Zadanie numeryczne 5

Oleg Semenov

Wstęp

W rozwiązaniu danego zadania skorzystałem z Pythona 3.7 z dodatkowym użyciem bibliotek NumPy i SciPy a szczególnie ich funkcji numpy.zeros() dla zainicjowania początkowej macierzy i wektorów używanych do późniejszych operacji, numpy.dot() oraz scipy.sparse.linalg.cg() korzystającej z metody gradientów sprzężonych. To rozwiązanie wykorzystuje modyfikowaną wersję rozwiązania zadania drugiego, w którym korzystałem ze wzoru Shermana-Morrisona, by umożliwić operowanie na danej wejściowej macierzy.

Wzór Shermana-Morissona

The Sherman-Morrison formula is a formula that allows a perturbed matrix to be computed for a change to a given matrix A. If the change can be written in the form:

$$u \otimes v \tag{1}$$

for two vectors u and v, then the Sherman-Morrison formula is

$$(A + u \otimes v)^{-1} = A^{-1} - \frac{(A^{-1}u) \otimes (v \cdot A^{-1})}{1 + \lambda} \tag{2}$$

where

$$\lambda \equiv v \cdot A^{-1}u \tag{3}$$

```
import numpy as np
import scipy.sparse.linalg as la
def sherman_morrison(mat, sol):
    u = np.ones(64)
    v = np.ones(64)

    for i in range(64):
        for j in range(64):
            mat[i][j] -= 1

    y = np.array(la.cg(mat, sol)[0])
    z = np.array(la.cg(mat, u)[0])
    x = y - (np.dot(np.array(v), y)) / (1 + np.dot(np.array(v), z)) * z
    return x
```

To już jest zmodyfikowana wersja, z użyciem metody gradientu sprzężonego.

Metoda gradientu sprzężonego

Tutaj używam metody scipy.sparse.linalg.cg(). Metoda gradientu sprzężonego to algorytm pozwalający rozwiązać układy równań przedstawione za pomocą dodatnio określonej symetrycznej macierzy. Jest to metoda iteracyjna.

```
import scipy.sparse.linalg as la
y = np.array(la.cg(mat, sol)[0])
z = np.array(la.cg(mat, u)[0])
```

Wykorzystuję tą metodę do dostosowania wzoru Shermana-Morissona dla tego zadania.

Przykładowa implementacja metody w Pythonie:

```
import numpy as np
def conjgrad(A, b, x):
    """
    A : matrix
        A real symmetric positive definite matrix.
    b : vector
        The right hand side (RHS) vector of the system.
    x : vector
        The starting guess for the solution.
    """
    r = b - np.dot(A, x)
    p = r
    rsold = np.dot(np.transpose(r), r)

    for i in range(len(b)):
        Ap = np.dot(A, p)
        alpha = rsold / np.dot(np.transpose(p), Ap)
        x = x + np.dot(alpha, p)
        r = r - np.dot(alpha, Ap)
        rsnew = np.dot(np.transpose(r), r)
        if np.sqrt(rsnew) < 1e-8:
            break
        p = r + (rsnew/rsold)*p
        rsold = rsnew
    return x
```

Użyłem metody z biblioteki SciPy dla lepszej czytelności kodu.

Inicjowanie macierzy

```
import numpy as np

matrix = np.ones((64, 64))

for i in range(64):
    matrix[i][i] = 5
for i in range(63):
    matrix[i + 1][i] = 2
    matrix[i][i + 1] = 2
for i in range(60):
    matrix[i][i + 4] = 2
    matrix[i + 4][i] = 2

b = np.ones(64)
```

Main i Wyniki

```
if __name__ == '__main__':
    with open("out.txt", 'w') as fd:
        res = sherman_morrison(matrix, b)
        for i in range(len(res)):
            fd.write("x{0} = {1:.16f}\n".format(i, res[i]))
```

Wyniki są zapisywane w pliku out.txt

```
x0 = 0.0211006681052197    x32 = 0.0135927644456062
x1 = 0.0142203599652245    x33 = 0.0135346998502064
x2 = 0.0159436143582596    x34 = 0.0136253188830662
x3 = 0.0176290223618382    x35 = 0.0135449149969627
x4 = 0.0099880273170677    x36 = 0.0135684458625622
x5 = 0.0146851656500665    x37 = 0.0136296280612656
x6 = 0.0129876086654239    x38 = 0.0134953652742372
x7 = 0.0121641557243757    x39 = 0.0136454357853093
x8 = 0.0152442263556428    x40 = 0.0135680601975721
x9 = 0.0126742517153858    x41 = 0.0134948299611228
x10 = 0.0138681388421481   x42 = 0.0137296785663688
x11 = 0.0140945085707096   x43 = 0.0134196221072796
x12 = 0.0128076267715461   x44 = 0.0136312329120274
x13 = 0.0141159291012948   x45 = 0.0137050232280658
x14 = 0.0133820305915898   x46 = 0.0132832294271319
x15 = 0.0133941351606664   x47 = 0.0139266469194198
x16 = 0.0139266469194198   x48 = 0.0133941351606664
x17 = 0.0132832294271319   x49 = 0.0133820305915898
x18 = 0.0137050232280658   x50 = 0.0141159291012948
x19 = 0.0136312329120274   x51 = 0.0128076267715461
x20 = 0.0134196221072796   x52 = 0.0140945085707096
x21 = 0.0137296785663688   x53 = 0.0138681388421481
x22 = 0.0134948299611228   x54 = 0.0126742517153858
x23 = 0.0135680601975721   x55 = 0.0152442263556428
x24 = 0.0136454357853093   x56 = 0.0121641557243757
x25 = 0.0134953652742372   x57 = 0.0129876086654239
x26 = 0.0136296280612656   x58 = 0.0146851656500665
x27 = 0.0135684458625622   x59 = 0.0099880273170677
x28 = 0.0135449149969627   x60 = 0.0176290223618382
x29 = 0.0136253188830662   x61 = 0.0159436143582596
x30 = 0.0135346998502064   x62 = 0.0142203599652245
x31 = 0.0135927644456062   x63 = 0.0211006681052197
```