Obliczenia naukowe
Sprawozdanie
Lista 4

Mateusz Laskowski

06.01.2019

1. Zadanie 1

1.1. Opis problemu

Zaimplementuj metodę rozwiązującą układ $Ax = b$ metodą eliminacji Gaussa uwzględniając specyficzną postać macierzy $A \in \mathbb{R}^{n \times n}$. Macierz A mamy obliczyć dla dwóch wariantów:

- bez wyboru elementu głównego
- z częściowym wyborem elementu głównego

gdzie wektor prawych stron to $b \in \mathbb{R}^n$.

Macierz A ma specyficzną postać, która jest przedstawiona poniżej:

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

Gdzie $v = \frac{n}{l}$ przy założeniu, że n jest podzielne przez l, gdzie l jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków): A_k, B_k, C_k. Bloki $A_k \in \mathbb{R}^{l \times l}$, $k = 1, \dots, v$ jest macierzą gęstą, 0 jest kwadratową macierzą zerową stopnia l. Macierz $B_k \in \mathbb{R}^{l \times l}$, $k = 2, \dots, v$ jest następującej postaci:

$$\boldsymbol{B_k} = \begin{pmatrix} \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{b_1^k} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{b_2^k} \\ \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{b_l^k} \end{pmatrix}$$

B_k ma tylko jedną, ostatnią kolumnę niezerową. Natomiast blok $C_k \in \mathbb{R}^{l \times l}$, $k = 1, \dots, v - 1$ jest macierzą diagonalną:

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

Podczas implementacji trzeba wziąć pod uwagę, że złożoność algorytmu nie może wynosić jak w standardowym algorytmie eliminacji Gaussa, czyli $O(n^3)$, lecz $O(n)$.

1.2. Analiza eliminacji Gaussa

Metodę eliminacji Gaussa można podzielić na dwa oddzielne etapy. Pierwszy z nich to doprowadzenie macierzy do postaci schodkowej, czyli taka postać, gdzie niezerowe komórki znajdują się jedynie powyżej komórek o indeksie wiersza i kolumny równym

sobie. Drugim etapem w metodzie to rozwiązanie odpowiadającej tej macierzy układu równań. Poniżej przykład macierzy, którą chcemy uzyskać, czyli macierz schodkowa:

$$M = \begin{pmatrix} a & a & a \\ 0 & a & a \\ 0 & 0 & a \end{pmatrix}, gdzie\ a \neq 0$$

1.3. Opis rozwiązania

1.3.1. Eliminacja Gaussa bez wyboru elementu głównego

Pierwszy etap algorytmu to manipulacja macierzą – za pomocą operacji elementarnych – w celu uzyskania macierzy schodkowej. Operacja tego działania opiera się na wyborze wiersza głównego, następnie mnożeniu go przez odpowiedni czynnik – oznaczony β – i dodawaniu do każdych kolejnych wierszy tak, aby wiersze te w danej kolumnie zostały wyzerowane.

$$W_n = W_n + \beta W_1$$

$$\begin{pmatrix} \boldsymbol{a_{1,1}} & \boldsymbol{a_{1,2}} & \boldsymbol{a_{1.3}} & \cdots & \boldsymbol{a_{1,n}} \\ \boldsymbol{0} & \boldsymbol{a_{2,2}} & \boldsymbol{a_{2,3}} & \cdots & \boldsymbol{a_{2,n}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{a_{n-1,2}} & \boldsymbol{a_{n-1,2}} & \cdots & \boldsymbol{a_{n-1,n}} \\ \boldsymbol{0} & \boldsymbol{a_{n,2}} & \boldsymbol{a_{n,3}} & \cdots & \boldsymbol{a_{n,n}} \end{pmatrix}$$

W tym wypadku za wiersz główny został wybrany wiersz W_1, a przy każdym innym wierszu współczynnik β_n jest równy $0 = a_{i,1} - \beta a_{1,1}, czyli\ po\ prostu\ \beta = \frac{a_{i,1}}{a_{1,1}}$. Po wykonaniu wszystkich działań na kolejnych postaciach macierzy $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \cdots \rightarrow A_n$, otrzymując macierz trójkątną górną, wcześniej opisaną jako macierzą schodkową.

