Michał Budnik – Sprawozdanie 5

Wstęp

Piąta lista jest zaprezentowana jako projekt optymalizacyjny dla jednostki badawczej pewnej firmy. Projekt ma na celu optymalizację rozwiązania układu równań linowych, w którym macierz współczynników jest macierzą rzadką.

Zadania

Zadanie 1

1.1 Opis zagadnienia

Zadanie polega na rozwiązaniu układu równań $Ax = b$ metodą eliminacji Gaussa. Rozwiązanie ma uwzględniać specyficzną postać macierzy. Zaimplementowane powinny zostać dwa warianty:

- bez wyboru elementu głównego,
- z częściowym wyborem elementu głównego.

1.2 Rozwiązanie

1.2.1 Opis metody eliminacji Gaussa

Metoda eliminacji Gaussa polega na stopniowej redukcji ilości niewiadomych w kolejnych równaniach układu. Celem metody jest sprowadzenie macierzy współczynników układu do macierzy górnotrójkątnej, co umożliwia proste rozwiązanie układu, wyznaczając niewiadome od końca. Dodatkowo w metodzie Gaussa zmienia się również wartości wektora wynikowego, stąd celem końcowym jest przekształcenie układu $Ax = b$ do układu $Ux = c$, gdzie U to macierz górnotrójkątna.

Algorytm sprowadzania macierzy do postaci trójkątnej, oraz aktualizacji wektora wynikowego przedstawia się następująco (dla eliminacji k-tej zmiennej z równań k+1,...,n):

```
For i in k+1:n
  l_ik = a_ik / a_kk
  subEquation = l_ik * (k-te równanie)
  (i-te równanie) -= subEquation
End
```

Założeniem poprawnego działania algorytmu jest $a_{kk} \neq 0$ ($a_{kk} \vee a_{k+1\,k} \vee \ldots \vee a_{nk} \neq 0$ dla wariantu wyboru elementu głównego). Po wykonaniu powyższego algorytmu dla k = 1,...,n, macierz A zostanie przekształcona do postaci górnotrójkątnej U, a wektor b do wektora docelowego c. W takiej formie wyznaczenie wektora wynikowego sprowadza się do wyznaczenia kolejnych wartości x zgodnie ze wzorem:

$$x_n = \frac{b_n}{u_{nn}}$$
$$x_k = \frac{b_k - \sum_{j=k+1}^{n} u_{kj} x_j}{u_{kk}}, dla\ k = n-1, \ldots, 1$$

Dla wariantu z częściowym wyborem elementu głównego a_{kk} w podanym algorytmie jest wyznaczane przy każdej iteracji algorytmu jako $\max|a_{ik}|\ dla\ k \leq i \leq n$. Jeżeli $i \neq k$ to dane równania zostają zamienione ze sobą kolejnością w macierzy U oraz wektorze c. Metoda jest stosowana w celu minimalizacji błędu, który może wystąpić przy małych wartościach a_{kk}.

1.2.2 Opis implementacji

Ze względu na specyficzną budowę macierzy A istnieje tutaj duża możliwość optymalizacji problemu. Wersja zaimplementowana oraz przedstawiona w tym sprawozdaniu działa w następujący sposób:

```
For v in 1 : n/l
  For k in startOfBlockRow : endOfBlockARow
    For i in k+1 : endOfBlockBRow
      l_ik = a_ik / a_kk
      For j in k+1 : endOfBlockCols
        A_ij = A_ij − l_ik * A_kj
      End
      b_i = b_i − l_ik * b_k
    End
  End
End
```

Powyższa część kodu pokazuje zaimplementowaną metodę Gaussa, uwzględniając jednak blokowość odczytywanej macierzy A. Zmienna v odpowiada w którym bloku rozmiaru $l \; x \; l$ aktualnie pracuje algorytm. Następnie przeprowadzana jest metoda Gaussa tylko w przedziale wyznaczanym na podstawie v. Objaśnienie zmiennych odpowiadających za odpowiednie przedziały:

- `startOfBlockRow` $= (v-1) * l + 1$: index wiersza (oraz kolumny) od którego zaczyna się blok v,
- `endOfBlockARow` $=$ startOfBlockRow $+ \; l \; - \; 1$: index wiersza kończącego macierz A_v,
- `endOfBlockBRow` $= min(rowStart + 2 * l - 1, \; n)$: index wiersza kończącego macierz B_{v+1},
- `endOfBlockCols` $= min(k * l + l, \; n)$: index kolumny kończącej macierz C_v.

