OBLICZENIA NAUKOWE
lista nr 5 - sprawozdanie

Łukasz Ciświcki

Styczeń 2024

1 Wstęp

Rozwiązanie układów równań liniowych jest podstawą wielu obliczeń. W celu uproszczenia opisu układów równań można zastosować macierze w następujący sposób.

$$Ax = b$$

1.1 Szczególny przypadek

W przypadku rozważanym w tym zadaniu, macierz współczynników A specyficzną formę. Macierze wewnętrzne A_k są gęste. Macierze $\mathbf{0}$ są macierzami zerowymi.

$$A = \begin{bmatrix} A_1 & C_1 & 0 & 0 & 0 & \dots & 0 \\ B_2 & A_2 & C_2 & 0 & 0 & \dots & 0 \\ 0 & B_1 & A_3 & C_3 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \dots & \vdots \\ 0 & \dots & 0 & B_{V-2} & A_{V-2} & C_{V-2} & 0 \\ 0 & \dots & 0 & 0 & B_{V-1} & A_{V-1} & C_{V-1} \\ 0 & \dots & 0 & 0 & 0 & B_v & A_v \end{bmatrix}$$

Macierze wewnętrzne B_k mają jedną kolumnę wartości niezerowych.

$$B_k = \begin{bmatrix} 0 & \dots & 0 & b_1^k \\ 0 & \dots & 0 & b_2^k \\ \vdots & & \vdots & \vdots \\ 0 & \dots & 0 & b_l^k \end{bmatrix}$$

Macierze wewnętrzne C_k są macierzami przekątniowymi.

$$C_k = \begin{bmatrix} c_1^k & 0 & 0 & \dots & 0 \\ 0 & c_2^k & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & c_{l-1}^k & 0 \\ 0 & \dots & 0 & 0 & c_l^k \end{bmatrix}$$

1.2 Potrzeba dostosowania algorytmów

Zadanie wymaga rozwiązania macierzy zbyt dużych do wykorzystania standardowych implementacji algorytmów. W celu poprawy złożoności obliczeniowej i pamięciowej należy uwzględnić specyficzną budowę macierzy współczynników **A** i zmodyfikować algorytmy.

2 Sposób zapamiętywania macierzy

Do zapamiętywania rzadkich macierzy jest wykorzystywana struktura `SparseMatrixCSC` z biblioteki SparseArrays. Pozwala ona zapisywać jedynie elementy niezerowe, co skutkuje liniową złożonością pamięciową.

3 Metoda Gaussa

3.1 Wersja podstawowa

Metoda eliminacji Gaussa składa się z dwóch etapów. W etapie pierwszym należy poprzez wykonywanie operacji elementarnych doprowadzić macierz współczyników do postaci macierzy trójkątnej górnej. Drugi etap polega na rozwiązaniu otrzymanego równania, zaczynając od dolnego wiersza macierzy.
Złożoność obliczeniowa podstawowego algorytmu jest równa $\mathcal{O}(n^3)$

3.1.1 Tworzenie macierzy trójkątnej

W celu doprowadzenia do macierzy trójkątnej najpierw jest usuwana niewiadoma x_1 z wierszy $i \in \{2, ...n\}$ poprzez odjęcie od nich odpowiedniej wielokrotności wiersza pierwszego. To działanie odpowiada wyznaczeniu niewiadomej x_1 z równania pierwszego i podstwieniu jej do równań w następnych wierszy. To skutkuje usunięciem niewiadomej x_1 ze wszystkich wierszy oprócz pierwszego. Tę procedurę należy powtórzyć dla każdego wiersza.
Gdy na przekątnej macierzy występuje zero algorytm nie zadziała. W przypadku wystąpienia bardzo małej liczby algorytm zwróci skrajnie niepoprawne wyniki. Aby rozwiązać taki przypadek należy dokonać odpowiedniej zamiany wierszy. Ten problem jest rozwiązywany w wersji algorytmu usprawnionej o częściowy wybór elementu głównego omówiony w dalszej części sprawozdania.

3.1.2 Rozwiązywanie układu od końca

Układ równań otrzymany w wyniku eliminacji można w łatwy sposób rozwiązać kolejno podstawiając pod samotne niewiadome w wierszach macierzy współczynników **A** wartości wektora prawych stron **b**, zaczynając od ostatniego wiersza. Otrzymany wynik służy do rozwiązania kolejnych wierszy.

