Obliczenia naukowe - Lista 5

Jakub Jaśków 268416

8 stycznia 2024

Opis

Rozważ poniższy problem:

$$\mathbf{Ax} = \mathbf{b}$$

dla danej macierzy współczynników $\mathbf{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $\mathbf{b} \in \mathbb{R}^n, n \geqslant 4$. Macierz $\mathbf{A}$ jest macierzą rzadką i blokową o następującej strukturze:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{C}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B}_2 & \mathbf{A}_2 & \mathbf{C}_2 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_3 & \mathbf{A}_3 & \mathbf{C}_3 & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{B}_{v-2} & \mathbf{A}_{v-2} & \mathbf{C}_{v-2} & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{B}_{v-1} & \mathbf{A}_{v-1} & \mathbf{C}_{v-1} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_v & \mathbf{A}_v \end{pmatrix},$$

gdzie $v = n/l$, zakładając, że n jest podzielne przez l, gdzie l jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków): $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$. Mianowicie, $\mathbf{A}_k \in \mathbb{R}^{l \times l}$, $k = 1, \dots, v$ jest macierzą gęstą, $\mathbf{0}$ jest kwadratową macierzą zerową stopnia l, macierz $\mathbf{B}_k \in \mathbb{R}^{l \times l}$, $k = 2, \dots, v$ jest następującej postaci:

$$\mathbf{B}_k = \begin{pmatrix} 0 & \dots & 0 & b_1^k \\ 0 & \dots & 0 & b_2^k \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & b_l^k \end{pmatrix},$$

$\mathbf{B}_k$ ma tylko jedną, ostatnią, kolumnę niezerową. Natomiast macierz $\mathbf{C}_k \in \mathbb{R}^{l \times l}$, $k = 1, \dots, v-1$ jest macierzą diagonalną:

$$\mathbf{C}_k = \begin{pmatrix} c_1^k & 0 & 0 & \dots & 0 \\ 0 & c_2^k & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & c_{l-1}^k & 0 \\ 0 & \dots & 0 & 0 & c_l^k \end{pmatrix}.$$

Cel

Napisanie funkcji rozwiązującej układ $\mathbf{Ax} = \mathbf{b}$ metodą eliminacji Gaussa uwzględniając specyficzną postać macierzy $\mathbf{A}$ dla dwóch wariantów:

a) bez wyboru elementu głównego,

b) z częściowym wyborem elementu głównego.

Opis algorytmów

Eliminacja Gaussa

Eliminacja Gaussa - sprowadzenie wejściowego układu równań do macierzy schodkowej górnej wykorzystując do tego celu jedynie operacje elementarne (dodawanie, odejmowanie i mnożenie przez czynnik $l \neq 0$) na wierszach i kolumnach. Ponadto można dowolnie przestawiać wiersze i kolumny macierzy. W efekcie powyższych działań otrzymujemy macierz, za pomocą której możemy znaleźć wektor $\mathbf{x}$ zawierający rozwiązania układu równań (jeżeli $\mathbf{b}$ zostało podane).

Eliminujemy elementy niezerowe pod diagonalą macierzy za pomocą *mnożników* (l_{ij}). Krok pierwszy: eliminacja niewiadomej x_1 z $n-1$ równań odejmując dla $i = 2, \ldots, n$ odpowiednią krotności pierwszego równania od $i-$tego równania, aby wyzerować w nim współczynnik x_1. Jest to równoważne wyznaczeniu x_1 z pierwszego równania i podstawienia do pozostałych równań w układzie. Zauważmy, że w celu wyzerowania elementu a_{ik} należy od $i-$tego wiersza odjąć $k-$ty wiersz pomnożony $l_{ik} \leftarrow \frac{a_{ik}}{a_{kk}}$. Opisany powyżej algorytm nie zadziała w przypadku, gdy na diagonali macierzy w $k-$tym kroku wystąpi wartość 0. Jesteśmy sobie w stanie poradzić z taką sytuacją poprzez zamianę miejscami kolumn lub wierszy. Przy dokonywaniu eliminacji należy również dokonać zmian w wektorze prawych stron $\mathbf{b}$. W celu obliczenia wektora $\mathbf{x}$, po zakończeniu procesu eliminacji Gaussa, należy wykorzystać algorytm podstawienia wstecz.

