Obliczenia naukowe - Lista 5

Wojciech Pakulski (250350

06-01-2021

1 Opis problemu

Zadanie polegało na rozwiązaniu układu równań liniowych: $\boldsymbol{Ax} = \boldsymbol{b}$ dla danej macierzy współczynników $\mathbf{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $\mathbf{b} \in \mathbb{R}^n$

Macierz $\mathbf{A}$ jest macierzą rzadką tj. mającą dużą elementów zerowych i blokową o następującej strukturze:

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

gdzie $v = \frac{n}{\ell}$ zakładając, że n jest podzielne przez ℓ, gdzie ℓ jest rozmiarem każdej z kwadratowych macierzy wewnętrznych (bloków):

- $\boldsymbol{A}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 1, ..., v$ jest macierzą gęstą
- $\boldsymbol{B}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 2, ..., v$ jest postaci, gdzie tylko jedna, ostatnia kolumna jest niezerowa:

$$\boldsymbol{B}_k = \begin{pmatrix} 0 & \cdots & 0 & b_1^k \\ 0 & \cdots & 0 & b_2^k \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & b_\ell^k \end{pmatrix}$$

- $\boldsymbol{C}_k \in \mathbb{R}^{\ell \times \ell}$, $k = 1, ..., v-1$ jest macierzą diagonalną:

$$\boldsymbol{C}_k = \begin{pmatrix} c_1^k & 0 & 0 & \cdots & 0 \\ 0 & c_2^k & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{\ell-1}^k & 0 \\ 0 & \cdots & 0 & 0 & c_\ell^k \end{pmatrix}$$

- $\mathbf{0} \in \mathbb{R}^{\ell \times \ell}$ jest macierzą zerową stopnia ℓ.

2 Metoda eliminacji Gaussa

Podstawowa wersja algorytmu

Metoda eliminacji Gaussa jest sposobem, który jest w stanie rozwiązać zadany układ równań jak i uzyskać rozkład LU zadanej macierzy. Podstawowym założeniem jest, żeby $\det(A) \neq 0$. Algorytm metody eliminacji Gaussa można podzielić na dwa etapy

- Pierwszy etap polega na sprowadzeniu macierzy kwadratowej do macierzy górnej trójkątnej. Celem jest wyzerowanie elementów pod przekątną. Żeby wyzerować element a_{21} dzielimy go przez element przekątniowy i mnożymy pierwszy wiersz przez ten iloraz (w k-tym kroku wynosi $I_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$), następnie odejmujemy go od drugiego wiersza. Analogicznie przeprowadzamy dla dalszych wierszy, a potem dla kolejnych kolumn, aż do uzyskania macierzy górnej trójkątnej. Oczywiście modyfikując j-ty wiersz macierzy, modyfikujemy również j-ty wiersz wektora prawych stron, tak aby układy były tożsame.

- Gdy mamy wyznaczoną macierz górną trójkątną, wykonujemy podstawienie wstecz. Wykorzystujemy do tego wiersze znajdujące się poniżej do obliczenia danej wartości wektora spełniającego równanie. Rozpoczynamy od podstawienia wiersza ostatniego. Wartość i-tą wektora x można wyrazić wzorem

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}}{a_{ii}}$$

Data: A, b, n
for $k \leftarrow 1$ *to* $n-1$ **do**
  for $i \leftarrow k+1$ *to* n **do**
    $I_{ik} \leftarrow \frac{A_{ik}}{A_{kk}}$
    for $j \leftarrow k+1$ *to* n **do**
      $A_{ij} \leftarrow A_{ij} - (I_{ik} * A_{kj})$
      $b_i \leftarrow b_i - (I_{ik} * b_k)$
    end
  end
end
for $i \leftarrow n$ *downto* 1 **do**
  $x_i \leftarrow \frac{b_i - \sum_{j=i+1}^{n} A_{ij}}{A_{ii}}$
end
return $\vec{x}$

Algorithm 1: Pseudokod metody eliminacji Gaussa

Złożoność obliczeniowa tej podstawowej wersji algorytmu wynosi $O(n^3)$, co jest jednak bardzo niewydajne przy macierzach o rozmiarze rzędu klikutysięcy.