3.1.3 Dostosowanie algorytmu do specyfiki problemu

Dla każdej kolumny trzeba wyeliminować $l - k\%l$ współczynników. Dodatkowo, dla każdej macierzy wewnętrznej A_k oprócz ostatniej konieczne jest wyeliminowanie całej kolumny macierzy wewnętrznej B_k długości l. To ograniczenie znacząco ogranicza liczbę współczynników koniecznych do wyzerowania.

3.1.4 Pseudokod

Algorithm 1 Eliminacja Gaussa

procedure GAUSSMETHOD(A, B, n, l)
  for $k = 1, \ldots, n-1$ **do** ▷ Pętla (1), $\mathcal{O}(n)$
    for $i \leftarrow k+1, \ldots, l - k\%l$ **do** ▷ Pętla (2), $\mathcal{O}(l)$
      $z = \dfrac{A[i,k]}{A[k,k]}$
      $A[i,k] \leftarrow 0$
      for $j \leftarrow k+1, \ldots, min(n, k+l)$ **do** ▷ Pętla (3), $\mathcal{O}(l)$
        $A[i,j] \leftarrow A[i,j] - z * A[k,j]$
      end for
      $B[i] \leftarrow B[i] - z * B[k]$
    end for
  end for
  $x \leftarrow \{0, \ldots, 0\}$
  for $k = n, \ldots, 1$ **do**
    s $= 0$
    for $j \leftarrow k+1, \ldots, min(n, k+l)$ **do**
      $s \leftarrow s + A[k,j] * X[j]$
    end for
    $x[k] \leftarrow \dfrac{B[k] - s}{A[k,k]}$
  end for
  return x
end procedure

3.1.5 Złożoność obliczeniowa

Zakładam zgodnie ze specyfikacją, że wszystkie operacje są wykonywane w czasie $\mathcal{O}(1)$. Z tego powodu do obliczenia złożoności obliczeniowej wystarczy pokazać

liczbę iteracji wykonywanych przez algorytm. Pętla (2) oraz Pętla(3) wykonają maksymalnie l iteracji, co oznacza, że złożonośc jest równa $\mathcal{O}(n * l^2)$

3.2 Wybór elementu głównego

Modyfikacja algorytmu eliminacji Gaussa niweluje problem pojawienia się zer lub bardzo małych liczb na przekątnej macierzy**A**. Polega ona na szukaniu w obecnie rozważanej kolumnie największego elementu i dokonaniu zamiany wiersza posiadającego znaleziony element z obecnie przetwarzanym wierszem. W praktyce wiersze nie są zamieniane miejscami w macierzy, a jedynie wartości tablicy permutacji odpowiadające pozycjom wierszy. To pozwala uniknąć wykonywania niepotrzebnych obliczeń.

3.2.1 Dostosowanie algorytmu do specyfiki problemu

Dostosowanie jest takie samo, jak w opisie standardowego algorytmu eliminacji Gaussa. Różnicą jest tworzenie wektora permutacji, określającego sposób redukcji macierzy.

3.2.2 Pseudokod

3.2.3 Złożoność obliczeniowa

W wersji algorytmu z częściowym wyborem w celu obliczenia złożoności obliczeniowej należy dodatkowo uwzględnić Pętlę (2), która jest odpowiedzialna za szukanie odpowiedniego elementu głównego. Dodatkowo, Pętla (4) może tym razem wykonać 2*l operacji. Te różnice nie zmieniają jednak asymptotycznej złożoności obliczeniowej, która nadal wynosi $\mathcal{O}(n * l^2)$

