Sprawozdanie

Obliczenia naukowe, lista 5

Aleksandra Malinowska, 244925, WPPT INF, semestr 5, styczeń 2020

1. Wstęp

Sprawozdanie to zawiera analizę oraz rozwiązanie problemu przedstawionego na liście zadań numer 5 doktora Pawła Zielińskiego z przedmiotu Obliczenia naukowe.

2. Opis problemu

Problemem tej listy było opracowanie modułu o nazwie `blocksys` w języku Julia, zawierającego funkcje, dzięki którym możliwe będzie rozwiązanie modeli pewnych zjawisk chemii kwantowej. Ich rozwiązanie bowiem sprowadza się do rozwiązania układu równań liniowych

$$Ax = b$$

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

$$A = \begin{pmatrix} A_1 & C_1 & 0 & 0 & 0 & \cdots & 0 \\ B_2 & A_2 & C_2 & 0 & 0 & \cdots & 0 \\ 0 & B_3 & A_3 & C_3 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & B_{v-2} & A_{v-2} & C_{v-2} & 0 \\ 0 & \cdots & 0 & 0 & B_{v-1} & A_{v-1} & C_{v-1} \\ 0 & \cdots & 0 & 0 & 0 & B_v & A_v \end{pmatrix}$$

gdzie $v = \frac{n}{l}$, zakładając $l \mid n$, gdzie $l \geq 2$ jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych: A_k, B_k, C_k. Mianowicie, $A_k \in \mathbb{R}^{l \times l}$ $(k = 1, \ldots$ $\mathbb{R}^{l \times l}$ $(k = 1, \ldots, v)$ jest macierzą gęstą, 0 jest kwadratową macierzą zerową stopnia l, macierz $B_k \in \mathbb{R}^{l \times l}$ $(k = 2, \ldots, v)$ ma tylko dwie ostatnie kolumny niezerowe, natomiast $C_k \in \mathbb{R}^{l \times l}$ $(k = 1, \ldots, v-1)$ jest macierzą diagonalną.

Istotą problemu jest takie zaprojektowanie funkcji w module, aby jak najlepiej zoptymalizować rozwiązywanie problemu pod względem czasowym i pamięciowym dla bardzo dużych wartości n.

3. Lista zadań

3.1. Zadanie 1

3.1.1. Opis problemu

Pierwszym problemem na liście było napisanie funkcji rozwiązującej układ $Ax = b$ metodą eliminacji Gaussa uwzględniając przy tym specyficzną postać macierzy A dla dwóch wariantów: (1) bez wyboru elementu głównego oraz (2) z częściowym wyborem elementu głównego.

3.1.2. Algorytm

3.1.2.1. Eliminacja Gaussa

Klasyczna wersja algorytmu eliminacji Gaussa polega na sprowadzeniu układu równań do macierzy trójkątnej wykonując przy tym tylko operacje takie jak dodawanie, odejmowanie czy mnożenie przez niezerową stałą na kolumnach i wierszach, które dodatkowo można dowolnie przestawiać. Wynikiem tych operacji jest macierz, dzięki której wraz z wektorem prawych stron można obliczyć wektor rozwiązań układu równań.

Kluczowym elementem algorytmu jest przetworzenie macierzy wejściowej na trójkątną. Aby tego dokonać należy wyeliminować z niej elementy niezerowe znajdujące się pod diagonalną. Przypatrując się strukturze macierzy podanej w zadaniu można zauważyć, że dla $l = 4$ w kolejnych kolumnach musimy wyzerować kolejno $3,2,5,4,3,2,5,4\ldots$ elementów, które występują pod diagonalną jedna pod drugą. Wiedząc to, przy oznaczeniu k jako numer kolumny możemy stwierdzić, że w każdej kolejnej kolumnie należy wyeliminować elementy w wierszach od k do $k + l - modulo(k, l)$, gdzie `modulo(a, b)` jest funkcją postaci:

$$
\begin{array}{l}
\boldsymbol{modulo}(a, b) \\
\quad if\ (a \bmod b = b - 1)\ then \\
\quad\quad return\ b \\
\quad else\ return\ a \bmod b \\
end\ function
\end{array}
$$

Budowa powyższej funkcji wynika z faktu, że dwie ostatnie kolumny w macierzy B_k są niezerowe (gdyby tylko ostatnia kolumna była niezerowa eliminowalibyśmy tylko elementy na pozycjach od k do $k + l - k \bmod l$).

