SPRAWOZDANIE NUM4

1.Wprowadzenie:

Celem ćwiczenia jest rozwiązanie w jak najszybszym czasie Ay=b, dla macierzy A która posiada

2.Uruchomeinie programu:

Po otwarciu katalogu i przejściu do projektu mamy do wyboru 3 możliwości, dla jakich są generowane wykresy:

- Uzyskanie wyniku
  python main.py 1
- Wykres zależności czasu od pracy własnego algorytmu
  python main.py 2
- Wykres zależności czasu do pracy własnego algorytmu oraz wykonania przez bibliotekę Numpy
- **python main.py 3**

Należy zamknąć okienko z wykresem aby móc uruchomić kolejne wykresy

3.Omówienie problemu:

A. Analiza macierzy

Macierz A, która mamy podana podobnie jak w zadaniu NUM3 nie jest macierzą gęstą. Jesteśmy więc w stanie zoptymalizować jej wyliczanie, uzyskując lepszą złożoność obliczeniową niż standardowymi metodami.

B. Rozkład macierzy A

Macierz A jest macierzą rzadką, ale dla lepszego zoptymalizowania, skrócenia wielu obliczeń, warto postarać się aby poza diagonalą oraz ”wstęgą” ósemek znajdujących się nad diagona, inne elementy były równe zeru. Ku temu rozbijemy macierz A na sume dwóch macierzy A=B+ uv^T:

$$\begin{pmatrix} 12 & 8 & \cdots & & 1 \\ \vdots & & \ddots & & \vdots \\ 1 & & \cdots & 12 & 8 \end{pmatrix} = \begin{pmatrix} 11 & 7 & \cdots & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & \cdots & 11 & 7 \end{pmatrix} + \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

Dzięki temu rozkładowi możemy przechowywać naszą macierz A jako macierz wtęgową, tak jak w NUM3. Naszą macierz składającą się z jedynek możemy przedstawić jako iloczyn wektorów: u=[1,1,1,1,..n],v^T=[1,1,1,1,…n]

4.Sposób na rozwiązanie

Nasze macierz A więc wygląda następująco: A = B + uv^T

Zapis naszej macierzy może więc wyglądać następująco:

(B + uv^T)y=b,

z racji, że y jest naszą niewiadomą przekształcamy na:

y=(B + uv^T)$^{-1}$b, zapisanie tu macierzy odwrotnej nie jest przypadkowe, możemy bowiem skorzystać wzorem Shermana Morrisona:

$$(B + uv^T)^{-1} = B^{-1} - \frac{B^{-1}uv^TB^{-1}}{1+v^TB^{-1}u}$$

Nasze równanie wygląda: Ay= b ⇔ y = $B^{-1}b - \frac{B^{-1}uv^TB^{-1}b}{1+v^TB^{-1}u}$

Stwórzmy więc wektor z = $B^{-1}b$ oraz z'=$B^{-1}u$ i nasze równanie będzie w postaci $z - \frac{z\prime v^T z}{1+v^T z\prime}$ gdzie $z'v^Tz$ jest suma wszytskich wartości wektora z, a $1 + v^Tz'$ suma wszytskich wartości wektora z' plus 1.

Musimy więc rozwiązać układ równań 2x2:

1. *z = $B^{-1}b$ ⇔ Bz=b*
2. z'=$B^{-1}u$ ⇔ *Bz'=u*

Świetna tu zastosowania będzie metoda backward substitution, bowiem macierz B jest macierzą trójkątną, czyli nie potrzebuje ona rozkładu.

W naszym przypadku gdzie przez B_0 rozumiemy główną diagonale a przez B_1 to diagonala nad diagonalą głowną:

- z

  \- dla n =! 79 $z_n = \frac{bn - B_1 \cdot z_{n+1}}{B_{0n}}$

  \- dla n=79 *z* = $\frac{b_{79}}{B_{079}}$

- z’

