Курсовая работа учебного года 2023-2024 по курсу «Численные методы»

Выполнил студент группы М8О-408Б-20 Фаттяхетдинов С.Д.
Преподаватель: Пивоваров Д. Е.

Вариант курсовой работы: 1

Вариант 1

Решение систем линейных алгебраических уравнений с симметричными разреженными матрицами большой размерности. Метод сопряженных градиентов.

Метод решения

В данной работе мною было реализовано решение систем линейных алгебраических уравнений вида **Ax=b** с симметричными разреженными матрицами большой размерности методом сопряжённых градиентов.
Рассмотрим сам алгоритм.
Инициализация:

1) $r_0 = b - Ax_0$ – вектор ошибки
2) $d_0 = r_0$ – вектор направления поиска

Алгоритм итерации (повторять до выполнения условия остановки)

1) $alpha_i=(r_i*r_i) / (d_i * A * d_i)$
2) $x_{i+1} = x_i + alpha_i d_i$
3) $r_{i+1} = r_i - alpha_i*A*d_i$
4) $beta_i = (r_{i+1} * r_{i+1})/(r_i * r_i)$
5) $d_{i+1} = r_{i+1} + beta_i d_i$
6) $i = i + 1$

Критерии остановки могут быть разными. В случае данной курсовой работы алгоритм завершает свою работу в одном из двух случаев:

1) Норма вектора ошибки меньше заданного epsilon
2) Достигнуто максимальное число итераций

О программе

Курсовая работа содержит в себе две программы:
1) generate_matrix.py – для генерации тестовых данных
2) cp.ipynb – реализация алгоритма метода сопряжённых градиентов

Листинг

generate_matrix.py

```
import argparse
from random import randint

import numpy as np
from scipy.sparse import csc_matrix, rand


def main():
    parser = argparse.ArgumentParser()

    output = "matrix2.txt"
    shape = int(input())
    if shape < 3:
        exit()

    matrix = rand(shape, shape, density=0.4,
random_state=randint(112, 154))
    matrix = matrix.toarray()
    for i in range(shape):
        for j in range(shape):
            matrix[j][i] = matrix[i][j]
    matrix = csc_matrix(matrix)
    with open(output, "w") as f:
        f.write(f"{shape}\n")
        for i in matrix.toarray().round(3):
            for j in i:
                f.write(f"{j} ")
            f.write("\n")
        d = np.random.randint(5, 53, shape)
        for i in d:
            f.write(f"{i} ")
        f.write("\n")


if __name__ == "__main__":
    main()
```

cp.ipynb

```
from loguru import logger
import numpy as np
from numpy.linalg import norm
from scipy.sparse import diags, csc_matrix
from time import time
import sys

logger_handlers = [
    {
        "sink": sys.stdout,
        "level": "INFO",
        "format": "<level>level={level}
{message}</level>",
    }
]
logger.configure(handlers=logger_handlers)
args = {}
args["input"] = "matrix2.txt"
args["output"] = "output.txt"
args["eps"] = 0.01
args["diag"] = False

def read_matrix(filename: str):
    with open(filename) as f:
        shape = int(f.readline())
        matrix = [[float(num) for num in line.split()]
                  for _, line in zip(range(shape), f)]
        matrix = csc_matrix(matrix)
        b = np.array([float(num) for num in
f.readline().split()])
        return matrix, b

class Solver:
    def __init__(self, matrix, b, output_file,
                 x0=None, eps=1e-5):
        self.output = 'res_default' if output_file is
None else output_file
        self.matrix = matrix
```

```
        self.b = b
        self.eps = eps
        self.shape = matrix.shape[0]
        self.x0 = np.array([0] * self.shape) if x0 is
None else x0
        self.k = 0



    def solve(self, max_iter=100000):
        x0 = self.x0
        r0 = self.b - self.matrix.dot(x0)
        p0 = np.copy(r0)
        for _ in range(max_iter):
            temp = self.matrix @ p0
            norm_0 = np.dot(r0, r0)
            alpha_i = norm_0 / (temp @ p0)
            x_new = x0 + p0 * alpha_i
            r_new = r0 - temp * alpha_i
            norm_new = r_new @ r_new
            beta_i = norm_new/norm_0
            p_new = r_new +  p0*beta_i

            r0 = r_new
            p0 = p_new
            x0 = x_new

            self.k+=1
            if norm(r_new) < self.eps:
                break
        return x0




    def solve_and_print(self):
        start = time()
        x = self.solve()
        end = time()
        start2 = time()
```

```
            x2 = np.linalg.solve(self.matrix.toarray(),
 self.b)
            end2 = time()
            logger.info('Custom solution:\n')
            logger.info(f'{x.round(5)}\n')
            logger.info(f'eps={self.eps} shape={self.shape}
 iterations={self.k} mean={np.mean(x)} time={round(end -
 start, 5)} seconds\n')
            logger.info('NumPy solution:\n')
            logger.info(f'{x2.round(5)}\n')
           logger.info(f'mean={np.mean(x2)}
time={round(end2 - start2, 5)} seconds\n')

 def main():

     matrix, b = read_matrix(args["input"])
     solver = Solver(matrix, b,
 output_file=args["output"], eps=args["eps"])
     solver.solve_and_print()


 if __name__ == "__main__":
     main()
```

Результаты

Входные данные (размер матрицы, матрица, вектор решений):

10
0.0 0.0 0.673 0.0 0.21 1.0 0.0 0.0 0.539 0.0
0.0 0.085 0.0 0.801 0.0 0.0 0.0 0.0 0.0 0.0
0.673 0.0 0.16 0.553 0.222 0.0 0.0 0.0 0.993 0.9
0.0 0.801 0.553 0.0 0.695 0.72 0.0 0.637 0.0 0.0
0.21 0.0 0.222 0.695 0.015 0.0 0.087 0.0 0.0 0.0
1.0 0.0 0.0 0.72 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.087 0.0 0.0 0.522 0.063 0.0
0.0 0.0 0.0 0.637 0.0 0.0 0.522 0.0 0.0 0.734
0.539 0.0 0.993 0.0 0.0 0.0 0.063 0.0 0.074 0.216
0.0 0.0 0.9 0.0 0.0 0.0 0.0 0.734 0.216 0.041
36 21 19 24 41 30 27 8 28 48

Результат работы программы (для сравнения также приведено решение встроенным методом numpy):

level=INFO Custom solution:

level=INFO [0.4867 -139.21825 36.03465 40.9907 160.46605 -17.3655
23.0181 26.00615 -8.50406 -41.04431]

level=INFO eps=0.01 shape=10 iterations=10 mean=8.087021967322894
time=0.002 seconds

level=INFO NumPy solution:

level=INFO [0.4867 -139.21825 36.03465 40.9907 160.46605 -17.3655
23.0181 26.00615 -8.50406 -41.04431]

level=INFO mean=8.08702196756482 time=0.002 seconds

Вывод

Благодаря выполнению данного курсового проекта мною были приобретены новые знания в области численных методов, а именно был изучен и реализован алгоритм метода сопряжённых градиентов, а также он был проиллюстрирован на примере.