Aby wyeliminować element a_{ik} należy od i-tego wiersza odjąć k-ty wiersz pomnożony przez współczynnik $a \leftarrow \frac{a_{ik}}{a_{kk}}$. Warto zauważyć, że może tu dojść do dzielenia przez zero. Aby tego uniknąć należy zamienić ze sobą miejscami kolumny lub wiersze. Należy pamiętać, aby równocześnie dokonywać zmian również w wektorze prawych stron.

Dokładny przebieg tego algorytmu przedstawia pseudokod poniżej.

$$
\begin{array}{l}
\boldsymbol{eliminacjaGaussa}(A, b, n, l, zRozkladem) \\
\quad for\ k \leftarrow 1\ to\ n-1\ do \\
\quad\quad m \leftarrow k + \big(l - modulo(l, k)\big) \\
\quad\quad for\ i \leftarrow k+1\ to\ m\ do \\
\quad\quad\quad a \leftarrow \frac{A_{ki}}{A_{kk}} \\
\quad\quad\quad if\ zRozkladem\ then \\
\quad\quad\quad\quad A_{ki} \leftarrow a \\
\quad\quad\quad else \\
\quad\quad\quad\quad A_{ki} \leftarrow 0 \\
\quad\quad\quad for\ j \leftarrow k+1\ to \min(k+l, n)\ do \\
\quad\quad\quad\quad A_{ji} \leftarrow A_{ji} - a * A_{jk} \\
\quad\quad\quad end\ for \\
\quad\quad\quad if\ zRozkladem = false\ then \\
\quad\quad\quad\quad b_i \leftarrow b_i - a * b_k \\
\quad\quad end\ for \\
\quad end\ for \\
\quad return\ A, b \\
end\ function
\end{array}
$$

Złożoność tego algorytmu wynosi $O(n)$, ponieważ główna pętla wykona się dokładnie n razy, natomiast zagnieżdżone w niej pętle co najwyżej l razy, stąd mamy $n * l^2$, gdzie l jest stałą, co daje złożoność liniową.

3.1.2.2. Eliminacja Gaussa z częściowym wyborem elementu głównego

Algorytm ten jest modyfikacją powyższego algorytmu, która ma na celu zapobieganie występowania zer na przekątnej macierzy. Polega ona na wyborze elementu głównego w kolumnie i odpowiednim przestawieniu wierszy macierzy, aby znalazł się on na przekątnej. Element ten (a dokładnie jego wartość bezwzględna) jest zawsze największym elementem w kolumnie. Przestawianie elementów w kolumnie można wyrazić następującym wzorem:

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

gdzie $s(k)$ jest wektorem permutacji.

Poniższy pseudokod przedstawia dokładny przebieg tego algorytmu.

