конструкции ветвления и встроенные функции python по работе с матрицами.

# 3. Пояснения к заданию № 1

Варианту № 19 соответствует матрица

Программа должна создать нулевую матрицу размером , заполнить нулевую строку единицами, далее в каждой -ой строке от нулевого до -го столбца поменять значение элементов на .

Блок-схема программы представлена на рис. № 1.



Рис. № 1 – Блок-схема для задания № 1

## Листинг программы

import numpy as np

# Задание размера матрицы

n = 10

# Объявление матрицы

A = np.zeros((n, n), dtype = np.int32)

# Заполнение нулевой строки единицами

A[0:1] = 1

# Цикл 1

for i in range(1, n):

    # Заполнение среза i-ой строки до n-i-го столбца

    A[i:i+1, :-i] = i+1

print(A)

## Результат работы программы

Результат работы представлен на рис. № 2.

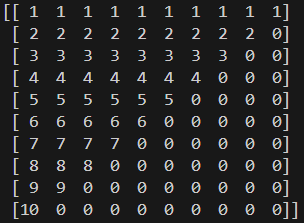


Рис. № 2 – Результат работы программы

# 4. Задание № 2

Выполнить расчёт токов в ветвях электрической схемы, заданной в приложении № 2.

1. По заданной системе уравнений Кирхгофа получить матрицу коэффициентов и матрицу свободных членов системы уравнений.
2. Написать программу, выполняющую решение полученной системы уравнений методом, заданным в варианте и соответствующую требованиям.
3. Протестировать полученный алгоритм путём сравнения результата с решением, полученным при помощи функции numpy.linalg.solve.

# 5. Пояснения к заданию № 2

Варианту № 19 соответствует следующая система уравнений Кирхгофа, которую необходимо решить через обратную матрицу:

Приведём систему к общему виду:

Этой СЛАУ будет соответствовать матрицы

Решение СЛАУ будет представлять из себя выражение

Блок-схемы алгоритмов представлены на рис. № 3 – 5.