Każdy blok w którym wykonuje się metoda Gaussa składa się więc z macierzy $A_v, C_v, A_{v+1}, B_{v+1}, C_{v+1}$. Można zauważyć, że macierz B_v nie jest brana pod uwagę, jednak dzieję się tak ponieważ przed obliczaniem bloku v wszystkie jej wartości są już równe 0. Dodatkowo należy zauważyć, że dla każdego bloku v nie należy uwzględniać macierzy o indeksie większym niż $v + 1$, ponieważ wartości dodawane do tych równań będą zawsze równe 0.

W przypadku wariantu z częściowym wyborem elementu głównego, w kodzie jest dodatkowo sprawdzane, czy w kolejnych wierszach (do wiersza `endOfBlockBRow`) istnieje wartość większa niż aktualne A_{kk}. Jeżeli tak, to wykorzystywana jest tablica permutacji która maskuje podmianę wierszy w macierzy A. Dzięki temu maskowaniu niepotrzebne jest kopiowanie faktycznych wartości macierzy (co może być kosztowne).

Ostatnią częścią zadania jest wyliczenie wartości układu. Jest to robione zgodnie ze wzorem podanym przy opisie metody Gaussa, zachowując przy tym przedziały wynikające z blokowości macierzy A.

```
For v in n/l : -1 : 1
  For i in endOfBlockARow : -1 : startOfBlockRow
    Sum = 0
    For j in i+1 : colEnd
      Sum += A[i, j]*x[j]
    End
    x[i] = (b[i] - sum)/A[i, i]
  End
End
```

1.2.3 Analiza złożoności

Złożoność obliczeniowa tego algorytmu jest zależna od ilości iteracji w każdej pętli, tak więc dla sprowadzania macierzy do postaci trójkątnej:

1. pętla blokowa: n/l operacji,
2. pętla wierszy macierzy A_v: l operacji,
3. pętla wierszy macierzy A_v i B_{v+1}: od $2l$ do l operacji (w zależności od stanu pętli 2)
4. pętla kolumn macierzy A_v i C_v: od $2l$ do l operacji (w zależności od stanu pętli 2)

Stąd całkowita ilość operacji jest mniejsza niż $\frac{n}{l} * l * 2l * 2l = 4l^2n$, co daje złożoność obliczeniową rzędu $O(n)$, przy założeniu, że l jest stałą. Podobną złożoność posiada część wyliczająca wartości niewiadomych. Ilość operacji to mniej niż $\frac{n}{l} * l * 2l = 2ln$, co również daje złożoność obliczeniową rzędu $O(n)$. Zatem całkowita złożoność obliczeniowa algorytmu to $O(n)$.

Z taką samą złożonością obliczeniową mamy do czynienia dla metody Gaussa z częściowym wyborem elementu głównego - złożoność zwiększa się jedynie o stałą.

Złożoność pamięciowa programu zależy od ilości zapamiętywanych danych. Dzięki wykorzystaniu SparseMatrix możliwe jest efektywne przechowywanie macierzy A o nawet bardzo wielkich rozmiarach. Należy również zauważyć, że wszystkie operacje wykonywane są wewnętrznie w macierzy A oraz wektorze b, więc na potrzeby algorytmu jedyną dodatkowo zaalokowaną pamięcią jest pamięć zachowująca wyniki wartości zmiennych x. Znając ilość elementów macierzy (ilość elementów w macierzy A_v to $n * l$, B_v to $n - l$, a C_v to $n - l$, a w wektorach b i x to n), można stwierdzić, że złożoność pamięciowa tego algorytmu to $O(n)$.