Zmodyfikowana wersja algorytmu

Jednak można zauważyć, że $\boldsymbol{A}$ jest szczególną macierzą, a mianowicie macierzą trójdiagonalną, rzadką, więc nie ma potrzeby przetrzymywać całej kwadratowej macierzy. W rozwiązaniu zastosowana będzie struktura *SparseMatrixSCS* z pakietu *SparseArrays*, dostępnego w języku *Julia*, która pamięta tylko niezerowe pola macierzy. Dodatkowo zauważmy, że ze względu na konstrukcję podmacierzy nie musimy zerować wszystkich elementów znajdujących się pod diagonalą, ponieważ część już wynosi zero.

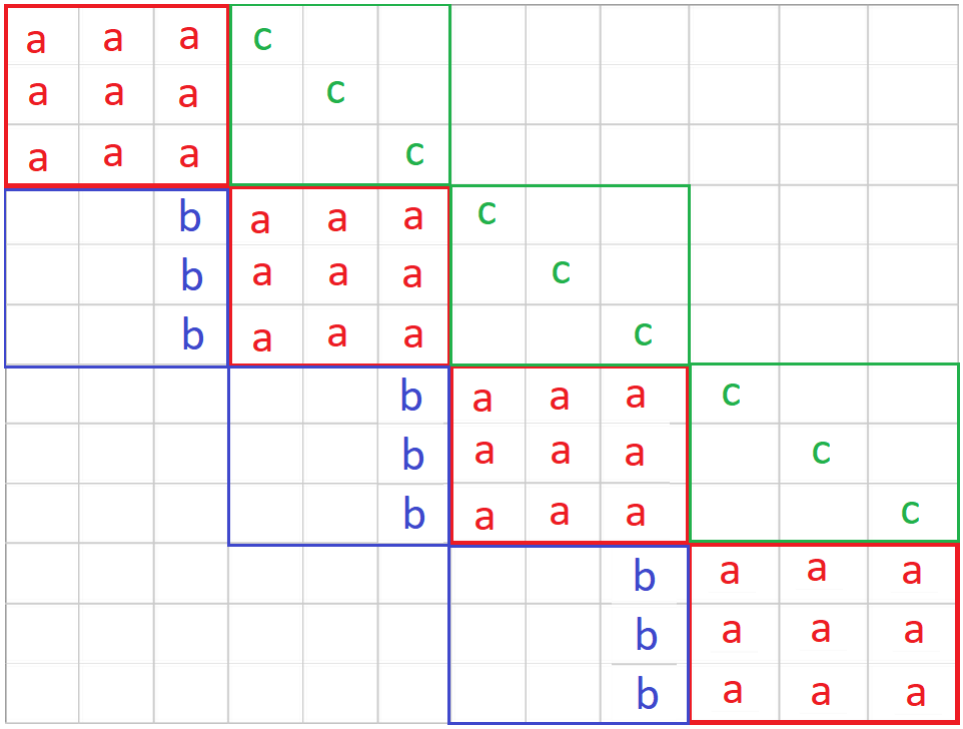
Jak można zaobserwować na przykładzie dla pierwszych l-1 kolumn, niezerowe elementy mogą znajdować się w jedynie w pierwszych l rzędach (jedynie te pochodzące z bloku A_1). Dla kolejnych ℓ kolumn, elementy niezerowe, mogą się znajdować jedynie w pierwszych 2ℓ rzędach. Podobnie dla kolejnych. Dzięki temu możemy wyliczyć, że maksymalny numer kolumny zawierający element niebędący zerem, w danym rzędzie można wyrazić zależnością:

$$ostatniaKolumna(wiersz) = \min(n, wiersz + \ell)$$

Tak samo można wyliczyć ostatni element niezerowy w wierszu:

$$ostatniWiersz(kolumna) = \min(n, \ell + \ell * \left\lfloor \frac{kolumna}{\ell} \right\rfloor)$$

Ograniczając w ten sposób liczbę iteracji po elementach macierzy można skutecznie zmniejszyć złożoność obliczeniową algorytmu.



