Zadanie 2 – dokumentacja

2. (Obowiązkowe) Rozwiązać układ równań

$$\begin{bmatrix} 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 5 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 8 & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

Wskazówka: wykorzystać wzór Shermana–Morissona pozwalający zapisać macierz główną układu równań w formie $A_1 = A + uv^T$, gdzie A to macierz trójdiagonalna

Rozwiązanie powinno zawierać:

- opis algorytmu wykorzystującego wzór Shermana–Morissona, ze wskazaniem metod użytych do rozwiązywania kolejnych układów równań pojawiających się w problemie
- kod programu i wyniki numeryczne
- komentarz dotyczący złożoności obliczeniowej i pamięciowej wykorzystanego algorytmu

Do rozwiązania tego równania użyłem algorytmu Shermanna-Morrisona dostępnego który służy do rozwiązania równania **A^-1*b=z** z wykorzystaniem wzoru Shermanna-Morrisona służącego do obliczenia odwrotności sumy macierzy odwracalnej **A1** oraz **iloczynu diadycznego uv^T** wektorów **u** i **v.** Po otrzymaniu macierzy A=A1-uv dostajemy dwa równania które rozwiązuję algorytmem Thomasa będący idealnym algorytmem do obliczania równań z macierzą trójdiagonalną. Po otrzymaniu wektorów z oraz q wstawiamy to do ostatniego równania który oblicza nam wyniki. Wyniki wypisuje w pętli.

```
# -*- coding: cp1250 -*-
import numpy as np
import copy

#Tworzymy macierz
A1 = np.diag([2.0,1.0,1.0,2.0,1.0,2.0,1.0,2.0],-1)\
    +np.diag([4.0,8.0,4.0,3.0,4.0,5.0,3.0,8.0,5.0])\
    +np.diag([2.0,1.0,1.0,2.0,1.0,2.0,1.0,2.0],1)
A1[0][8] =1; A1[8][0]=1
b = np.array([1.,2.,3.,4.,5.,6.,7.,8.,9.]) #wektor b
u = np.array([1.,0.,0.,0.,0.,0.,0.,0.,1.]) #wektor u
u_v = np.zeros(shape=[9,9])
u_v[0][0]=1
u_v[8][8]=1
#print(u*v)
A = A1 - u_v #tworzymy macierz tródiagonalną
#Korzystam z algorytmu Shermanna-Morrisona z wykładu prof.Góry
#Aby obliczyć ten układ równań muszę rozwiązać dwukrtnie równanie
#z macierzą trójdiagonalną, do tego celu używam algorytmu Thomasa
#ze złożonością O(N)
#Algorytm Thomasa - faktoryzacja LU
L=np.diag([1.0]*9,0)
U=np.zeros(shape=[9,9])
U[0][0]=A[0][0] #gdyż f1 = 1*f1'
#Dekompozycja LU
for i in range(0,8):
    U[i][i+1]=A[i][i+1]
    L[i+1][i] = (A[i+1][i]/U[i][i])
    U[i+1][i+1]=A[i+1][i+1]-L[i+1][i]*U[i][i+1]
```

```
#Tworzę kopię dwóch wektorów do obliczeń
b_temp=copy.copy(b)
u_temp=copy.copy(u)
#Tworzę wektor z i q
z = [0.,0.,0.,0.,0.,0.,0.,0.,0.]
q = [0.,0.,0.,0.,0.,0.,0.,0.,0.]

#Przeprowadzam forward-substitution
for x in range(1,9):
    b_temp[x] = b[x]-((L[x][x-1])*(b_temp[x-1]))
    u_temp[x] = u[x]-((L[x][x-1])*(u_temp[x-1]))

#Przeprowadzam back-substitution
z[8]=b_temp[8]/U[8][8]
q[8]=u_temp[8]/U[8][8]
for i in range(7,-1,-1):
    z[i]=(b_temp[i]-((A[i][i+1])*(z[i+1])))/U[i][i]
    q[i]=(u_temp[i]-((A[i][i+1])*(q[i+1])))/U[i][i]

#Tworzę wektor x z wynikami
Results = np.array([0.,0.,0.,0.,0.,0.,0.,0.,0.,])
#Rozwiązuje równanie z podpunktu 3 z wykładu
for i in range(9):
    Results[i] = z[i]-(np.dot(u,z)*q[i])/((1+np.dot(u,q)))
for i in range(9):
    print("x%d"%(i+1)+" = %f"%Results[i])
```

Wyniki:

```
x1 = -0.315596
x2 = 0.259203
x3 = 0.557569
x4 = 0.510520
x5 = 0.955436
x6 = 0.157218
x7 = 2.129238
x8 = 0.297850
x9 = 1.743979
```

Złożoność obliczeniowa:

Na złożoność obliczeniową i pamięciową tego rozwiązania składa się tylko koszt algorytmu Thomasa który wynosi **O(N).**