

Kacper Kula

Zadanie numeryczne nr 3

Omówienie

Problem polega na rozwiązaniu układu macierzowego $y = A^{-1}x$, wybierając odpowiednią, czyli najbardziej optymalną metodę, wykorzystując przy tym charakterystyczną budowę macierzy.

Macierz A jest zdefiniowana następująco:

$$A = \begin{pmatrix} 1.2 & \frac{0.1}{1} & \frac{0.15}{1^2} & & & & & & \\ 0.2 & 1.2 & \frac{0.1}{2} & \frac{0.15}{2^2} & & & & & \\ & 0.2 & 1.2 & \frac{0.1}{3} & \frac{0.15}{3^2} & & & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \\ & & & & & 0.2 & 1.2 & \frac{0.1}{N-2} & \frac{0.15}{(N-2)^2} \\ & & & & & & 0.2 & 1.2 & \frac{0.1}{N-1} \\ & & & & & & & 0.2 & 1.2 \end{pmatrix}$$

Wektor x zdefiniowany jest następująco:

$$x = (1, 2, 3, \dots, N)^T$$

Ponadto, należy rozwiązać wyznacznik macierzy A oraz stworzyć wykres zależności czasu działania utworzonego programu od parametru N.

Narzędziem do napisania programu rozwiązującego zadanie jest język Python oraz biblioteka pythona służąca do tworzenia wykresów – matplotlib.

Do sprawdzenia poprawności programu wykorzystana została biblioteka z zakresu algebry liniowej - numpy.

Rozwiązanie układu macierzowego

Macierze, w których liczba miejsc niezerowych rośnie wolniej niż wymiar macierzy, nazywamy macierzami rzadkimi.

Wykorzystując odpowiednie algorytmy do operowania na macierzach rzadkich, możemy zaoszczędzić czas wykonania i pamięć potrzebną na obliczenia.

Macierz A jest typem macierzy wstęgowej lub pasmowej, czyli taką, której wszystkie elementy są zerami oprócz jej diagonal i elementów wokół niej.

Aby efektywnie obliczyć równanie $y = A^{-1}x$ należy:

- (1) Zastosować rozkład LU dla macierzy A
- (2) Rozwiązać układ przy pomocy metody „back substitution”

Dla ogólnego przypadku złożoność czasowa (1) wynosi $O(n^3)$, a dla punktu (2) wynosi $O(n^2)$, czyli całkowita złożoność czasowa to $O(n^3) + O(n^2) = O(n^3)$.

Wykorzystując jednak strukturę macierzy A i stosując odpowiednią metodę, która pomija obliczenia dla elementów zerowych, dla (1), jak i (2) możemy osiągnąć złożoność czasową $O(n)$.

Rozkład macierzy A do postaci LU

Wzory ogólne na poszczególne elementy macierzy rozkładu przedstawiają się następująco :

dla $i \in \{1, 2, \dots, N\}$

- $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} * u_{kj}, j \in \{i, i+1, \dots, N\}$
- $l_{ij} = \frac{1}{u_{ii}} * (a_{ji} - \sum_{k=1}^{i-1} l_{jk} * u_{ki}), j \in \{i+1, i+2, \dots, N\}$

Z budowy macierzy A wynika jednak, że jedyne niezerowe elementy mają postać:

dla $i \in \{1, 2, \dots, N\}$

$u_{ii}, u_{ii+1}, u_{ii+2}, l_{i+1i}, l_{ii} = 1$

Możemy więc wyprowadzić następujące wzory:

- $u_{ii} = a_{ii} - l_{ii-1} * u_{i-1i}$
- $l_{i+1i} = \frac{1}{u_{ii}} (a_{i+1i})$
- $u_{ii+1} = a_{ii+1} - l_{ii-1} * u_{i-1i+1}$
- $u_{ii+2} = a_{ii+2}$

Do rozkładu LU zawsze będziemy N-razy wykonywać 4 operację, co daje nam liniową złożoność problemu.

Ponadto, wiedząc, że pod diagonalą znajdują się tylko jedna „wstążka”, a nad diagonalą tylko dwie „wstążki”, jedyna potrzebna nam pamięć dla elementów macierzy L i U to 4 tablice o długości N.

Rozwiązanie układu $y = A^{-1}x$ metodą „back substitution”

Aby efektywnie rozwiązać układ z zadania, należy wykorzystać wcześniej wyliczoną postać LU.

Postawiając do wzoru $Ay = x$ otrzymujemy: $LUy = x$, podstawiając $Uy = z$, tworzymy układ równań:

- $Lz = x$
- $Uy = z$

Dla ogólnego przypadku wzór na składowe wektora wynikowego to:

$$x_n = \frac{b_{nn} - \sum_{i=1}^{n-1} l_{ni} * x_i}{l_{nn}}$$

Wiedząc jednak, które elementy są niezerowe, dla każdej składowej x_1, x_2, \dots, x_n wykonamy tylko skończoną liczbę operacji, zamiast n liczby operacji, co prowadzi do osiągnięcia złożoności czasowej $O(n)$.

Obliczanie wyznacznika macierzy A

Aby z łatwością obliczyć wyznacznik macierzy A, możemy wykorzystać wcześniej wyliczoną postać LU tej macierzy.

Z własności wyznacznika: $\det(A) = \det(LU) = \det(L) * \det(U)$.

Wiemy również, że macierz L jest macierzą trójkątną dolną, a macierz U macierzą trójkątną górną.

Wyznacznik macierzy trójkątnych obliczamy z wzoru: $\det(B) = \prod_{i=1}^n b_{ii}$.

Jednak wiemy, że elementy diagonalne macierzy L wynoszą 1, zatem $\det(L) = 1$.

Podsumowując: $\det(A) = \det(LU) = \det(L) * \det(U) = 1 * \det(U) = \prod_{i=1}^n u_{ii}$

Porównanie i sprawdzenie wyników

Przy pomocy funkcji `matmul()` z biblioteki `numpy` wymnożyłem macierze `L` i `U`, a następnie porównałem wszystkie elementy powstałej macierzy z elementami oryginalnej macierzy `A`, z dokładności do trzech miejsc po przecinku przy pomocy funkcji `allclose()`.

Macierze okazały się równe.

Wyznacznik macierzy `A` obliczony przy pomocy funkcji `linalg.get()` z biblioteki `numpy` wynosi:

$$\det(A) = 6141973498.857896$$

Wyznacznik macierzy `A` obliczony przy pomocy opisanej metody wynosi:

$$\det(A) = 6142221870.809967$$

Przy pomocy funkcji `linalg.solve()` z biblioteki `numpy` obliczyłem układ macierzowy $y = A^{-1}x$, a następnie porównałem wynik z wektorem, który otrzymałem stosując opisane metody, z dokładnością do trzech miejsc po przecinku.

Wektory okazały się równe.

Czas działania programu

Średni czas wykonania programu dla $N = 124$ wyniósł: 0.1215 milisekundy. Przeprowadzone zostało 100 testów i wyciągnięta średnia czasu.

Program wykonałem również dla $N = 10, 200, 300, \dots, 10000$.

Czas liczony był tylko przy operacjach rozkładu macierzy do postaci LU i rozwiązywania układu macierzowego metodą „back substitution”.

Dla otrzymanych rezultatów sporządziłem wykres zależności czasu działania programu od parametru N .

Spodziewaliśmy się zależności czasowej $O(n)$ - tę zależność potwierdza nam widziana linia prosta na wykresie, po której rozkładają się wyniki poszczególnych testów.