Przykładowa macierz $\boldsymbol{A}$ dla $n = 12, \ell = 3$

Data: A, b, n, ℓ
for $k \leftarrow 1$ *to* $n-1$ **do**
  $lastColumn \leftarrow \min(n, k+\ell)$
  $lastRow \leftarrow \min(n, \ell + \ell * \lfloor \frac{k}{\ell} \rfloor)$
  for $i \leftarrow k+1$ *to* $lastRow$ **do**
    $I_{ik} \leftarrow \frac{A_{ik}}{A_{kk}}$
    for $j \leftarrow k+1$ *to* $lastColumn$ **do**
      $A_{ij} \leftarrow A_{ij} - (I_{ik} * A_{kj})$
    end
    $b_i \leftarrow b_i - (I_{ik} * b_k)$
  end
end
for $i \leftarrow n$ *downto* 1 **do**
  $sum \leftarrow 0$
  $lastColumn \leftarrow min(n, i+l)$
  for $j \leftarrow i+1$ *to* $lastColumn$ **do**
    $sum \leftarrow sum + A_{jp_i} * x_j$
    $x_i \leftarrow \frac{(b_{p_i} - sum)}{A_{ip_i} * x_j}$
  end
end
return $\vec{x}$

Algorithm 2: Pseudokod zmodyfikowanej metody eliminacji Gaussa

Z uwagi na to, że zewnętrzna pętla wykona $n-1$ obrotów, a dwie wewnętrzne po co najwyżej ℓ razy. Wyznaczenie rozwiązania (czyli ostatnia pętla) zajmie n iteracji i co najwyżej ℓ w wewnętrznej. Zakładając, że ℓ jest stałą, daje to złożoność obliczeniową równą:

$$(n-1) * \ell * \ell + n * \ell = O(n)$$

Zmodyfikowana metoda eliminacji Gaussa z częściowym wyborem elementu głównego

Poprzednia wersja algorytmu wydawała się dobra, jednak gdy w głownej diagonali pojawi się wartość zero jest problem, gdyż przeliczenie algorytmu do końca wymagałoby dzielenia przez nie. Tu przydatna jest druga wersja tego algorytmu, mianowicie metoda eliminacji Gaussa z częściowym wyborem elementu głównego. Działa ona podobnie,

jednak każdorazowo przed zerowaniem kolumny, wyszukujemy wśród elementów poniżej przekątnej współczynnika największego co do modułu i zamieniamy odpowiednie wiersze (oczywiście wraz z kolumną wyrazów wolnych). Postępujemy tak, nie tylko gdy napotkamy element zerowy na przekątnej. W ten sposób eliminujemy znika zagrożenie dzielenia przez zero, a dodatkowo otrzymane rozwiązanie jest dokładniejsze, gdyż unikamy szkodliwego ze względów numerycznych dzielenia przez małe liczby. Jednak zamiana wierszy jest bardzo kosztowna, szczególnie dla dużych macierzy, dlatego w algorytmie tworzymy nowy wektor permutacji wierszy, w którym zapisywana jest informacja, o aktualnej pozycji danego wiersza w macierzy. Taki zabieg znacznie poprawia wydajność algorytmu.

Trzeba jeszcze zwrócić uwagę, że przez zamiany wierszy, trzeba na nowo wyznaczyć wzór na maksymalny indeks kolumny. Po zastanowieniu się nad budową macierzy. Zauważmy, że przy eliminowaniu elementów w pierwszych ℓ kolumnach najdalszym indeksem zawierającym niezerowy element macierzy jest ten o numerze 2ℓ. Przy kolejnych ℓ kolumnach, ten element znajdzie się w 3ℓ kolumnie itd. Z tego można wykoncypować wzór:

$$ostatniaKolumna(wiersz) = min(n, 2l + l * \left\lfloor \frac{wiersz}{l} \right\rfloor)$$