Złożoność obliczeniowa przedstawionego powyżej algorytmu wynosi $\mathcal{O}(n^3)$.

Eliminacja Gaussa z częściowym wyborem elementu głównego

Modyfikacja algorytmu Gaussa, dzięki której można poradzić sobie z zerami występującymi na diagonali macierzy. Można zauważyć, że warunek $a_{kk} \neq 0$ nie zapewnia numerycznej stabilności algorytmu. W celu zapobiegnięcia takim sytuacjom stosuje się algorytm Gaussa rozszerzony o **wybór elementu głównego w kolumnie**. Polega on na znalezieniu w kolumnie największego (z dokładnością do wartości bezwzględnej) elementu i odpowiednim przestawieniu wierszy macierzy w taki sposób, aby wybrany element znalazł się w określonym miejscu na diagonali.
Możemy wyrazić to następującym wzorem:

$$|a_{kk}| = |a_{s(k),k}| = max\{|a_{ik}| : i = k, \ldots, n\}$$

,gdzie $s(k)$ - wektor permutacji, w którym pamiętana jest kolejność przestawień.

Dostosowanie algorytmów eliminacji Gaussa do macierzy A

$$\begin{pmatrix}
a_{11}^1 & a_{12}^1 & a_{13}^1 & a_{14}^1 & c_{11}^1 & 0 & 0 & 0 & \dots \\
a_{21}^1 & a_{22}^1 & a_{23}^1 & a_{24}^1 & 0 & c_{22}^1 & 0 & 0 & \dots \\
a_{31}^1 & a_{32}^1 & a_{33}^1 & a_{34}^1 & 0 & 0 & c_{33}^1 & 0 & \dots \\
a_{41}^1 & a_{42}^1 & a_{43}^1 & a_{44}^1 & 0 & 0 & 0 & c_{44}^1 & \dots \\
0 & 0 & 0 & b_1^2 & a_{11}^2 & a_{12}^2 & a_{13}^2 & a_{14}^2 & \dots \\
0 & 0 & 0 & b_2^2 & a_{21}^2 & a_{22}^2 & a_{23}^2 & a_{24}^2 & \dots \\
0 & 0 & 0 & b_3^2 & a_{31}^2 & a_{32}^2 & a_{33}^2 & a_{34}^2 & \dots \\
0 & 0 & 0 & b_4^2 & a_{41}^2 & a_{42}^2 & a_{43}^2 & a_{34}^2 & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

Algorytm eliminacji Gaussa

Przy takiej konstrukcji macierzy $\mathbf{A}$ w kolejnych kolumnach, w celu otrzymania górnej macierzy trójkątnej, musimy z każdej kolejnej kolumny musimy pozbyć się następującej liczby składników: $3, 2, 1, 4, 3, 2, 1, 4, \dots$. Możemy zauważyć, że dla każdej podmacierzy $\mathbf{A}_k$ eliminujemy $l-1, l-2, l-3$ współczynników a następnie całą ostatnią kolumnę podmacierzy $\mathbf{B}_k$, czyli l współczynników. Otrzymujemy zatem podany wcześniej ciąg.

Każdy wyraz ciągu możemy zapisać za pomocą wzoru

$$(l-k) \mod l$$

, gdzie k jest numerem kolejnej kolumny w macierzy $\mathbf{A}$.

```
function GAUSS(A, b, n, l)
    for k ← 1 to n do                                          ▷ Pętla (1)
        for i ← k + 1 to k + l − (k  mod l) do                 ▷ Pętla (2)
            multiplier ← A[i,k] / A[k,k]
            A[i, k] ← 0
            for j ← k + 1 to MIN(k + l, n) do                  ▷ Pętla (3)
                A[i, j] ← A[i, j] − A[k, j]
            end for
            b[i] ← b[i] − multiplier · b[k]
        end for
    end for
    x[1 : n] ← {0, . . . , 0}
    for i ← n downto 1 do
        ∑ ← 0
        for j ← i + 1 to MIN(n, i + l) do
            sum ← sum + A[i, j] ∗ x[j]
        end for
        x[i] ← (b[i] − sum) / A[i,i]
    end for
    return x
end function
```

Opis parametrów:
`A` - macierz rzadka **A**,
`b` - wektor prawych stron
`n` - rozmiar macierzy **A**
`l` - rozmiar podmacierzy

Dane wynikowe:
Wektor rozwiązań układu **x**.

