Obliczenia naukowe

Sprawozdanie nr 5

Barbara Banaszak
236514

06-01-2019

1 Opis problemu

Zadanie polegało na zaproponowaniu bardziej optymalnego algorytmu rozwiązania układu równań $Ax = b$ niż algorytmy dostępne w standardowych bibliotekach, dla macierzy A, $A \in R^{nxn}$, będącej macierzą rzadką i wektora prawych stron b, $b \in R^n$. Macierz A jest macierzą rzadką o strukturze blokowej:

$$A = \begin{bmatrix} A_1 & C_1 & 0 & .. & .. & 0 \\ B_2 & A_2 & C_2 & .. & .. & 0 \\ 0 & B_3 & A_3 & .. & .. & 0 \\ 0 & 0 & B_4 & .. & .. & 0 \\ : & : & : & : & : & \mathrm{C}_{n-1} \\ 0 & 0 & 0 & .. & .. & A_n \end{bmatrix}$$

Natomiast macierze A_i, B_i, C_i to kwadratowe macierze, każda rozmiaru lxl. Dodatkowo macierz $A_i \in R^{lxl}$ jest macierzą gęstą, $\mathbf{0}$ jest macierzą zerową kwadratową stropnia l, macierz $B_i \in R^{lxl}$ ma następującą postać:

$$B = \begin{bmatrix} 0 & 0 & .. & \mathrm{b}_1 \\ 0 & 0 & .. & \mathrm{b}_2 \\ : & : & .. & : \\ 0 & 0 & .. & \mathrm{b}_n \end{bmatrix}$$

macierz $C_i \in R^{lxl}$ jest macierzą diagonalną o następującej postaci:

$$B = \begin{bmatrix} \mathrm{c}_1 & 0 & .. & 0 \\ 0 & \mathrm{c}_2 & .. & 0 \\ : & : & .. & : \\ 0 & 0 & .. & \mathrm{c}_n \end{bmatrix}$$

Zaproponowane rozwiązania mają się opierać o

1. metodę eliminacji Gaussa
   (a) bez wyboru elementu głównego
   (b) z częściowym wyborem elementu głównego
2. rozkład LU macierzy A
   (a) bez wyboru elementu głównego
   (b) z częściowym wyborem elementu głównego
3. rozwiązywanie układu równań z zastosowaniem rozkładu LU z poprzedniego podpunktu

2 Rozwiązanie

2.1 Przechowywanie macierzy A

Warto na samym początku zauważyć, że skoro macierz A jest macierzą rzadką, trójdiagonalną, bardzo nieefektywne byłoby przechowywanie jej w postaci dwuwymiarowej tablicy nxn, ponieważ w ten sposób przechowywalibyśmy bardzo wiele niepotrzebnych elementów będacych zerami. Do efektywniejszego przechowywania macierzy A zostanie więc wykorzystana specjalna struktura dostępna w języku Julia, służąca do przechowywania macierzy rzadkich SparseMatrixCSC, w której będziemy przechowywać tylko niezerowe elementy macierzy.

2.2 Opis algorytmów

2.2.1 Rozwiązanie metodą eliminacji Gaussa

Metoda eliminacji Gaussa to algorytm popularnie stosowany miedzy innymi do rozwiązywania układów równań liniowych, a także do wyznaczania rozkładu LU macierzy. Algorytm metody eliminacji Gaussa opiera się na sprowadzeniu macierzy do postaci macierzy trójkątnej górnej. Aby uzyskac taką postać odejmujemy od siebie pokolei wiersze macierzy w taki sposób, aby jednocześnie zerować kolumny pod danym wierszem. Przykładowo w celu wyzerowania elementu $a_{i,j}$ od i-tego wiersza odejmujemy j-tą kolumnę wymnożoną przez $coef = a_{i,j}/a_{j,j}$. Gdy już otrzymamy macierz trójkątną górną w celu obliczenia wektora rozwiązań x macierzy stosujemy algorytm podstawiania wstecz, wyglądający następująco:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{i,j} * x_j}{a_{i,i}}$$

Metoda eliminacji Gaussa ma złożoność $O(n^3)$, a algorytm podstawiania wstecz $O(n^2)$, zatem rozwiązywanie układu równań tą metodą ma złożoność $O(n^3)$.

