Metody Obliczeniowe w Nauce i Technice

Maciej Grzybacz

05.05 .2024

Treści zadan

1. Napisz program, który:

   (a) Jako parametr pobiera rozmiar układu równań n i ilość rozwiązań $b \in \mathbb{R}^n$.
   (b) Generuje macierz układu $A \in \mathbb{R}^{n \times n}$ i wektor wyrazów wolnych $b \in \mathbb{R}^n$
   (c) Rozwiązuje układ równań $Ax = b$ na trzy sposoby:
   i. przez dekompozycję LU macierzy $A : A = LU$;
   ii. przez odwrócenie macierzy $A : x = A^{-1}b$ i sprawdzić czy $AA^{-1} = I$ oraz $A^{-1}A = I$ (macierz jednostkowa);
   iii. przez dekompozycję QR macierzy $A : A = QR$.
   (d) Sprawdzić poprawność rozwiązania (tj., czy $Ax = b$).
   (e) Zmierzyć całkowity czas rozwiązania układu.
   (f) Porównać czasy z trzech sposobów: przez dekompozycję LU, przez odwrócenie macierzy i przez dekompozycję QR.

2. Zadanie domowe: Narysuj wykres zależności całkowitego czasu rozwiązania układu (LU, QR, odwrócenie macierzy) od rozmiaru układu równań. Wykonaj pomiary dla 5 wartości n z przedziału od 10 do 100 . Uwaga: można się posłużyć funkcjami z biblioteki numerycznej danego języka programowania.

Zadanie 1A

Program napisany w Pythonie generuje losową macierz układu $A \in \mathbb{R}^{n \times n}$ oraz wektor wyrazów wolnych $b \in \mathbb{R}^n$. Następnie rozwiązuje układ równań $Ax = b$ za pomocą dekompozycji LU.

Dekompozycja LU dzieli macierz A na iloczyn dwóch macierzy trójkątnych: dolnej (L) i górnej (U). Rozwiązanie uzyskuje się poprzez rozwiązanie dwóch układów równań:

$$Ly = b$$
$$Ux = y$$

Program używa funkcji lu_factor i lu_solve z biblioteki scipy.linalg.

Kod programu w Pythonie

```
import numpy as np
from scipy.linalg import lu, lu_solve, lu_factor
def solve_with_lu(n):
    # Generowanie losowej macierzy A i wektora b
    A = np.random.randn(n, n)
    b = np.random.randn(n)
    # Dekompozycja LU
    lu_piv = lu_factor(A)
    x = lu_solve(lu_piv, b)
    return A, b, x
```

Listing 1: Rozwiązywanie układu równań za pomocą dekompozycji LU

Zadanie 1B

Program napisany w Pythonie generuje losową macierz układu $A \in \mathbb{R}^{n \times n}$ oraz wektor wyrazów wolnych $b \in \mathbb{R}^n$. Następnie rozwiązuje układ równań $Ax = b$ poprzez odwrócenie macierzy.

Rozwiązanie układu równań uzyskuje się poprzez obliczenie odwrotności macierzy A oraz pomnożenie jej przez wektor b :

$$x = A^{-1}b$$

Program sprawdza również, czy $AA^{-1} = I$ oraz $A^{-1}A = I$, gdzie I jest macierzą jednostkowa.

Kod programu w Pythonie

```
import numpy as np
def solve_with_inverse(n):
    # Generowanie losowej macierzy A i wektora b
    A = np.random.randn(n, n)
    b = np.random.randn(n)
    # Odwracanie macierzy A
    A_inv = np.linalg.inv(A)
    x = np.dot(A_inv, b)
    # Sprawdzenie czy A * A_inv = I i A_inv * A = I
    identity_1 = np.allclose(np.dot(A, A_inv), np.eye(n))
    identity_2 = np.allclose(np.dot(A_inv, A), np.eye(n))
    return A, b, x, identity_1, identity_2
```

Listing 2: Rozwiązywanie układu równań poprzez odwrócenie macierzy

Zadanie 1C

Program napisany w Pythonie generuje losową macierz układu $A \in \mathbb{R}^{n \times n}$ oraz wektor wyrazów wolnych $b \in \mathbb{R}^n$. Następnie rozwiązuje układ równań $Ax = b$ poprzez dekompozycję QR.

Dekompozycja QR dzieli macierz A na iloczyn ortogonalnej macierzy Q oraz trójkątnej macierzy R :

$$A = QR$$

Rozwiązanie uzyskuje się poprzez rozwiązanie układu równań:

$$Rx = Q^T b$$

Program używa funkcji qr i solve_triangular z biblioteki scipy.linalg.

