МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ОТЧЕТ

к лабораторной работе №6 по дисциплине
«Вычислительная математика»

Направление подготовки
01.03.02 «Прикладная математика и информатика»

Выполнил студент
гр. Б9119-01.03.02систпро
Нагорнов С.С.
(ФИО) *(Подпись)*
«20» июня 2021 г.

г. Владивосток
2021

Содержание

Введение **2**

Постановка задачи **2**

Алгоритм работы **2**

Приложения **3**

Вывод **4**

Введение

В данной лабораторной работе необходимо найти обратную матрицу с помощью метода окаймления.

Постановка задачи

Дана матрица A. Необходимо найти матрицу A^{-1} методом окаймления.

Алгоритм работы

1. Представим матрицу $A = A_n$. Данную матрицу мы представляем в следующем виде:

$$A_n = \begin{pmatrix} A_{n-1} & u_n \\ \upsilon_n & a_{nn} \end{pmatrix}$$

$\upsilon_n = (a_{n1}, a_{n2}, ..., a_{nn-1})$, $u_n = (a_{1n}, a_{2n}, ..., a_{n-1n})$, A_{n-1} – матрица $n-1$-го порядка.

2. Матрицу A^{-1} также будем представлять окаймленной:

$$A_n^{n-1} = \begin{pmatrix} P_{n-1} & r_n \\ q_n & \dfrac{1}{\alpha_n} \end{pmatrix}$$

.

3. Коэффициенты окаймленной обратной матрицы будет искать по следующим формулам:

$$r_n = -\frac{A_{n-1}^{-1} u_n}{\alpha_n}, \quad \alpha_n = a_{nn} - \upsilon_n A_{n-1}^{-1} u_n,$$

$$P_{n-1} = A_{n-1}^{-1} - A_{n-1}^{-1} u_n q_n, \quad q_n = -\frac{\upsilon_n A_{n-1}^{-1}}{\alpha_n}$$

.

4. Таким образом, получаем такую матрицу:

$$A^{-1} = \begin{pmatrix} A_{n-1}^{-1} - \dfrac{A_{n-1}^{-1} u_n \upsilon_n A_{n-1}^{-1}}{a_{nn} - \upsilon_n A_{n-1}^{-1} u_n} & -\dfrac{A_{n-1}^{-1} u_n}{a_{nn} - \upsilon_n A_{n-1}^{-1} u_n} \\ -\dfrac{\upsilon_n A_{n-1}^{-1}}{a_{nn} - \upsilon_n A_{n-1}^{-1} u_n} & \dfrac{1}{a_{nn} - \upsilon_n A_{n-1}^{-1} u_n} \end{pmatrix}$$

Приложения

```python
import numpy as np


def get_inv(A, depth=0):
  n = len(A)
  k = n - 1

  if n == 1:
    return np.matrix([[1 / A[0, 0]]])

  Ap = A[:k, :k]
  V, U = A[k, :k], A[:k, k]

  Ap_inv = get_inv(Ap, depth + 1)

  alpha = 1 / (A[k, k] - V * Ap_inv * U).item()
  Q = -V * Ap_inv * alpha
  P = Ap_inv - Ap_inv * U * Q
  R = - Ap_inv * U * alpha

  A_inv = np.matrix([[0.0] * n for _ in range(n)])
  A_inv[:k, :k] = P
  A_inv[k, :k] = Q[0]

  A_inv[:k, k] = R[:, 0]
  A_inv[k, k] = alpha

  return A_inv


  def main():
  mat = np.matrix([
    [1, 2, 4],
    [3, 3, 2],
    [4, 1, 3]
```

```
36   ])
37
38   print(get_inv(mat))
39
40
41 if __name__ == '__main__':
42   main()
43
```

Листинг 1: Компьютерная реализация алгоритма

Вывод

В данной лабораторной работе была произведена реализация вычисления обратной матрицы методом окаймления.