Лабораторная работа No2

«Реализация метода Гаусса»

Цель работы

Реализовать простейший метод Гаусса и научиться оценивать погрешности решения системы уравнения для матриц произвольной размерности.

Задание

• Реализовать метод Гаусса для действительных квадратных матриц произвольной размерности n. Это в дальнейшем потребуется для проведения оценок скорости алгоритма и его точности. • Реализовать возможность ручного ввода элементов матрицы произвольной размерности. • Реализовать возможность генерации матриц со случайными элементами в заданном диапазоне [-a, b], где a и b принадлежат R. При этом необходимо уметь регулировать условие диагонального преобладания, другими словами сделать возможность принудительно увеличивать на заданный порядок среднее значение генерируемых диагональных элементов aii матрицы A системы уравнений A · x = b. • Реализовать алгоритм тестирования задачи, который заключается в том, что мы заведомо определяем значения координат вектора x, данный вектор заведомо является решением уравнения A · x = b, вычисляем b путем прямого перемножения матрицы A на вектор x и далее производим поиск решения уравнения A · x = b методом Гаусса, получая xчисл, после чего производим сравнение полученного xчисл с заданным x, а также решением xбибл , полученным с использованием сторонней библиотеки выбранной студеном. При этом сравнение производится по евклидовой норме разности вектора (x-xчисл) и (x - xбибл).

Реализация

Были написаны собственный метод Гаусса, ручной ввод матрицы с файла, заполнение цифрами матрицы, увелечение значениний диагональных элементов матрицы, а так же алгоритм тестирования с приминением евклидовой нормы

```
import math
import random

import numpy as np

from my_matrix import MyMatrix
```

```
def gauss_method(a):
    n = len(a)
    x = [0 for i in range(n)]

    for i in range(n):
        if a[i][i] == 0.0:
            print('Divide by zero detected!')

        for j in range(i + 1, n):
            rat = a[j][i] / a[i][i]

            for k in range(n + 1):
                a[j][k] = a[j][k] - rat * a[i][k]

    x[n - 1] = a[n - 1][n] / a[n - 1][n - 1]
    for i in range(n - 2, -1, -1):
        x[i] = a[i][n]

        for j in range(i + 1, n):
            x[i] = x[i] - a[i][j] * x[j]

        x[i] = x[i] / a[i][i]

    return x


def prepare_to_gauss(a, v):
    for i in range(0, len(v)):
        a[i].append(v[i])
    return a


def read_matrix_from_file():
    f = open('matr.txt', 'r')
    a = []
    for line in f:
        a.append([float(x) for x in line.split()])
    return a


def fill_matrix(n, a, b):
    a = [[random.uniform(-a, b) for i in range(n)] for j in range(n)]
    return a


def increase_diag(a, pow):
    for i in range(0,len(a)):
        a[i][i] = abs(a[i][i]*(10**pow))
    return a


def calc_evklid_diff(x1,x2):
    x = []
```

```
    for i in range(0, len(x1)):
        x.append(abs(x1[i] - x2[i]))

    sum = 0
    for i in range(0, len(x)):
        sum += x[i]**2

    return math.sqrt(sum)


def testing_algo():
    a = MyMatrix(
        [
            [1,1,1,1],
            [2,-3,4,5],
            [3,4,5,10],
            [3, 6, 7,8]
        ]
    )
    x = [1.0, 3.0, 5.0, 7.0]
    calc_b = a.mult_on_vector(x)

    a_prepared = prepare_to_gauss(a.matrix, calc_b)
    calc_x = gauss_method(a_prepared)

    a_numpy = np.array([
            [1,1,1,1],
            [2,-3,4,5],
            [3,4,5,10],
            [3, 6, 7,8]
        ])
    x_numpy = np.array(x)
    b_numpy = np.matmul(a_numpy, x_numpy)
    x_numpy_calc = np.linalg.solve(a_numpy, b_numpy)

    evkl_diff = calc_evklid_diff(x, calc_x)
    print("Evklid difference between calculated x and given x: %.30f" % evkl_diff)

    evkl_diff_numpy = calc_evklid_diff(calc_x, x_numpy_calc)
    print("Evklid difference between calculated x and numpy x: %.30f" % evkl_diff_n


if __name__ == '__main__':
    # Реализовать метод Гаусса для действительных
    # квадратных матриц произвольной размерности n.
    print("My Gauss method: {}".format(gauss_method(
        [
            [1,1,1,9],
            [2,-3,4,13],
            [3,4,5,40]
        ]
    )))

    # Реализовать возможность ручного ввода
    # элементов матрицы произвольной размерности.
```

```
print("My read matrix: {}".format(read_matrix_from_file()))

# Реализовать возможность генерации матриц со случайными
# элементами в заданном диапазоне [-a, b], где a и b принадлежат R.
print("My random generated matrix: {}".format(fill_matrix(4, 5, 8)))
print("My diagonal increasing method: {}".format(increase_diag([[1,1,1,9],
        [2,-3,4,13],
        [3,4,5,40]],
        3)))

#Реализовать алгоритм тестирования задачи
testing_algo()
```

```
My Gauss method: [1.0, 3.0, 5.0]
My read matrix: [[1.0, 1.0, 1.0, 9.0], [2.0, -3.0, 4.0, 13.0], [3.0, 4.0, 5.0,
My random generated matrix: [[-1.1901279546076617, -4.598931464123968, 3.96221
My diagonal increasing method: [[1000, 1, 1, 9], [2, 3000, 4, 13], [3, 4, 5000
Evklid difference between calculated x and given x: 0.00000000000000015383701499
Evklid difference between calculated x and numpy x: 0.00000000000000044630413233
```