Algorithm 2 Eliminacja Gaussa z częściowym wyborem

procedure GaussMethodChoose(A, B, n, l)
  $p \leftarrow \{1, \ldots, n\}$
  for $k \leftarrow 1, \ldots, n-1$ **do** ▷ Pętla (1), $\mathcal{O}(n)$
    $maxRow \leftarrow k$
    $maxValue \leftarrow |A[k, p[i]|$
    for $i \leftarrow k, \ldots, k + (l - k\%l)$ **do** ▷ Pętla (2), $\mathcal{O}(l)$
      if $|A[k, p[i]]| > maxValue$ **then**
        $maxValue \leftarrow |A[k, p[i]]|$
        $maxRow \leftarrow i$
      end if
    end for
    $p[k] \iff p[maxRow]$
    for $i = k + 1, \ldots, k + (l - k\%l)$ **do** ▷ Pętla (3), $\mathcal{O}(l)$
      $z \leftarrow A[p[i], k]/A[p[k], k]$
      $A[p[i], k] = 0$
      for $f \leftarrow k + 1, \ldots, min(n, k + 2 * l)$ **do** ▷ Pętla (4), $\mathcal{O}(2 * l)$
        $A[p[i], j] \leftarrow A[p[i], j] - z * A[p[k], j]$
      end for
      $B[perm[i]] \leftarrow B[perm[i]] - z * B[perm[k]]$
    end for
  end for
  $x \leftarrow \{0, \ldots, 0\}$
  for $k = n, \ldots, 1$ **do**
    s $= 0$
    for $j \leftarrow k + 1, \ldots, min(n, k + 2 * l)$ **do**
      $s \leftarrow s + A[k, j] * X[j]$
    end for
    $x[k] \leftarrow \dfrac{B[p[k]] - s}{A[p[k], k]}$
  end for
  return x
end procedure

4 Wyniki eksperymentów

4.1 Omówienie wyników i obserwacje

4.1.1 Złożoność obliczeniowa

Zakładamy, że wszystkie operacje sa wykonywane ze złożonością $\mathcal{O}(1)$. To sprawia, że chcąc eksperymentalnie wyznaczyć złożoność algorytmów wystarczy policzyć liczbe wykonanych iteracji. Wykres 1 wskazuje, że dostosowane algorytmy są wykonywane z liniową złożonościa obliczeniową $\mathcal{O}(n)$.

4.1.2 Czas wykonywania

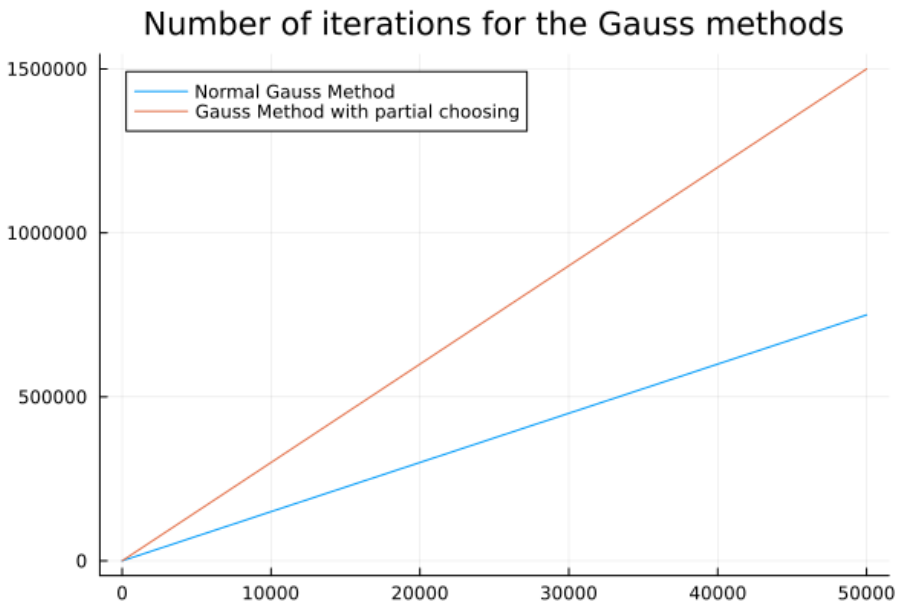
Czas wykonywania algorytmów został zmierzony przez makro @timed. Wykres 2 wskazuje, że rzeczywisty czas wykonania algorytmów przypomina wykres złożoności $\mathcal{O}(n^2)$.