$$\begin{pmatrix} \boldsymbol{a_{1,1}} & \boldsymbol{a_{1,2}} & \boldsymbol{a_{1,3}} & \cdots & \boldsymbol{a_{1,n-1}} & \boldsymbol{a_{1,n}} \\ 0 & \boldsymbol{a_{2,2}} & \boldsymbol{a_{2,3}} & \cdots & \boldsymbol{a_{2,n-1}} & \boldsymbol{a_{2,n}} \\ 0 & 0 & \boldsymbol{a_{3,3}} & \cdots & \boldsymbol{a_{3,n-1}} & \boldsymbol{a_{2,n}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{a_{n-1,n-1}} & \boldsymbol{a_{n-1,n}} \\ 0 & 0 & 0 & \cdots & 0 & \boldsymbol{a_{n,n}} \end{pmatrix}$$

Mając już taką macierz trójkątną można przejść do etapu drugiego, czyli obliczania szukanego wektora. Zaczynając od ostatniego wiersza ostatniej kolumny można wyliczyć szukany wektor *x* rozwiązań równania.

1.3.2. Eliminacja Gaussa z wyborem elementu głównego

Algorytm wykonuje wszystkie operacje jak w powyższym opisanym schemacie działania, lecz trzeba dodać dodatkowe wstępne założenie, gdzie komórki $a_{k,k}$, które są używane w metodzie eliminacji Gaussa jako dzielniki nie mogą być równe zero. W przeciwnym wypadku dochodziłoby do dzielenia przez zero. Na dodatek trzeba

pamiętać, że wartości nie mogą być bliskie zera, ponieważ zostaną zaokrąglone do zera. Na straży oscylowaniu przy zerze będzie stał macheps, gdzie przy przyrównaniu z nim zwróci błąd i zakończy dalszemu wyliczaniu. Doprowadzenie macierzy do postaci, w której komórki $a_{k,k}$ mają jak największą wartość, co pozwala poprawić numeryczną dokładność późniejszych wyliczeń. Więc szukam wiersza, spośród wierszy W_i takich, że $i \in (k, \dots, n)$. Wybór spośród wierszy $i < k$ zaburzyłby strukturę macierzy, do której dążymy, czyli macierzy schodkowej, a dokładniej zerową dolną lewą macierz trójkątną. Przykładowe działanie powyżej opisanego schematu:

$$\begin{pmatrix} \boldsymbol{a_{1,1}} & \boldsymbol{a_{1,2}} & \boldsymbol{a_{1,3}} & \cdots & \boldsymbol{a_{1,n-1}} & \boldsymbol{a_{1,n}} \\ \boldsymbol{0} & \boldsymbol{a_{2,2}} & \boldsymbol{a_{2,3}} & \cdots & \boldsymbol{a_{2,n-1}} & \boldsymbol{a_{2,n}} \\ \boldsymbol{0} & \boldsymbol{a_{3,2}} & \boldsymbol{a_{3,3}} & \cdots & \boldsymbol{a_{3,n-1}} & \boldsymbol{a_{3,n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{a_{n-1,2}} & \boldsymbol{a_{n-1,3}} & \cdots & \boldsymbol{a_{n-1,n-1}} & \boldsymbol{a_{n-1,n}} \\ \boldsymbol{0} & \boldsymbol{a_{n,2}} & \boldsymbol{a_{n,3}} & \cdots & \boldsymbol{a_{n,n-1}} & \boldsymbol{a_{n,n}} \end{pmatrix}$$

Wybierana jest komórka o największej wartości spośród $a_{i,2}$, gdzie $i \in (2, n)$, a następnie zamienione zostają miejscami wiersze W_2 z wierszem W_i.