$\boldsymbol{eliminacjaGaussaZWyborem}(A, b, n, l, zRozkladem)$
$\quad perm[1:n] \leftarrow \{1, \dots, n\}$
$\quad for\ k \leftarrow 1\ to\ n\ do$
$\quad\quad max_{wiersz} \leftarrow k$
$\quad\quad max_{el} \leftarrow |A_{kk}|$
$\quad\quad for\ i \leftarrow k + 1\ to\ k + l - modulo(k, l)\ do$
$\quad\quad\quad if\ \left|A_{k,perm_i}\right| > max_{el}\ then$
$\quad\quad\quad\quad max_{el} \leftarrow \left|A_{k,perm_i}\right|$
$\quad\quad\quad\quad max_{wiersz} \leftarrow i$
$\quad\quad end\ for$
$\quad\quad perm_k \leftrightarrow perm_{max_{wiersz}}$
$\quad\quad for\ i \leftarrow k + 1\ to\ k + l - modulo(k, l)\ do$
$\quad\quad\quad a \leftarrow \frac{A_{k,perm_i}}{A_{k,perm_k}}$
$\quad\quad\quad if\ zRozkladem\ then$
$\quad\quad\quad\quad A_{k,perm_i} \leftarrow a$
$\quad\quad\quad else$
$\quad\quad\quad\quad A_{k,perm_i} \leftarrow 0$
$\quad\quad\quad for\ j \leftarrow k + 1\ to\ min(k + 2 * l, n)\ do$
$\quad\quad\quad\quad A_{j,perm_i} \leftarrow A_{j,perm_i} - a * A_{j,perm_k}$
$\quad\quad\quad end\ for$
$\quad\quad\quad if\ zRozkladem = false\ then$
$\quad\quad\quad\quad b_{perm_i} \leftarrow b_{perm_i} - a * b_{perm_k}$
$\quad\quad end\ for$
$\quad end\ for$
$\quad return\ A, perm, b$
$end\ function$

Ze względu na swoje podobieństwo do algorytmu eliminacji Gaussa, algorytm ten ma również podobną złożoność. Jest ona liniowa, jednak ze względu na obecność dodatkowej pętli, w której wybierany jest główny element, wykonujemy dodatkowe operacje, które powodują, że złożoność ta jest większa, jednak z dokładnością do stałej.

3.1.2.3. Pseudokod funkcji

Poniżej znajduje się pseudokod algorytmów rozwiązujących układ równań przy pomocy odpowiednio eliminacji Gaussa oraz eliminacji Gaussa z częściowym wyborem elementu głównego.

$$
\begin{aligned}
&\boldsymbol{rozwiazUkladGauss}(A, b, n, l)\\
&\quad A', b' \leftarrow eliminacjaGaussa(A, b, n, l, false)\\
&\quad for\ i \leftarrow n\ to\ 1\ do\\
&\quad\quad suma \leftarrow 0\\
&\quad\quad for\ j \leftarrow i+1\ to\ \min(n, i+l)\ do\\
&\quad\quad\quad suma \leftarrow suma + A'_{ji} * x_j\\
&\quad\quad end\ for\\
&\quad\quad x_i \leftarrow \frac{(b'_i - suma)}{A'_{ii}}\\
&\quad end\ for\\
&\quad return\ x\\
&end\ function
\end{aligned}
$$

$$
\begin{aligned}
&\boldsymbol{rozwiazUkladGaussZWyborem}(A, b, n, l)\\
&\quad A', perm, b' \leftarrow eliminacjaGaussaZWyborem(A, b, n, l, false)\\
&\quad for\ i \leftarrow n\ to\ 1\ do\\
&\quad\quad suma \leftarrow 0\\
&\quad\quad for\ j \leftarrow i+1\ to\ min(n, i+2*l+1)\ do\\
&\quad\quad\quad suma \leftarrow suma + A'_{j,perm_i} * x_j\\
&\quad\quad end\ for\\
&\quad\quad x_i \leftarrow \frac{\left(b'_{perm_i} - suma\right)}{A'_{i,perm_i}}\\
&\quad end\ for\\
&\quad return\ x\\
&end\ function
\end{aligned}
$$

3.1.3. Rozwiązanie

Rozwiązaniem tego zadania jest implementacja powyższego pseudokodu w języku Julia w postaci funkcji odpowiednio `eliminacjaGaussa`, `eliminacjaGaussaZWyborem`, `rozwiazUkladGauss` oraz `rozwiazUkladGaussZWyborem` w module `blocksys`.