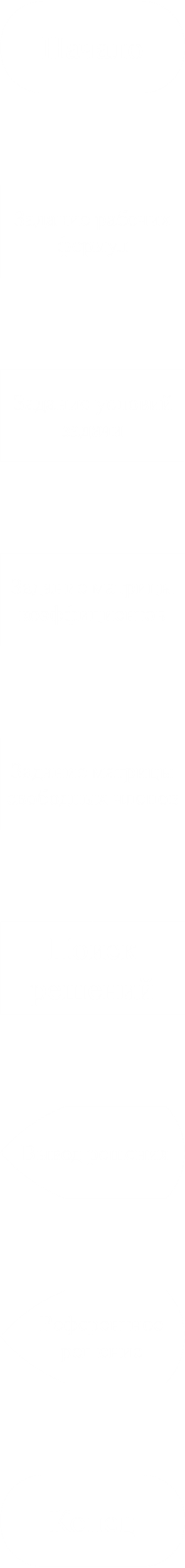


Рис. № 3 – Блок-схема для задания № 2

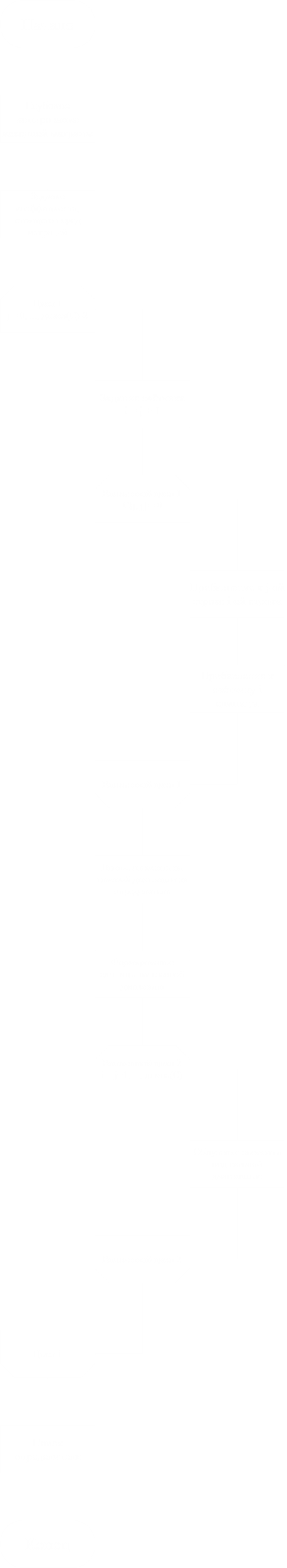


Рис. № 4 – Поиск определителя

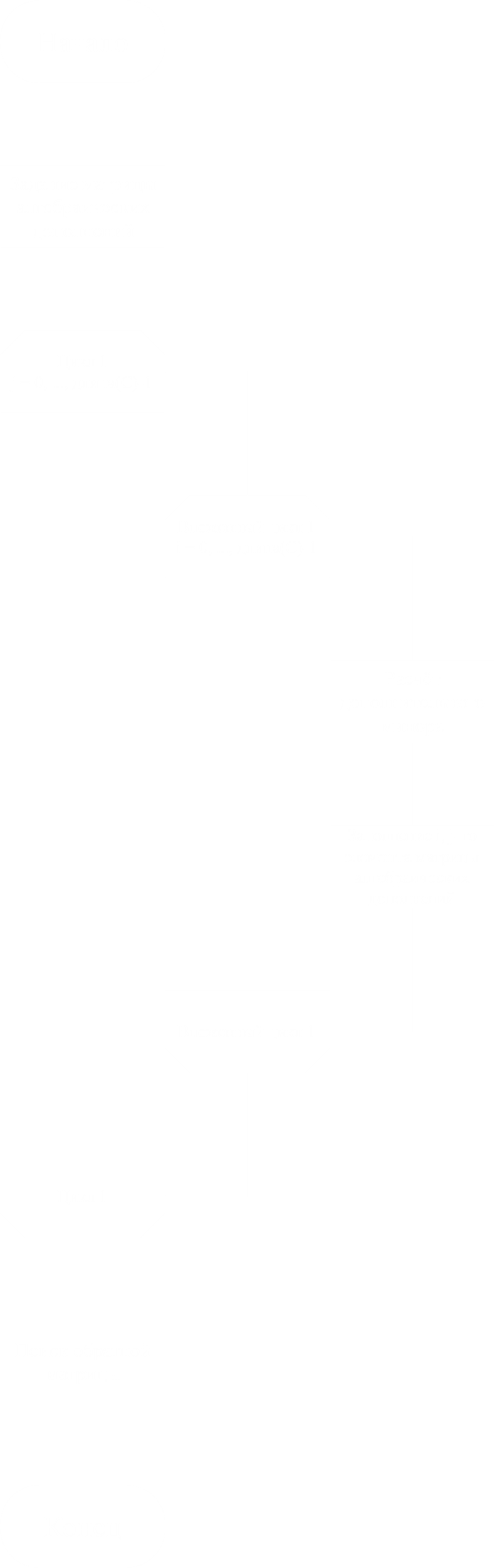


Рис. № 5 – Поиск обратной матрицы

Листинг программы

import numpy as np

# Рабочие формулы

# Поиск определителя

def opr(A):

    # Глубокое копирование исходной матрицы

    C = np.copy(A)

    C = C.astype('float64')

    #  Коэффициент, стоящий перед матрицей

    K = 1

    # Цикл 1

    for j in range(len(C)-1):

        ## Устранение нулей на главной диагонали

        # Задание счётчика

        i = j + 1

        # Вложенный цикл 1

        while C[j, j] == 0:

            # Прибавление к j-ой строке i-ой строки

            C[j:j+1] += C[i:i+1]

            # Прибавление к счётчику i единицы

            i += 1

        # Вынос элемента на главной диагонали за определитель

        K \*= C[j, j]

        # Формирование единицы на главной диагонали

        C[j:j+1] = C[j:j+1]\*(1/C[j, j])

        # Вложенный цикл 2

        for i in range(j+1, len(C)):

            # Обнуление элемента под главной диагональю

            C[i:i+1] += -(C[j:j+1])\*C[i, j]

    # Поиск определителя (элемент в правом нижнем углу матрицы является определителем)

    return K\*C[len(C)-1, len(C)-1]

# Поиск обратной матрицы

def obr(A):

    # Задание матрицы алгебраических дополнений

    C = np.zeros((len(A), len(A)), dtype='float64')

    # Заполнение матрицы алгебраических дополнений

    # Цикл 1

    for i in range(len(C)):

        # Вложенный цикл 1

        for j in range(len(C)):

            # Расчёт дополнительного минора

            M = np.delete(A, i, 0)

            M = np.delete(M, j, 1)

            m = opr(M)

            # Заполнение i, j-го элемента матрицы алгебраических дополнений

            C[i, j] = np.power(-1, i+j) \* m

    # Поиск обратной матрицы

    return (1/opr(A))\*np.transpose(C)

# Задание условий задачи

R1 = 1

R2 = 7

R3 = 5

R4 = 4

R5 = 1

R6 = 1

E1 = 1

E2 = 1

E3 = 1

# Задание матрицы коэффициентов

A = np.array([

    [0, -1, 1, 0, 0, -1],

    [0, 1, 0, -1, 1, 0],

    [-1, 0, -1, 0, -1, 0],

    [0, -R4, -R3, 0, R5, 0],

    [0, -R4, 0, R1, 0, R6],

    [R2, 0, 0, R1, R5, 0]

])

# Задание матрицы свободных членов

B = np.array([

    [0],

    [0],

    [0],

    [0],

    [-E1-E2],

    [E1+E3]

])

# Поиск решений

X = obr(A) @ B

# Вывод решения

print()

print('Решение с помощью программы')

print(X)

print()

# Референтное решение

print('Референтное решение')

print(np.linalg.solve(A, B))

print()

## Результат работы программы

Результат работы программы представлен на рис. № 6.

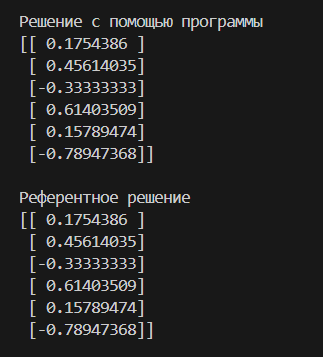


Рис. № 6 – Результат работы программы

# 6. Выводы

Были изучены операции по созданию и работе с массивами numpy, индексации произвольной группы элементов, операций над матрицами. Освоены методы решения систем линейных алгебраических уравнений (СЛАУ).