Pętla (1) wykona się n razy, a pętle (2) i (3) co najwyżej l razy.
Złożoność obliczeniowa: $\mathcal{O}(n \cdot l^2) = \mathcal{O}(n)$.

Algorytm eliminacji Gaussa z częściowym wyborem elementu głównego

Algorytm eliminacji Gaussa z częściowym wyborem elementu głównego jest tożsamy z powyższym algorytmem. Różnica polega jedynie na dodatkowym czynniku - wektorze permutacji, w którym to zapamiętywane są przestawienia elementów w kolumnach.

```
function GAUSS-WITH-CHOOSE(A, b, n, l)
    perm[1 : n] ← {1, . . . , n}
    for k ← 1 to n do                                          ▷ Pętla (1)
        maxrow ← k
        maxelement ← |A[k, k]|
        for i ← k + 1 to k + l − (k mod l) do                  ▷ Pętla(2)
            if |A| > maxelement then
                maxelement ← |A[k, perm[i]]|
                maxrow = i
            end if
        end for
        SWAP(perm[k], perm[maxrow])
        for i ← k + 1 to k + l − (k mod l) do                  ▷ Pętla (3)
            multiplier ← A[perm[i],k] / A[perm[k],k]
            A[perm[i], k] ← 0
            for j ← k + 1 to MIN(k + 2 ∗ l, n) do              ▷ Pętla (4)
                A[perm[i], j] ← A[perm[i], j] − A[perm[k], j]
            end for
            b[perm[i]] ← b[perm[i]] − multiplier · b[perm[k]]
        end for
    end for
    x[1 : n] ← {0, . . . , 0}
    for i ← n downto 1 do
        ∑ ← 0
        for j ← i + 1 to MIN(n, i + l) do
            sum ← sum + A[perm[i], j] ∗ x[j]
        end for
        x[i] ← (b[perm[i]] − sum) / A[perm[i],i]
    end for
    return x
end function
```

Opis parametrów:
`A` - macierz rzadka **A**,
`b` - wektor prawych stron
`n` - rozmiar macierzy **A**
`l` - rozmiar podmacierzy

Dane wynikowe:
Wektor rozwiązań układu **x**.

Złożoność obliczeniowa: $\mathcal{O}(n)$ Złożoność obliczeniowa tego algorytmu jest nieco większa (co do stałej) niż wersji bez częściowego wybierania.

Sposób pamiętania macierzy rzadkich w języku `Julia`

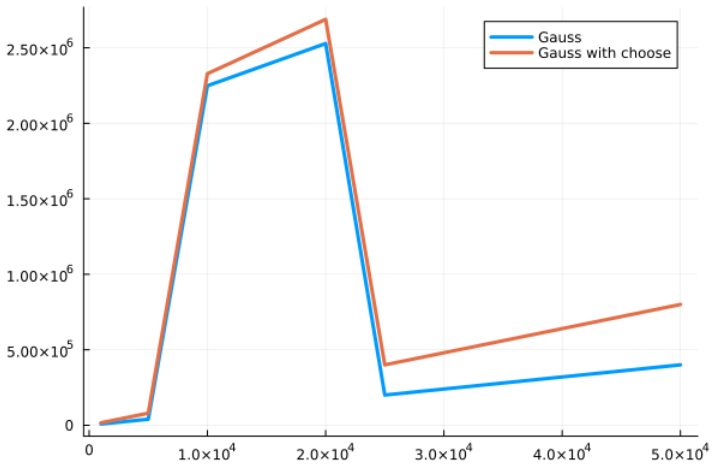
Julia udostępnia specjalną strukturę do pamiętania macierzy rzadkich - `SparseArrays`. Pamięta ona jedynie niezerowe elementy zadanej macierzy rzadkiej, co oszczędza pamięć. Sposób implementacji `SparseArrays` pozwala na szybszy dostęp do elementów macierzy poprzez odwoływanie się do nich przez kolumny zamiast przez wiersze. Dla celów badania złożoności naszych algorytmów przyjmujemy, że dostęp do elementu `SparseArrays` jest stały: $\mathcal{O}(1)$ (w rzeczywistości - $\mathcal{O}(n^2)$).