Analizując złożoność algorytmu można zauważyć, że zewnętrzna pętla wykona się n-1 raza kolejna ℓ razy, następna ℓ razy, a wewnątrz niej $2*\ell$ i dwie ostatnie zewnętrzne pętle $n-1$ raza i $2*\ell$ iteracji wewnątrz. Zakładając, że ℓ jest stałą, daje to złożoność obliczeniową równą:

$$(n-1)(l + l * 2l) + (n-1) * 2l = O(n)$$

Data: A, b, n, ℓ
for $k \leftarrow 1$ *to* $n-1$ **do**
  $lastColumn \leftarrow 0$
  $lastRow \leftarrow 0$
  for $i \leftarrow k$ *to* $min(n, k+l)$ **do**
    if $abs(A[perm[i], k]) > lastColumn$ **then**
      $lastColumn \leftarrow abs(A[perm[i], k])$
      $lastRow \leftarrow i$
    end
  end
  $perm[lastRow], perm[k] \leftarrow perm[k], perm[lastRow]$
  for $i \leftarrow k+1$ *to* $min(n, k+l)$ **do**
    $z \leftarrow \frac{A[perm[i],k]}{A[perm[k],k]}$
    $A[perm[i], k] \leftarrow 0.0$
    for $j \leftarrow k+1$*to* $min(n, k+2*l)$ **do**
      $A[perm[i], j] \leftarrow A[perm[i], j] - z * A[perm[k], j]$
    end
    $b[perm[i]] \leftarrow b[perm[i]] - z * b[perm[k]]$
  end
end
for $k \leftarrow 1$ *to* $n-1$ **do**
  for $i \leftarrow k+1$*:*$min(n, k+2*l)$ **do**
    $b[perm[i]] \leftarrow b[perm[i]] - A[perm[i], k] * b[perm[k]]$
  end
end
for $i \leftarrow n$ *downto* 1 **do**
  $currentSum \leftarrow 0$ **for** $j \leftarrow i+1$ *to* $min(n, i+2*l)$ **do**
    $currentSum \leftarrow A[perm[i], j] * result[j]$
  end
  $result[i] \leftarrow (b[perm[i]] - currentSum)/A[perm[i], i]$
end
return result

Algorithm 3: Pseudokod zmodyfikowanej metody eliminacji Gaussa z częściowym wyborem elementu głównego

3 Podsumowanie

Wyniki i obserwacje

Poniższe testy zostały przeprowadzone na procesorze Intel Core i3-8100, o taktowaniu 3.6 GHz na systemie Windows 10.

Tabelka z zużyciem pamięci w zależności od rodzaju algorytmu eliminacji Gaussa

Rozmiar macierzy	Bez wyboru	Z wyborem	Wbudowana
16	208 bytes	624 bytes	40.594 KiB
10000	2.076 MiB	2.153 MiB	14.118 MiB
50000	390.703 KiB	1.145 MiB	70.514 MiB

Tabelka z wynikami czasu działania aglorytmów eliminacji Gaussa

Rozmiar macierzy	Bez wyboru	Z wyborem	Wbudowana
16	0.000015 seconds	0.000038 seconds	0.005684 seconds
10000	0.134000 seconds	0.156441 seconds	0.014283 seconds
50000	7.700355 seconds	7.739713 seconds	0.111017 seconds

Tabelka z błędami względnymi

Rozmiar macierzy	Bez wyboru	Z wyborem
16	5.102800490722269e-16	2.8844440295753465e-16
10000	8.252489306370072e-15	4.2158809864856956e-16
50000	1.8398915733251618e-13	3.9285991202114373e-16

Eliminacja Gaussa z bez częściowego wyboru elementu głównego zdecydowanie jest najszybsza jeśli chodzi o mniejsze macierze, druga metoda radzi sobie minimalnie gorzej. Jeśli chodzi o wykorzystaną pamięć również najlepiej sprawuje się pierwsza metoda, ale tym razem niezależnie od wielkości macierzy. Największa różnica jest właśnie między nią, a wbudowaną operacją eliminacji Gaussa, która przy macierzy o rozmiarze 50 000 zajęła 70 MiB więcej. Jeśli chodzi o dokładność obliczeń zdecydowanie najlepiej sprawdziła się metoda z wyborem elementu głównego, bo precyzja jest wtedy rzędu 10^{-16} i nie spada nawet przy większych macierzach, w przeciwieństwie od metody pierwszej.

Wnioski

Jak widać czasem wbudowane metody nie są tymi najlepszymi i lepiej dostosować sobie je do swoich warunków, albo danych wejściowych. Przy przeanalizowaniu budowy macierzy i lekkim zmodyfikowaniu algorytmu podstawowego, byliśmy w stanie uzyskać rozwiązanie problemu, zaoszczędzając przy tym bardzo dużo miejsca w pamięci.