Naszym zadaniem jest zmodyfikowanie algorytmu metody Gaussa w taki sposób aby dla szczególnej postaci macierzy A możliwie zmniejszyć złożoność algorytmu. Zauważmy, że macierz A jest macierzą trójdiagonalną oraz, że dla pierwszych $l-1$ kolumn elementy niezerowe znajdują się w l pierwszych rzędach, dla kolejnych l kolumn w $2l$ rzędach i tak dalej, aż do ostatnich l kolumn, gdzie elementy niezerowe znajdziemy do kolumny $n - tej$ włącznie. Powyższe odserwacje pozwalają na zapisanie wzorów na indeksy ostatniego niezerowego miejsca w danej kolumnie i w danym wierszu:

$$lastNonZeroInColumn = \min\{l * floor(\frac{i}{l}) + l, n\}$$

$$lastNonZeroInRow = \min\{i + l, n\}$$

Po takich modyfikacjach złożoność obliczeniowa algorytmu wynosi: $(n-1) * 2l * l + n * l$, co przy założeniu, że l jest stałą daje złożoność $O(n)$. Algorytm eliminacji gaussa wygląda następująco:

INPUT: n (wielkość macierzy A), l (wielkość bloku w macierzy A), A (struktura SparseMatrixCSC przechowująca macierz A), b (wektor prawych stron równania)
OUTPUT: x (wektor długości n zawierający rozwiązanie równania)

for $k = 1$ **to** $n - 1$ **do**
  $lastNonZeroInColumn \leftarrow \min\{l * floor(\frac{i}{l}) + l, n\}$
  $lastNonZeroInRow \leftarrow \min\{i + l, n\}$
  for $j = i + 1$ **to** $lastNonZeroInColumn$ **do**
    $multpCoef \leftarrow A_{j,i}/A_{i,i}$
    $A_{j,i} \leftarrow 0$
    for $k = i + 1$ **to** $lastNonZeroInRow$ **do**
      $A_{j,k} \leftarrow A_{j,k} - multpCoef * A_{i,k}$
      $b_j \leftarrow b_j - multpCoef * b_i$
    end for
  end for
end for
$x_n \leftarrow b_n/A_{n,n}$
for $j = n - 1$ **to** 1 **do**
  $sum \leftarrow 0$
  $lastNonZeroInRow \leftarrow \min\{i + l, n\}$
  for $j = i + 1$ **to** $lastNonZeroInRow$ **do**
    $sum \leftarrow sum + x_j * A_{i,j}$
  end for
  $x_i \leftarrow (b_i - sum)/A_{i,i}$
end for
return x

2.2.2 Rozwiązanie metodą eliminacji Gaussa z częsciowym wyborem elementu głównego

Ta metoda elimininacji Gaussa polega na wybieraniu "elementu głównego" za każdym razem gdy zerujemy nową kolumnę. Wybór ten to w rzeczywistości znajdowanie elementu o największej wartości i ustawianie wiersza z tym elementem jako wiersz, od którego będziemy odejmować kolejne zerując kolumny. Przestawianie wierszy w macierzy może jest dosyć kosztowną operacją, więc zamiast przestawiać wiesze stworzymy kolecję służącą do przechowywania permutacji macierzy ($perm$). Zauważmy też, że drodnej korekty będzie wymagać wzór na ostatnie niezerowego miejsca w danej kolumnie, gdyż podczas eliminowania $l-1$ pierwszych kolumn najdalszy niezerowy element można stworzyć w kolumnie i indeksem $2l$ i tak dalej. Zatem poprawiony wzór:

$$lastNonZeroInColumn = \min\{l * floor(\frac{i}{l}) + l * 2, n\}$$

Algorytm eliminacji gaussa z cześciowym wyborem elemenru głównego wygląda następująco:

INPUT: n (wielkość macierzy A), l (wielkość bloku w macierzy A), A (struktura SparseMatrixCSC przechowująca macierz A), b (wektor prawych stron równania)
OUTPUT: x (wektor długości n zawierający rozwiązanie równania)

for $k = 1$ **to** $n - 1$ **do**
  $lastNonZeroInColumn \leftarrow \min\{l * floor(\frac{i}{l}) + l * 2, n\}$
  $lastNonZeroInRow \leftarrow \min\{i + l, n\}$
  for $j = i + 1$ **to** $lastNonZeroInColumn$ **do**
    $rowWithMaxElem \leftarrow findRowWithMaxElem()$
    swap $(perm[i], perm[rowWithMaxElem])$
    $A_{perm(j),i} \leftarrow 0$
    for $k = i + 1$ **to** $lastNonZeroInRow$ **do**
      $A_{perm(j),k} \leftarrow A_{perm(j),k} - multpCoef * A_{perm(i),k}$
      $b_{perm(j)} \leftarrow b_{perm(j)} - multpCoef * b_{perm(i)}$
    end for
  end for
end for
$x_n \leftarrow b_n / A_{n,n}$
for $j = n - 1$ **to** 1 **do**
  $sum \leftarrow 0$
  $lastNonZeroInRow \leftarrow \min\{i + l, n\}$
  for $j = i + 1$ **to** $lastNonZeroInRow$ **do**
    $sum \leftarrow sum + x_j * A_{perm(i),j}$
  end for
  $x_i \leftarrow (b_{perm(i)} - sum) / A_{perm(i),i}$
end for
return x

2.2.3 Rozkład LU macierzy

Rozkład LU jest przedstawieniem zadaniej macierzy w postaci dwóch macierzy: trójkątnej górnej (U) oraz trójkątnej dolnej (L), gdzie elementy leżace na diagonali jednej z macierzy to same 1. Zakładamy, że jest to diagonala macierzy L. Rozkład LU macierzy uzyskujemy wykonując algorytm eliminacji Gaussa, z tą różnicą, że zamiast zerować dolny trójkąt macierzy wpisujemy w te miesjca wartości $multpCoef$ (dzięki czemu uzyskujemy też optymalny sposób przechowywania rozkładu LU w jednej macierzy) oraz nie wykonujemy algorytmu podstawiania wstecznego.

2.2.4 Rozwiązanie z zastosowaniem rozkładu LU

W celu rozwiązania układu równań z zastosowaniem rozkładu LU należy podzielić rozwiązywanie na dwa etapy. Na początku mamy układ postaci $LUx = b$. Pierwszym krokiem jest rozwiązanie układu $Ly = b$, a następnie układu $Ux = y$.

Do rozwiązania pierwszego układu wykorzystujemy algorytm podstawiania w przód (analogiczny do algorytmu podstawiania wstecz), a do drugiego układu właśnie algorytm podstawiania wstecz. Warto zauważyć także, że macierz A w postaci LU ciągle ma zera na tych samych miejscach, zatem możemy zastosować podobne modyfikacje jak w przypadku algorytmu Gaussa i tym samym otrzymać złożoność obliczniową algorytmu rzędu $O(n)$. Algorytm rozwiązywania równania z rozkładu LU wygląda następująco:
INPUT: n (wielkość macierzy A), l (wielkość bloku w macierzy A), A (struktura SparseMatrixCSC przechowująca macierz A), b (wektor prawych stron równania)
OUTPUT: x (wektor długości n zawierający rozwiązanie równania)

for $i = 1$ **to** n **do**
  $sum \leftarrow 0$
  $lastNonZeroInRow \leftarrow \min\{i + l, n\}$
  for $j = lastNonZeroInRow$ **to** $i - 1$ **do**
    $sum \leftarrow sum + y_j * A_{i,j}$
  end for
  $y_i \leftarrow (b_i - sum)$
end for
$x_n \leftarrow y_n / A_{n,n}$
for $j = n - 1$ **to** 1 **do**
  $sum \leftarrow 0$
  $lastNonZeroInColumn \leftarrow \min\{l * floor(\frac{i}{l}) + l, n\}$
  for $j = i + 1$ **to** $lastNonZeroInColumn$ **do**
    $sum \leftarrow sum + x_j * A_{i,j}$
  end for
  $x_i \leftarrow (y_i - sum)/A_{i,i}$
end for
return x

3 Wyniki

Wartości błędu względnego rozwiązań równania		
n	Gauss	Gauss z wyborem
16	1.2281842209915794e-15	1.2281842209915794e-15
10000	3.9500624993138445e-16	3.9500624993138445e-16
50000	2.2448709557920664e-17	2.2448709557920664e-17

