Grzegorz Mikołajczyk

Zadanie numeryczne 4

7. **(zadanie numeryczne NUM4)** Rozwiąż równanie macierzowe $\mathbf{Ay} = \mathbf{b}$ dla

$$\mathbf{A} = \begin{pmatrix} 10 & 8 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ 1 & 10 & 8 & 1 & \dots & 1 & 1 & 1 & 1 \\ 1 & 1 & 10 & 8 & \dots & 1 & 1 & 1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & 1 & \dots & 1 & 10 & 8 & 1 \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 & 10 & 8 \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 10 \end{pmatrix}$$

oraz $\mathbf{b} \equiv (5, \dots, 5)^T$. Macierz $\mathbf{A}$ ma liczby 10 na diagonali, 8 na pierwszej pozycji nad diagonalą, a pozostałe elementy są równe 1. Wymiar macierzy ustalamy na $N = 50$. Odpowiedni algorytm, podobnie jak dla zadania **NUM3**, należy zaimplementować samodzielnie (mile widziane jest sprawdzenie wyniku przy użyciu procedur bibliotecznych lub pakietów algebry komputerowej).

Wprowadzenie

Celem zadania jest napisanie programu rozwiązującego równanie ze skomplikowaną macierzą w bardzo dobrej złożoności obliczeniowej. W tym celu powinien zostać wykorzystany wzór Shermana-Morrisona, pozwalający rozbić obliczanie na operacje rzędu $O(n)$.

Wynik

Problem rozwiązuje program *program.py*, który nie wykorzystuje żadnych dodatkowych bibliotek, ponad bazowe możliwości pythona.

Jako że po rozbiciu zapisaniu macierzy wejściowej w formie $A + \vec{u}\vec{v}^T$, gdzie wektory $\vec{u}$ i $\vec{v}$ są równe $(1,1, \dots ,1)^T$, otrzymujemy macierz A która ma na diagonali same 9 i w pierwszym pasie nad diagonalą same 7, rozkład LU jest nie potrzebny, jako że dana macierz jest macierzą trójkątną górną, więc dla takiego A: $A = LU = U$, a więc wszystkie działania tą macierzą na wektory można od razu wykonywać poprzez „backward substitution”.

Program nie przechowuje macierzy U w pamięci, używa funkcji pomocniczej, która po prostu zwraca odpowiednią wartość macierzy U na podstawie otrzymanych indeksów. Tak samo program nie przechowuje w pamięci wektora $\vec{b}$, ani $\vec{u}$, ani $\vec{v}$, jako że wszystkie ich wartości są takie same.

Najpierw program oblicza $\vec{z}$ i $\vec{z}'$, rozwiązując równania $A\vec{z} = \vec{b}$ i $A\vec{z}' = \vec{u}$.

Następnie program oblicza elementy równania wyprowadzonego ze wzoru Shermana-Morrisona $\vec{y} = \vec{z} - \frac{\vec{z}'(\vec{v}^T\vec{z})}{1+\vec{v}^T\vec{z}'}$ co daje wynik:

$\vec{y}$ = (0.07525844089350037, 0.07525904117533852, 0.07525826938440369, 0.07525926168703423, 0.07525798586936636, 0.07525962620636797, 0.07525751720165161, 0.07526022877914404, 0.07525674246522518, 0.07526122486883524, 0.07525546177847939, 0.07526287146607977, 0.07525334472487927, 0.07526559339213706, 0.07524984510566277, 0.07527009290255826, 0.07524406002083556, 0.07527753086876468, 0.07523449692142722, 0.07528982628228975, 0.07521868853260927, 0.07531015135362706, 0.07519255629803279,

0.07534374994093965, 0.07514935811434514, 0.07539929046282381, 0.0750779488719227, 0.07549110234593845, 0.07495990502220382, 0.07564287300986267, 0.07476477131144413, 0.0758937592094108, 0.07444220334059656, 0.07630848945764337, 0.07390897873572605, 0.07699406394961972, 0.07302752581747077, 0.0781273605588052, 0.07157043017708939, 0.08000076923929544, 0.06916176187360193, 0.08309762848663654, 0.0651800856984491, 0.08821692642611872, 0.058598131204829124, 0.09667943934648732, 0.04771775745006959, 0.11066849131689238, 0.029731833488120224, 0.13379325069654147) ^{T}

Dyskusja wyników

W celu weryfikacji wyników napisałem dodatkowy program *test.py*, który po wygenerowaniu wejściowej macierzy A, wykorzystuje funkcję biblioteczną *numpy.linalg.solve* do wyliczenia wektora $\vec{y}$:

```
$ py program.py
[0.07525844089350037, 0.07525904117533852, 0.07525826938440369, 0.07525926168703423, 0.07525798586936636, 0.07525962620636797, 0.07525751720165161, 0.07526022877914404, 0.07525674246522518, 0.07526122486883524, 0.07525546177847939, 0.07526287146607977, 0.07525334472487927, 0.07526559339213706, 0.07524984510566277, 0.07527009290255826, 0.07524406002083556, 0.07527753086876468, 0.07523449692142722, 0.07528982628228975, 0.07521860853260927, 0.07531015135362706, 0.07519255629803279, 0.07534374994093965, 0.07514935811434514, 0.07539929046282381, 0.0750779488719227, 0.07549110234593845, 0.07495990502220382, 0.07564287300986267, 0.07476477131144413, 0.0758937592094108, 0.07444220334059656, 0.07630848945764337, 0.07390897873572605, 0.07699406394961972, 0.07302752581747077, 0.0781273605588052, 0.07157043017708939, 0.08000076923929544, 0.06916176187360193, 0.08309762848663654, 0.0651800856984491, 0.08821692642611872, 0.058598131204829124, 0.09667943934648732, 0.04771775745006959, 0.11066849131689238, 0.029731833488120224, 0.13379325069654147]
time: 0.0010001659393310547
```

```
$ py test.py
[0.07525844 0.07525904 0.07525827 0.07525926 0.07525799 0.07525963
 0.07525752 0.07526023 0.07525674 0.07526122 0.07525546 0.07526287
 0.07525334 0.07526559 0.07524985 0.07527009 0.07524406 0.07527753
 0.0752345  0.07528983 0.07521869 0.07531015 0.07519256 0.07534375
 0.07514936 0.07539929 0.07507795 0.0754911  0.07495991 0.07564287
 0.07476477 0.07589376 0.0744422  0.07630849 0.07390898 0.07699406
 0.07302753 0.07812736 0.07157043 0.08000077 0.06916176 0.08309763
 0.06518009 0.08821693 0.05859813 0.09667944 0.04771776 0.11066849
 0.02973183 0.13379325]
time: 0.005000114440917969
```

Mimo niższej precyzji rozwiązania bibliotekowego widać, że wyniki są zgodne.

Oprócz tego oba programy mierzą swój czas wykonania, dzięki czemu widać, że *program.py* działa około 5 razy szybciej.