1.4. Złożoność algorytmu

Przy podanej w zadaniu macierzy, w przypadku normalnego działania eliminacji Gaussa, osiągnęlibyśmy złożoność $O(n^3)$. Zadanie polegało na tym, aby zminimalizować złożoność algorytmu do złożoności liniowej oraz osiągnąć jak najlepszą złożoność pamięciową. W celu osiągnięcia jak najmniejszej złożoności pamięciowej, do przechowywania wszelkich macierzy zostały użyte macierze typu SparseMatrixCSC w bibliotece języka Julia. Tego typu macierze są wykorzystywane w macierzach rzadkich, ponieważ zapamiętuje jedynie elementy niezerowe podawane do komórek. Aby móc osiągnąć złożoność liniową w eliminacji Gaussa na podanym schemacie macierzy należy wziąć jedynie komórki z wartościami niezerowymi, które znajdują się w blokach $A, B\ oraz\ C$. Aby tak uczynić, algorytm zawęża w każdej iteracji zakres wierszy do wyzerowania kolumny do l, ponieważ taka jest maksymalna rozpiętość niezerowych wartości. Natomiast maksymalny zakres kolumn w danym wierszu może być do $3l$. Takimi oto sprytnymi sposobami otrzymujemy złożoność $O(n \cdot l \cdot 3l)$, pamiętając, że l jest stałą, czyli niezależną od n, co w notacji duże *O* daje nam to liniową złożoność.

2. Zadanie 2

2.1. Opis problemu

Napisać funkcję wyznaczającą rozkład *LU* macierzy A metodą eliminacji Gaussa uwzględniającą specyficzną postać podanej macierzy A dla dwóch wariantów:

- bez wyboru elementu głównego
- z wyborem elementu głównego

Również trzeba pamiętać, że rozkład *LU* musi być efektywnie pamiętany.

2.2. Analiza rozkładu macierzy *LU*

Rozkład *LU* to dwie macierze, gdzie *L* – to macierz będąca lewą dolną macierzą trójkątną, gdzie na indeksach a_{kk} ma wartości 1, zaś macierz *U* – jest prawą górną macierzą. Iloczyn obu macierzy powinien być wyjściową macierzą, której rozkładu *LU* szukamy:

$$M = LU$$

2.3. Opis algorytmu

Do znalezienia rozkładu LU podanej macierzy można posłużyć się wcześniej zaimplementowaną metodą eliminacji Gaussa. Podczas pierwszej pętli zerującej lewą dolną macierz trójkątną – w metodzie eliminacji Gaussa – po wyzerowaniu macierz staje się szukaną prawą górną macierzą U. Natomiast ze współczynników a_{ij}, które używane są do wyzerowania zawartości komórek, tworzona jest kolejna macierz – używając macierzy jednostkowej – gdzie następnie wypełniana jest odpowiednimi mnożnikami. Tak utworzona macierz jest szukaną macierzą L. Pamięć podczas wyliczeń jest jak najbardziej zminimalizowana ponieważ wykorzystywana jest macierz typu SparseMatrixCSC z biblioteki języka Julia.

$$U = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & 0 & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 & a_{n,n} \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_{2,1} & 1 & 0 & \cdots & 0 \\ a_{3,1} & a_{3,2} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \cdots & 1 \end{pmatrix}$$

3. Zadanie 3

3.1. Opis problemu

Napisać funkcję rozwiązującą układ równań $Ax = b$, gdzie jest uwzględniana specyficzna postać macierzy *A*, za pomocą wcześniej wyznaczonego rozkładu *LU* przez zaimplementowaną funkcję z **Zadanie 2.**

3.2. Analiza problemu

Rozwiązanie równania $LUx = b$, trzeba podzielić na dwa etapy obliczeń. Pierwszy etap to wyznaczenie $Ly = b$, a drugi etap to wyliczenie $y = Ux$.

3.3. Opis rozwiązania

Pierwszy etap to znalezienie *y*, czyli rozwiązanie $Ly = b$. Macierz *L* jest w postaci trójkątnej, więc można w prosty sposób wykonując na układzie *backward substitution*, znaleźć wektor *y*. Po odnalezieniu wektora *y*, należy rozwiązać równanie $Ux = y$, gdzie tutaj wykonujemy *forward substitution*, które daje poszukiwany wektor x.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & 0 & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \vdots & a_{n-1,n} \\ 0 & 0 & 0 & 0 & \vdots & a_{n,n} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

W powyższym układzie nieznane są jedynie współczynniki y_i wektora Y. *Backward substitution* zaczynamy od wyliczenia $y_n = \frac{b_n}{a_{n,n}}$. Wyznaczenie kolejnych współczynników y_k wykonujemy podstawiając do wzoru $y_k = \frac{b_k - \sum_{j=k+1}^{n} a_{k,j} x_j}{a_{k,k}}$, gdzie cofamy się do wcześniejszych wierszy i wyliczamy kolejne y_k korzystając z wcześniejszych wyników.