Taką samą złożoność daje metoda Gaussa z częściowym wyborem elementu głównego – należy jedynie zwiększyć stałą o 1, w związku z zapamiętaniem dodatkowego wektora permutacji.

1.3 Wyniki

Dla odpowiednich funkcji zostały przeprowadzone testy na bazie losowo generowanych macierzy funkcją blockmat z modułu matrixgen.

n	Bez wyboru elementu głównego		Z wyborem elementu głównego	
	Czas (s)	Błąd względny	Czas (s)	Błąd względny
16	0.007970	-4.440892098500626e-14	0.007773	3.3306690738754696e-14
10000	0.423703	-2.220446049250313e-14	0.430909	4.440892098500626e-15
50000	5.848441	-8.881784197001252e-14	6.457342	8.220446049250313e-15

1.4 Wnisoki

Można zauważyć delikatną różnicę w błędzie względnym pomiędzy metodami. Stosując metodę Gaussa uwzględniającą wybór elementu głównego, błąd względny maleje. Nie ma jednak zauważalnych różnic jeżeli chodzi o czas wykonywania programów. Z tych względów wywnioskować można, że bezpieczniejsze jest używanie wariantu z wyborem elementu głównego, ponieważ zarówno teoretycznie, jak i praktycznie, jest on bezpieczniejszy.

Mając również na uwadzę złożoność obliczeniową, można zauważyć, że analizując dokładnie zadany problem i dostosowując algorytmy pod niego można uzyskać o wiele lepsze wyniki czasowe (w tym zadaniu optymalizacja ze standardowej złożoności sześciennej do liniowej).

Zadania 2 i 3

2.1 Opis zagadnienia

Zadanie 2 polega na implementacji funkcji wyznaczającej rozkład LU macierzy A metodą eliminacji Gaussa. Funkcja powinna uwzględniać specyficzną postać macierzy A, oraz umżliwać pracy w dwóch wariantach:

- bez wyboru elementu głównego,
- z częściowym wyborem elementu głównego.

Zadanie 3 polega na implementacji funkcji rozwiązującej układ $Ax = b$, używajac wcześniej wyznaczonego rozkładu LU.

2.2 Rozwiązanie

2.2.1 Opis metody

Rozkład macierzy A do postaci $A = LU$ jest równoważna metodzie eliminacji Gaussa. Wyznaczona macierz L jest macierzą dolnotrójkątną, a macierz U jest górnotrójkątna. Wyznaczenie tych macierzy wynika bezpośrednio z metody eliminacji Gaussa, mianowicie proces sprowadzania macierzy A do macierzy górnotrójkątnej U w $n-1$ krokach można zapisać jako

$$U = L^{(n-1)}L^{(n-2)}\dots L^{(1)}A$$

Stąd: $A = {L^{(1)}}^{-1} * {L^{(2)}}^{-1} * \dots {L^{(n-1)}}^{-1} * U$

Znając rozkład $A = LU$ zadanie sprowadza się do rozwiązania dwóch układów: $Ly = b$ i $Ux = y$.

2.2.2 Opis implementacji

W celu uzyskania rozkładu LU należy wykonać algorytm analogiczny do algorytmu w zadaniu 1. Należy jednakże nie zmieniać wartości wektora b. Dodatkowo, w celu optymalizacji, rozkład LU można wyznaczyć bez alokacji dodatkowej pamięci. W tym celu należy zapamiętywać mnożniki l_{ik} na miejscu wyzerowanych współczynników w k-tym kroku algorytmu. Uwzględniając częściowy wybór elementu głównego również stosowana jest metoda analogiczna do zadania 1 – tworzony jest dodatkowo wektor permutacji p, będący mapą wierszy macierzy A.