  - dla n =! 79 $z_n = \frac{1-B_{1n} \cdot z\prime_{n+1}}{B_{0n}}$

  - dla n=79 $z = \frac{1}{B_{079}}$

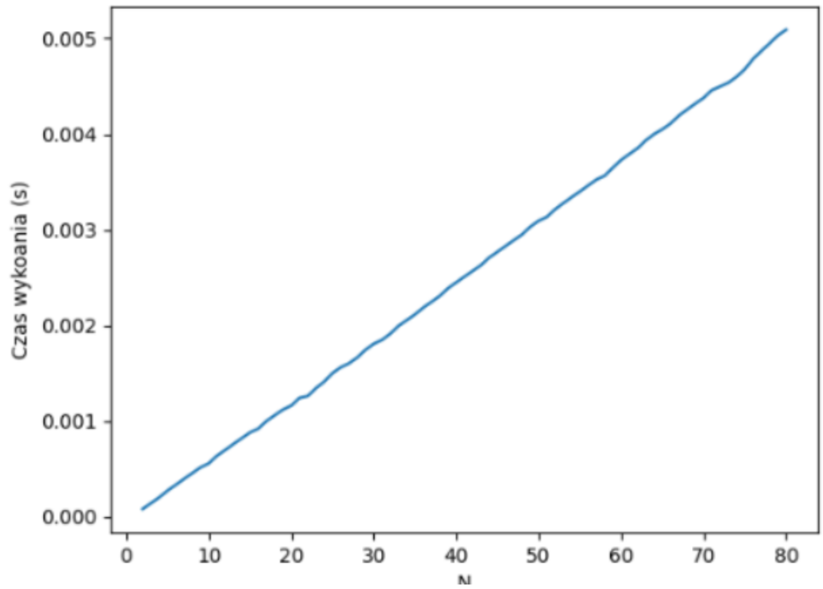
5.Wyniki

Rozwiązaniem Ay=b jest wektor:

[0.05081874647092033, 0.050818746470920384, 0.050818746470920356, 0.05081874647092041, 0.0508187464709203, 0.050818746470920495, 0.05081874647092027, 0.05081874647092055,0.050818746470919995, 0.05081874647092097, 0.05081874647091941, 0.05081874647092183, 0.05081874647091811, 0.050818746470923964, 0.050818746470914694, 0.05081874647092918, 0.05081874647090645, 0.0508187464709422, 0.05081874647088608, 0.05081874647097434, 0.05081874647083556, 0.050818746471053555, 0.050818746470711024, 0.05081874647124937, 0.05081874647040335, 0.05081874647173287, 0.05081874646964363, 0.05081874647292664, 0.050818746467767545, 0.05081874647587481, 0.05081874646313475, 0.050818746483154875, 0.05081874645169471, 0.050818746501132134, 0.05081874642344486, 0.05081874654552479, 0.050818746353684746, 0.05081874665514777, 0.05081874618142013, 0.050818746925849256, 0.050818745756032124, 0.05081874759431618, 0.050818744705584146, 0.05081874924502011, 0.05081874211162077, 0.05081875332124841, 0.05081873570611911, 0.05081876338703656, 0.0508187198884521, 0.05081878824337041, 0.05081868082849883, 0.05081884962329711, 0.050818584374328346, 0.05081900119413643, 0.050818346191580877, 0.05081937548131102, 0.05081775802602076, 0.050820299741476865, 0.05081630561718861, 0.050822582098213026, 0.050812719056603256, 0.050828218121990176, 0.05080386244781063, 0.05084213565009277, 0.05078199204650666, 0.05087650342357056, 0.0507279855453272, 0.05096137078256674, 0.050594622552618984, 0.05117094119967963, 0.05026529761144152, 0.051688451821529896, 0.049452066634248226, 0.05296638621426236, 0.04744388401709723, 0.05612210175549953, 0.04248490245229597, 0.0639147870716158, 0.030239254098398893, 0.08315794877059696]

6.Wnioski

Dzięki stworzonemu algorytmowi jesteśmy w stanie rozwiązać równanie w czasie O(n).



Rozbicie macierzy na dwie macierze, jedną składającą się z samych jedynek, druga będącą macierzą wstęgową oraz późniejsze zastosowanie wzoru Shermana-Morissona i backward substitution, przyczyniło się do tego, że w przypadku naszej macierzy wiele obliczeń się skróciło w porównaniu do algorytmu zawartego w bibliotece Numpy jak widać poniżej w uśrednionym czasie.