3.2. Zadanie 2

3.2.1. Opis problemu

Kolejnym problemem było napisanie funkcji wyznaczającej rozkład LU macierzy A metodą eliminacji Gaussa uwzględniając przy tym specyficzną postać macierzy A dla dwóch wariantów: (1) bez wyboru elementu głównego oraz (2) z częściowym wyborem elementu głównego.

3.2.2. Algorytm

Rozkład LU jest algorytmem silnie związanym z algorytmem eliminacji Gaussa, w którym wyznacza się nie jedną a dwie macierze trójkątne: górną (U) oraz dolną (L). W ten sposób otrzymujemy układ równań $LUx = b$, którego rozwiązanie sprowadza się do rozwiązania następujących dwóch układów równań:

$$Ux = b'$$

$$Lb' = b$$

Do wyznaczenia obu tych macierzy posłuży nieco zmodyfikowany algorytm eliminacji Gaussa, w którym zamiast zerować elementy w dolnej macierzy trójkątnej, wstawiamy w ich miejsce wartość a użytą do wyeliminowania tego elementu. Dodatkowo pomijana jest aktualizacja wektora b, ponieważ następuje to dopiero w momencie rozwiązywania układu przy użyciu rozkładu LU. Dotyczy to zarówno rozkładu LU metodą eliminacji Gaussa jak również metodą eliminacji Gaussa z częściowym wyborem elementu głównego.

Złożoność tych algorytmów jest taka sama jak złożoność algorytmu eliminacji Gaussa i wynosi $O(n)$.

3.2.3. Rozwiązanie

Rozwiązaniem tego zadania są funkcje `rozkladLU` oraz `rozkladLUZWyborem`, które uruchamiają odpowiednio funkcje `eliminacjaGaussa` oraz `eliminacjaGaussaZWyborem` z flagą `zRozkladem` ustawioną na wartość `true`.

3.3. Zadanie 3

3.3.1. Opis problemu

Ostatnim problemem na liście było napisanie funkcji, która rozwiąże układ równań $Ax = b$, jeśli wcześniej został już wyznaczony rozkład LU przez funkcję z zadania nr 2.

3.3.2. Algorytm

Jak już zostało wspomniane w rozdziale 3.2.2. aby rozwiązać układ równań z wyznaczonym rozkładem LU należy rozwiązać dwa układy: $Ux = b'$ oraz $Lb' = b$. Algorytm rozwiązywania tych układów (przy obliczeniu rozkładu poprzez eliminację Gaussa) przedstawia pseudokod poniżej.

$\boldsymbol{rozwiazUkladLU}(A, b, n, l)$
 $for\ i \leftarrow 1\ to\ n\ do$
  $suma \leftarrow 0$
  $for\ j \leftarrow \max\left(1, l * \left\lfloor \frac{i-1}{l} \right\rfloor\right)\ to\ i-1\ do$
   $suma \leftarrow suma + A_{ji} * b_j$
  $end\ for$
  $b_i \leftarrow b_i - suma$
 $end\ for$
 $for\ i \leftarrow n\ to\ 1\ do$
  $suma \leftarrow 0$
  $for\ j \leftarrow i+1\ to\ \min(n, i+l+1)\ do$
   $suma \leftarrow suma + A_{ji} * x_j$
  $end\ for$
  $x_i \leftarrow \frac{b_i - suma}{A_{ii}}$
 $end\ for$
 $return\ x$
$end\ function$

Algorytm rozwiązywania tych układów (przy obliczeniu rozkładu poprzez eliminację Gaussa z częściowym wyborem elementu głównego) przedstawia poniższy pseudokod.