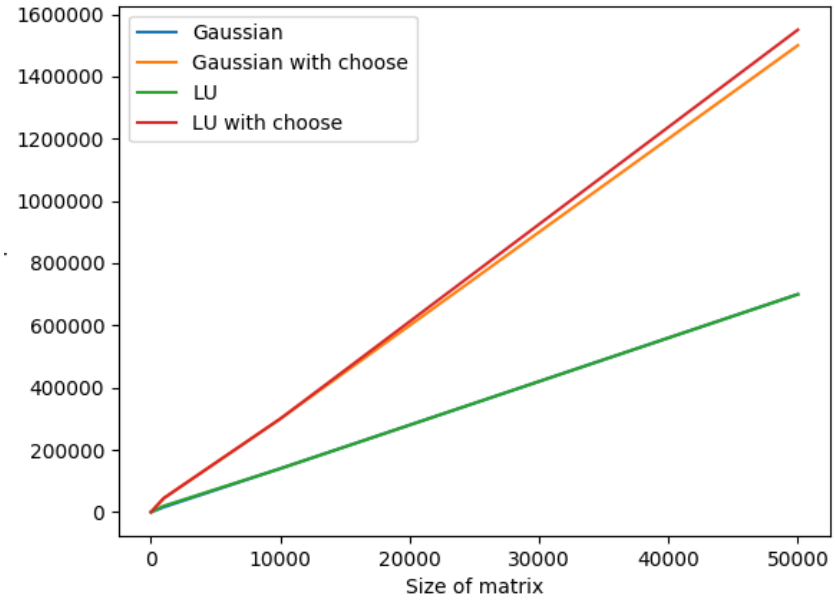
Wartości błędu względnego rozwiązań równania		
n	LU	LU z wybore
16	3.9500624993138445e-16	1.2281842209915794e-15
10000	3.9500624993138445e-16	3.9500624993138445e-16
50000	2.2448709557920664e-17	2.2448709557920664e-17

Powyższe tabele pokazują błędy względne rozwiązań policzonych za pomocą zmodyfikowanych algorytmów, błędy te nie są duże i wynikają z niedokładności obliczeń. Możemy zatem stwierdzić, że zaimplementowane algorytmy działają poprawnie.

Zestawienie czasu/ pamięci dla eliminacji Gaussa			
n	x = A / b	bez wyboru	z wyborem
16	0.468430s/ 13.548 MiB	0.100119s/ 4.267 MiB	0.083971s/999.335KiB
100	0.472470s/ 13.549 MiB	0.102510s/4.290 MiB	0.101958s/ 1.201 MiB
1000	0.511455s/ 13.555 MiB	0.203764s/ 5.107 MiB	0.165108s/ 23.881 MiB
10000	0.464217s/13.624 MiB	0.305966s/8.476 MiB	2.031771s / 2.236 GiB
50000	–/–	5.910332s/ 15.326 MiB	47.430418 /55.881 GiB

Zestawienie czasu/ pamięci dla rozkladu LU			
n	x = A / b	bez wyboru	z wyborem
16	0.468430s/ 13.548 MiB	0.208710s/ 4.633 MiB	0.110134s/1.082 MiB
100	0.472470s/ 13.549 MiB	0.209397s/4.665 MiB	0.112148s/ 1.531 MiB
1000	0.511455s/ 13.555 MiB	0.207953s/ 5.279 MiB	0.242956s/ 46.891 MiB
10000	0.464217s/13.624 MiB	0.422708s/9.625 MiB	3.577839s / 4.472 GiB
50000	–/–	6.393552s/ 19.362 MiB	93.94221s/111.762 GiB

Po przyjrzeniu się powyższym dwóm tabelą niewątpliwie najważniejszą obserwacją jest to, że dla macierzy A o rozmiarze $50000x50000$ algorytm eliminacji Gaussa zaimplementowany w języku Julia nie jest w stanie rozwiązać układu równań $Ax = b$, natomiast zaimplementowane z uwzględnieniem specifiki macierzy algorytmy radzą sobie z takimi wielkościami, zatem udało się spełnić najeważniejsze z założeń zadania. Udało się zredukować złożość algorytmu z $O(n^3)$ do $O(n)$ dla szczególnej macierzy A. Co widać także na poniższym wykresie przedstawiającym ilość operacji potzebną do znalezienia rozwiązania w zależności od wielkości macierzy.


Gaussian
Gaussian with choose
LU
LU with choose
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
0
10000
20000
30000
40000
50000
Size of matrix