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_{2,1} & 1 & 0 & \cdots & 0 \\ a_{3,1} & a_{3,2} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}$$

W drugim etapie mamy analogiczną sytuację, choć tutaj występuje macierz trójkątna prawa górną. W tym przypadku użyjemy *forward substitution*. Zaczynamy od x_1, które jest po prostu równe y_1. Kolejne x_i wyliczamy za pomocą takiego oto wzoru $x_k = \frac{y_k - \sum_{j=1}^{k-1} a_{k,j} x_j}{a_{k,k}}$.

4. Opis testów

4.1. Eliminacja Gaussa

Testy zostały przeprowadzone na trzech różnych macierzach A o n równych, 16, 10000 oraz 50000. Rozmiar ich bloku wynosi 4. Testowane macierze spełniają warunki zadania, czyli są blokowe oraz rzadkie. Oczekiwany wynik równań to tablica wektorów x równa lub jak najbliżej oscylująca wartości 1.0. Porównywałem czas oraz potrzebną pamięć do wykonania poszczególnych algorytmów. Testowałem również czy wyniki rzeczywiście oscylują 1.0, pamiętając o błędach numerycznych. Poniżej przedstawiłem wyniki testów. Żaden z algorytmów nie zwrócił wartości, która nie byłaby oczekiwana, czyli działa poprawnie dając poprawne wyniki.

n	Eliminacja Gaussa bez wyboru		Eliminacja Gaussa z wyborem	
	czas	pamięć	czas	pamięć
16	0.000104 s	11.172 KiB	0.000107 s	15.813 KiB
10000	0.219824 s	4.632 MiB	0.285965 s	8.167 MiB
50000	4.770681 s	21.644 MiB	5.621124 s	39.425 MiB

Eliminacja Gaussa z wyborem okazuje się być dużo lepsza, ponieważ czasowo nie jest to duża różnica pomiędzy algorytmami, jedynie można zauważyć, że pamięć rośnie dwukrotnie. Jednakże wyniki są dużo bardziej poprawne, więc poświęcając trochę więcej czasu, oraz dwukrotnie więcej pamięci na obliczenia uzyskujemy dużo bardziej dokładne wyniki.

4.2. Rozkład LU oraz Rozwiązanie LU

n	Rozkład LU bez wyboru		Rozkład LU z wyborem	
	czas	pamięć	czas	pamięć
16	0.000145 s	9.859 KiB	0.000150 s	14.500 KiB
10000	0.281526 s	4.978 MiB	0.392536 s	8.513 MiB
50000	5.147104 s	20.880 MiB	8. 281526 s	38.660 MiB

n	Rozwiązanie LU z wyborem	
	czas	pamięć
16	0.000111s	5.656 KiB
10000	0.033896 s	2.834 MiB
50000	0.189575 s	13.159 MiB

4.3. Wnioski

Poprzez optymalizację eliminacji Gaussa pod konkretny rodzaj macierzy rzadkiej jesteśmy w stanie osiągnąć bardzo dobre wyliczenia w dużo krótszym czasie, dzięki małej złożoności algorytmu w porównaniu do zwykłej eliminacji Gaussa. Jeżeli chodzi o dokładność wszystkich obliczeń to dużo lepszą dokładność otrzymujemy z wyborem. Warto pamiętać, że wybór naprawia możliwie zepsute macierze z powodu zerowych elementów diagonali. Niestety poprzez to zwiększamy zużycie pamięci oraz zwiększamy czas działania, jednak warto dla dużo lepszych wyników. W porównaniu LU, a eliminacja Gaussa, LU wypada lekko lepiej. Czas oraz zużycie pamięci jest bardzo podobne, ponieważ działania obu algorytmów jest bardzo podobne. Ale warto pamiętać, że samo rozwiązanie układu za pomocą LU jest dużo szybsze. Dzięki temu, na niezmiennej macierzy A, wyliczając raz LU można wyliczać wektory x dla różnych wektorów b, co zaoszczędzi nam dużo więcej czasu.