Wyniki

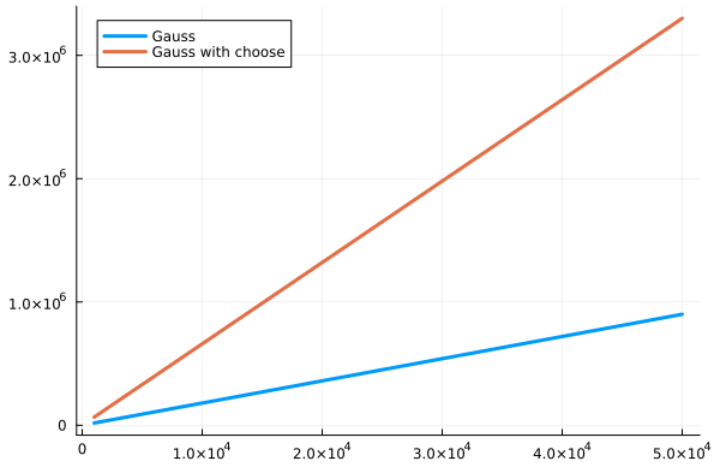
Przeprowadzone testy zaimplementowanych algorytmów miały na celu zbadanie złożoności pamięciowej oraz czasowej. Testy czasowe zostały przeprowadzone w dwóch wariantach:

- Zakładając, że czas dostępu do elementu `SparseArrays` to $\mathcal{O}(1)$,
- Za pomocą makra `@timed` zwracającego czas działania programu oraz zużytą przez niego pamięć.

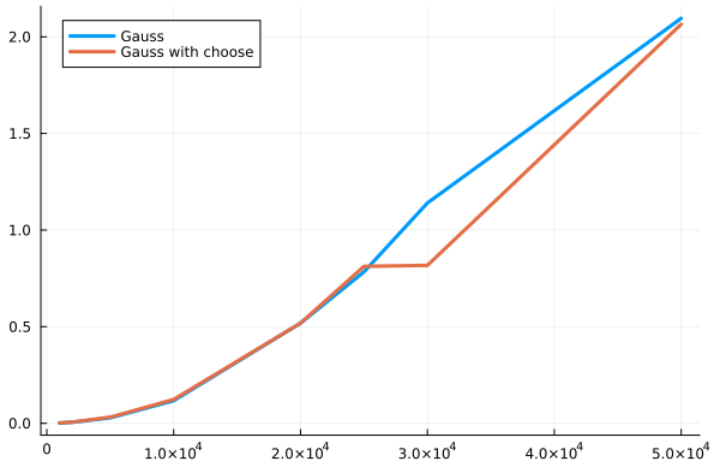
Wykresy



Rysunek 1: Wykres wykorzystanej pamięci



Rysunek 2: Wykres ilości przeprowadzonych operacji



Rysunek 3: Wykres czasu trwania

Obserwacje

- Wraz ze wzrostem rozmiaru macierzy algorytmy są bardziej efektywne pod względem ilości pamięci potrzebnej do wykonania operacji.
- Liczba iteracji dla obu metod jest proporcjonalna do rozmiaru macierzy **A**.
- Rzeczywisty czas trwania programu jest kwadratowy.

Wnioski

Na rzeczywisty czas wykonania wpłynęły odwołania do elementów `SparseArrays`, co poskutkowało kwadratowym czasem trwania testu.
W przypadku założenia, że czas odwołania się do elementów `SparseArrays` jest stały: algorytmy dobrze sprawdzają się dla macierzy rzadkich $\mathbf{A}$, oraz liczba operacji każdego z algorytmów wynosi $\mathcal{O}(n)$.
Wykres nr. 2 mówi również, że algorytm eliminacji Gaussa z częściowym wyborem elementu głównego ma o wiele większą złożoność obliczeniową niż wersja podstawowa algorytmu eliminacji Gaussa.