4.1.3 Złożoność pamięciowa

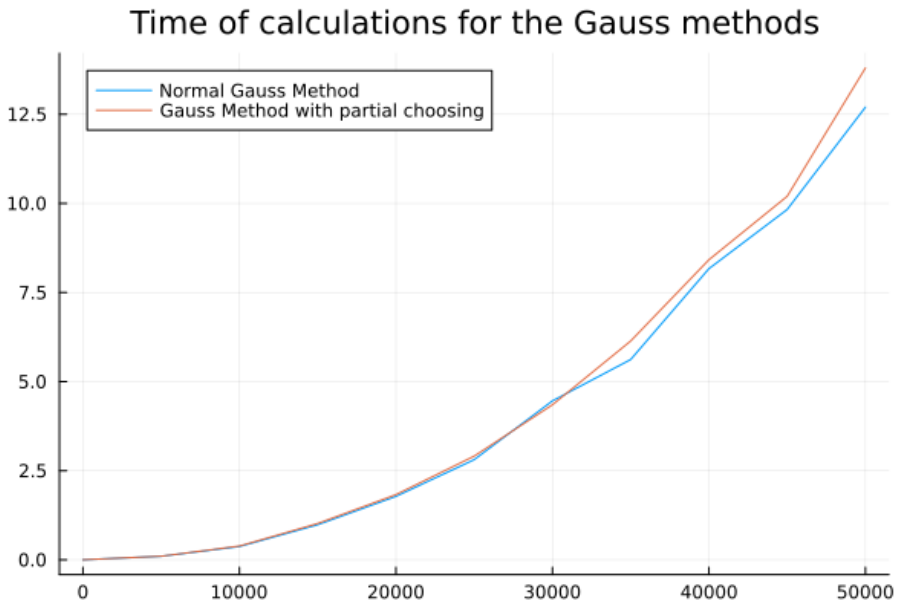
Ilość wykorzystanej pamięci została zmierzona przez makro @allocated. Wykres 3 wskazuje, że w modyfikacja algorytmów zapewniła w praktyce liniową złożoność pamięciową $\mathcal{O}(n)$.

4.2 Wnioski

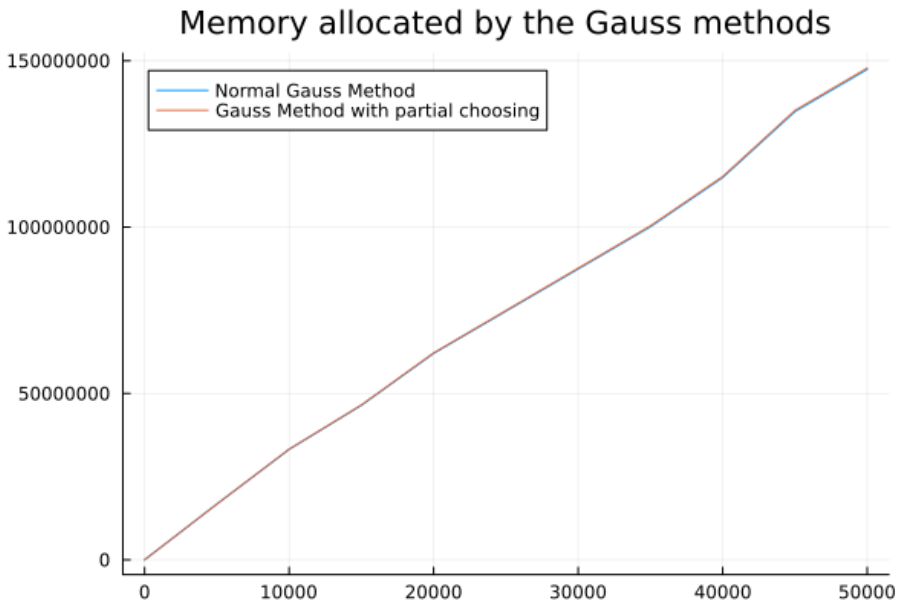
Założenie o czasie wykonywania operacji algorytmów w czasie $\mathcal{O}(1)$ nie ma pokrycia w rzeczywistości. Mimo to, ograniczenie liczby iteracji wymaganej do przeprowadzenia algorytmów umożliwiło znaczne zmniejszenie złożoności obliczeniowej i pamięciowej. Algorytm wybierający element główny wykonuje więcej iteracji co do stałej od standardowego algorytmu, lecz zapewnia przeprowadzenie dokładnych obliczeń nawet w sytuacji, gdy macierz współczynników ma niekorzystny układ.



Rysunek 1: Liczba iteracji wykonanych przez algorytmy



Rysunek 2: Rzeczywisty czas wykonywania algorytmów



Rysunek 3: Pamięć wykorzystana przez algorytmy