Posiadając rozkład LU, oraz wektor permutacji p, rozwiązanie sprowadza się do wyliczenia dwóch układów równań $Ly = b$ oraz $Ux = y$. Wyliczenie $Ly = b$ jest niczym innym, niż aktualizacją wektora b z metody eliminacji Gaussa. Następuje więc iteracja po odpowiednich miejscach macierzy A (w której zapisany został rozkład LU). Obliczenie $Ux = y$ jest natomiast identyczne do obliczenia wartości niewiadomych z zadania 1.

```
#Wyliczanie Ly = b
  For k in 1 : n/l
    For i in startOfBlockRow : endOfBlockARow
      For j in i+1 : endOfBlockBRow
        b[j] = b[j] - A[j,i]*b[i]
      End
    End
  End

#Wyliczanie Ux = y
  For k in n/l : -1 : 1
    For i in endOfBlockARow : -1 : startOfBlockRow
      Sum = 0
      For j in i+1 : endOfBlockCols
        Sum += A[i,j] * x[j]
      end
      x[i] = (b[i] - Sum) / A[i, i]
    end
  end
```

2.2.3 Analiza złożoności

Ze względu na podobieństwo metody do metody eliminacji Gaussa należy przeanalizować jedynie różnice między nimi, żeby stwierdzić jak zmieniła się złożoność obliczeniowa oraz pamięciowa.

Mówiąc o złożoności obliczeniowej należy zauważyć, że przy wyznaczaniu rozkładu LU zmieniły się tylko dwie linie kodu, jedna została kompletnie usunięta (aktualizowanie wartości wektora b), druga natomiast zapisuje wartość l_{ik} w miejsca macierzy w trakcie wykonywania rozkładu. Dzięki temu złożoność obliczeniowa rozkładu jest rzędu $O(n)$. Przy obliczaniu wartości niewiadomych pojawiła się teraz dodatkowa funkcja. Wyliczanie $Ux = y$ odpowiada temu z rozkładu Gaussa, więc jego złożoność to również $O(n)$. Funkcja wyliczająca $Ly = b$ ma natomiast mniej niż $\frac{n}{l} * l * 2l$ operacji, co daje jej również złożność rzędu $O(n)$. Tak więc zarówno rozkład LU jak i wyliczanie równań $Ly = b$ oraz $Ux = y$ mają złożoność liniową.

Złożoność pamięciowa zadań 2 i 3 również pozostała taka, jak w zadaniu pierwszym. Zarówno do wyznaczenia rozkładu LU jak i do obliczenia niewiadomych nie została zaalokowana żadna dodatkowa pamięć, stąd złożoność pamięciowa tych programów to również $O(n)$.

2.3 Wyniki

Dla odpowiednich funkcji zostały przeprowadzone testy na bazie losowo generowanych macierzy funkcją blockmat z modułu matrixgen.

n	Bez wyboru elementu głównego		Z wyborem elementu głównego	
	Czas (s)	Błąd względny	Czas (s)	Błąd względny
16	0.008297	5.551115123125783e-14	0.009153	2.220446049250313e-14
10000	0.431370	3.3306690738754696e-14	0.373741	2.220446049250313e-14
50000	7.206663	2.220446049250313e-14	7.174395	4.440892098500626e-15

2.4 Wnioski

Tak jak w przypadku rozwiązywania układu równań metodą elimiacji Gaussa, nie widać tutaj dużej różnicy pomiędzy działaniem algorytmu bez wyboru elementu głównego, a z jego wyborem. Warto jednak tutaj porównać metodę wyznaczania i obliczania LU do metody Gaussa. Można zauważyć, że czasy wykonania dla wszystkich przypadków są delikatnie wyższe dla metodu wykorzystującej rozkład LU w porównaniu do metody rozkładu Gaussa. Dzieję się tak prawdopodobnie przez konieczność osobnego obliczenia równań $Ly = b$, któremu odpowiadająca część jest wykonywana automatycznie w metodzie Gaussa. Jednakże, posiadając taką samą macierz A, a kilka równych wektorów prawych stron (b), metoda rozkładu LU byłaby o wiele wydajniejsza, ponieważ raz obliczona macierz LU mogłaby zostać zutylizowana ponownie.