Kod programu w Pythonie

```
import numpy as np
from scipy.linalg import qr, solve_triangular
def solve_with_qr(n):
    # Generowanie losowej macierzy A i wektora b
    A = np.random.randn(n, n)
    b = np.random.randn(n)
    # Dekompozycja QR
    Q, R = qr(A)
    x = solve_triangular(R, np.dot(Q.T, b))
    return A, b, x, Q, R
```

Listing 3: Rozwiązywanie układu równań poprzez dekompozycję QR

Weryfikacja poprawności

Aby sprawdzić poprawność rozwiązania układu równań $Ax = b$, obliczamy iloczyn Ax i porównujemy go z wektorem b. Jeśli $Ax \approx b$ (z dopuszczalnym błędem numerycznym), rozwiązanie uważa się za poprawne. Dla metody odwrócenia macierzy dodatkowo weryfikujemy, czy macierz odwrotna spełnia właściwości macierzy jednostkowej: $AA^{-1} = I$ i $A^{-1}A = I$. W przypadku dekompozycji LU oraz QR sprawdzamy poprawność, obliczając odpowiednio iloczyn LU oraz QR i porównujac je z oryginalną macierzą A. W każdym przypadku wykorzystujemy funkcję np.allclose, aby uwzględnić błędy numeryczne.

Pomiary czasu wykonania

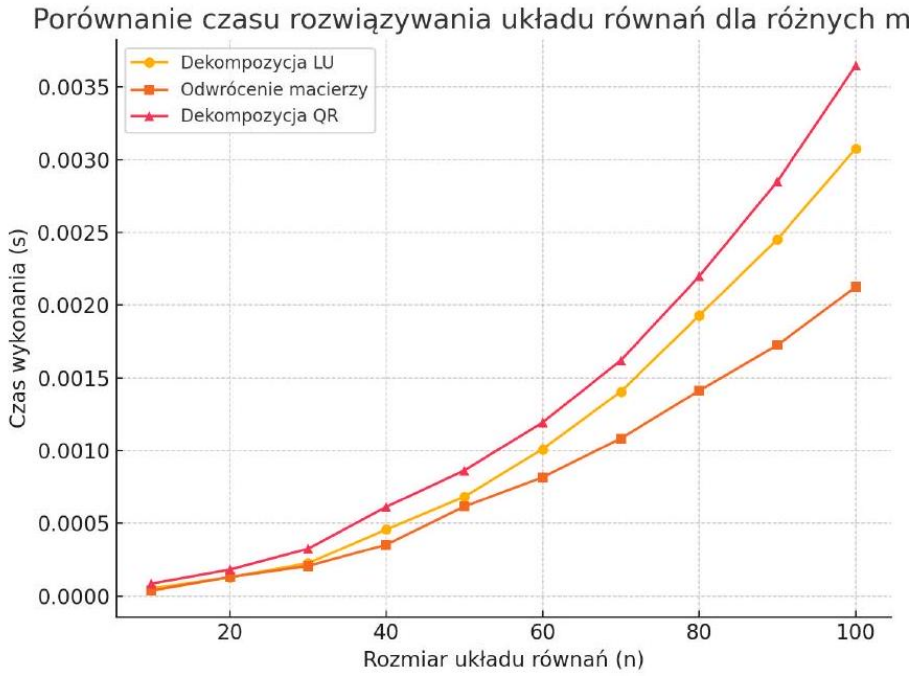
Aby ocenić efektywność różnych metod rozwiązywania układów równań $Ax = b$, zmierzyłem czas wykonania dla trzech metod:

1. Dekompozycja LU,
2. Odwrócenie macierzy A,
3. Dekompozycja QR.

Testy zostały przeprowadzone dla macierzy o rozmiarach od 10 do 100z krokiem 10. Każda metoda była testowana na losowo wygenerowanych macierzach układu A i wektorach b. Wyniki przedstawiono w tabeli 1.

Rozmiar	Czas LU (s)	Czas Inverse (s)	Czas QR (s)
10	0.000053	0.000038	0.000086
20	0.000129	0.000131	0.000182
30	0.000226	0.000207	0.000325
40	0.000457	0.000353	0.000613
50	0.000683	0.000616	0.000864
60	0.001011	0.000817	0.001194
70	0.001406	0.001084	0.001621
80	0.001929	0.001412	0.002198
90	0.002453	0.001725	0.002851
100	0.003076	0.002125	0.003647

Tabela 1: Porównanie czasu rozwiązywania układu równań dla różnych metod rozwiązania układu



Wnioski

Na podstawie wykresu można wyciągnąć następujące wnioski:

- Dekompozycja LU okazuje się być szybsza od innych metod dla mniejszych rozmiarów macierzy,
- Metoda odwrócenia macierzy A ma umiarkowany czas wykonania, ale jej efektywność spada dla większych rozmiarów macierzy,
- Dekompozycja QR wykazuje się większą stabilnością numerycznac, ale jej czas wykonania jest nieco dłuższy w porównaniu z dekompozycją LU.

1 Bibliografia

- Włodzimierz Funika Materialy ze strony internetowej
- Katarzyna Rycerz Wyklad z przedmiotu Metody Obliczeniowe w Nauce i Technice
- https://www.wolframalpha.com
- https://pl.wikipedia.org/wiki/Rozklad_QR
- https://pl.wikipedia.org/wiki/Macierz_odwrotna
- https://pl.wikipedia.org/wiki/Metoda_LU