$$
\begin{array}{l}
\boldsymbol{rozwiazUkladLUZWyborem}(A, perm, b, n, l) \\
\quad for\ i \leftarrow 1\ to\ n\ do \\
\quad\quad suma \leftarrow 0 \\
\quad\quad for\ j \leftarrow max\left(1, l * \left\lfloor \frac{i-1}{l} \right\rfloor\right) to\ i-1\ do \\
\quad\quad\quad suma \leftarrow suma + A_{j,perm_i} * b_j \\
\quad\quad end\ for \\
\quad\quad b_i \leftarrow b_{perm_i} - suma \\
\quad end\ for \\
\quad for\ i \leftarrow n\ to\ 1\ do \\
\quad\quad suma \leftarrow 0 \\
\quad\quad for\ j \leftarrow i+1\ to\ min(n, i+2*l+1)\ \ do \\
\quad\quad\quad suma \leftarrow suma + A_{j,perm_i} * x_j \\
\quad\quad end\ for \\
\quad\quad x_i \leftarrow \frac{b_i - suma}{A_{i,perm_i}} \\
\quad end\ for \\
\quad return\ x \\
end\ function
\end{array}
$$

Złożoność przedstawionych powyżej algorytmów wynosi $O(n)$.

3.3.3. Rozwiązanie

Rozwiązaniem tego zadania jest implementacja powyższego pseudokodu w języku Julia w funkcjach odpowiednio `rozwiazUkladLU` oraz `rozwiazUkladLUZWyborem` w module `blocksys`.

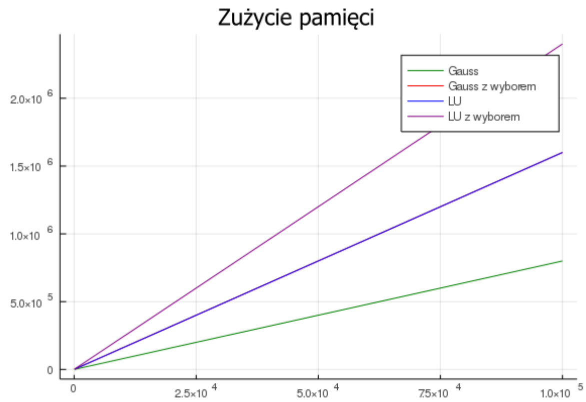
3.4. Sposób przechowywania macierzy rzadkich

Ważnym zastrzeżeniem w zadaniu było użycie specjalnej struktury, która pozwoli na zapamiętanie tylko niezerowych wartości macierzy A w ramach oszczędności pamięci. W języku Julia została ona nazwana `SparseArrays`. Implementacja tej struktury pozwala na szybszy dostęp do elementów poprzez kolumny niż przez wiersze. Na potrzeby obliczania złożoności czasowej powyższych algorytmów przyjęto, że dostęp do elementu macierzy przechowywanej w tej strukturze jest stały.

4. Testy

4.1. Wyniki

Aby porównać złożoność czasową i pamięciową przedstawionych algorytmów, przeprowadzone zostały testy dla wartości $l = 4$ oraz coraz większych wartości n. Do obliczenia rzeczywistego czasu potrzebnego do wykonania algorytmu oraz rzeczywistej ilości pamięci użyto dostępnego w języku Julia makra `@timed`. Wyniki tych testów przedstawiają poniższe wykresy.



4.2. Wnioski

Analizując powyższe wykresy można dojść do wniosku, że algorytmy opierające się na algorytmie eliminacji Gaussa z częściowym wyborem elementu głównego są (z dokładnością do stałej) bardziej czaso- i pamięciochłonne. Widać też, że zużycie pamięci rośnie liniowo wraz ze wzrostem wartości n. Czas działania natomiast, mimo iż powinien być liniowy, bardziej przypomina wykres funkcji kwadratowej. Wynika to z faktu, że w rzeczywistości dostęp do elementów macierzy nie jest stały, co powoduje wydłużenie czasu działania algorytmów. Można zauważyć również, że rozwiązywanie układu przy pomocy rozkładu LU jest na ogół szybsze niż przy pomocy eliminacji Gaussa.

5. Podsumowanie

Projekt ten zwraca szczególną uwagę na fakt, iż w niektórych sytuacjach mała modyfikacja klasycznego algorytmu daje możliwość znajdowania rozwiązań dla skomplikowanych problemów w szybki i łatwy sposób.