Jan Pułtorak

Pracownia z analizy numerycznej

sprawozdanie do zadania **P2.13**

Prowadzący: mgr Filip Chudy

Wrocław, 21 kwietnia 2023

1. Wstęp

Macierze są fundamentalymi obiektami w algebrze liniowej, reprezentującymi transformacje liniowe między przestrzeniami wektorowymi. Przy ich pomocy można zamodelować wiele zjawisk, między innymi układy równań liniowych. W szczególności, ważnym zagadnieniem jest stwierdzanie czy macierz kwadratowa ma odwrotność, a jeśli tak, to w jaki sposób ją wyznaczyć. Celem niniejszego sprawozdania jest opisanie efektywnego i stabilnego algorytmu odwracającego macierz odwrotną i zaprezentowania go na konkretnych przykładach.

Opisane algorytmy i obliczenia zostały zrealizowane przy użyciu przy użyciu programu *Julia*, a omówione przykłady umieszczone zosały w skrypcie *program.ipynb*.

2. Rozkład LU

2.1. Układy równań liniowych

Odwrotnością do macierzy $A \in GL_n(\mathbb{R})$ nazywamy macierz B spełniającą

$$AB = I := \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \dots & \vdots \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Oznaczając poprzez B_i i-tą kolumnę macierzy B zauważmy, że dla $i \in \{1, 2, \dots, n\}$ musi zachodzić $AB_i = I_i$. Wystarczy więc rozwiązać n układów równań, żeby otrzymać macierz odwrotną. Od teraz będziemy się skupiać na problemie rozwiązywania układu

$$A\boldsymbol{x} = \boldsymbol{b} \tag{2.1.1}$$

2.2. Macierze trójkątne

Rozważymy pewien szczególny rodzaj macierzy, dla których problem (2.1.1) staje się bardzo prosty. Niech $L = [l_{ij}]_{n\times n}$, będzie macierzą, taką że $l_{ij} = 0$ dla $j > i$. Podobnie, niech $U =$

$[u_{ij}]_{n\times n}$, będzie macierzą, taką że $u_{ij} = 0$ dla $j < i$. Ten rodzaj macierzy nazywamy odpowiednio *dolnotrójkątnymi* i *górnotrójkątnymi*.

$$L\boldsymbol{x} = \boldsymbol{b}$$

Jest równoważne z równaniem macierzowym

$$\begin{bmatrix} l_{11} & & & & 0 \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Wyznaczamy x_1:

$$l_{11}x_1 = b_1 \implies x_1 = \frac{b_1}{l_{11}}$$

Załóżmy, że zostało już obliczone $x_1, x_2, \dots x_{k-1}$. Chcemy obliczyć x_k.

$$\begin{aligned} \sum_{i=1}^{k} l_{k,i} \cdot x_i &= b_k \\ l_{k,k} \cdot x_k + \sum_{i=1}^{k-1} l_{k,i} \cdot x_i &= b_k \\ x_k &= \frac{b_k - \sum_{i=1}^{k-1} l_{k,i} \cdot x_i}{l_{k,k}} \end{aligned}$$

Z tego wynika, że bardzo łatwo możemy wyznaczać kolejne x_i znając poprzednie wyrazy. W ten sposób możemy rozwiązać układ w czasie $O(n^2)$. Zaczynając od x_n i postępując analogicznie jak w powyższym przypadku, możemy łatwo rozwiązać układ $Ux = b$.

2.3. Rozkład LU

Załóżmy, że jesteśmy w stanie zapisać macierz A w postaci $A = LU$. Wtedy układ $A\boldsymbol{x} = \boldsymbol{b}$ jest równoważny z układem $LU\boldsymbol{x} = \boldsymbol{b}$. Przyjmijmy $\boldsymbol{y} := U\boldsymbol{x}$. Na podstawie rozważań w sekcji (2.2) jesteśmy w stanie wyznaczyć z łatwością $\boldsymbol{y}$ rozwiązując układ $L\boldsymbol{y} = b$. Następnie z $U\boldsymbol{x} = \boldsymbol{y}$ możemy równie łatwo obliczyć $\boldsymbol{x}$. Zajmiemy się wyznaczeniem takiego rozkładu. Musi zachodzić

$$A = LU$$

Możemy wymnożyć macierze i zobaczyć jakie warunki muszą zostać spełnione.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & \dots & a_{1,n} \\ a_{21} & a_{22} & \dots & \dots & a_{2,n} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & & 0 \\ l_{21} & l_{22} & & & \\ l_{31} & l_{32} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1,n} \\ & u_{22} & u_{23} & \dots & u_{2,n} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ 0 & & & & u_{n,n} \end{bmatrix}$$

$$a_{i,j} = \sum_{k=1}^{n} l_{i,k} \cdot u_{k,j} = \sum_{k=1}^{min(i,j)} l_{i,k} \cdot u_{k,j} \tag{2.3.1}$$

Warto zaznaczyć, co widać z powyższego wzoru, że rozkład LU niekoniecznie musi istnieć. Na ten moment zakładamy, że w żadnym kroku nie występuje dzielenie przez 0 (co jest równoważne z powodzeniem rozkładu).

$$a_{1,j} = l_{1,1} \cdot u_{1,j}$$
$$a_{i,1} = l_{i,1} \cdot u_{1,1}$$

Zauważmy, że $l_{i,1}$ lub $u_{1,j}$ jednoznacznie określa resztę wartości. To znaczy, że jeśli ustalimy z góry jakąś wartość w pierwszej kolumnie L lub w pierwszym wierszu U, to reszta kolumny/wiersza też zostanie wyznaczona.

Załóżmy teraz, że $k-1$ wierszy U i kolumn L zostało już wyznaczonych. Dla $i \geqslant k$

$$a_{i,k} = \sum_{s=1}^{k} l_{i,s} \cdot u_{s,k} = \sum_{s=1}^{k-1} l_{i,s} \cdot u_{s,k} + l_{i,k} \cdot u_{k,k}$$
$$a_{k,i} = \sum_{s=1}^{k} l_{k,s} \cdot u_{s,i} = \sum_{s=1}^{k-1} l_{k,s} \cdot u_{s,i} + l_{k,k} \cdot u_{k,i}$$

Przekształcając równości otrzymujemy

$$a_{i,k} - \sum_{s=1}^{k-1} l_{i,s} \cdot u_{s,k} = l_{i,k} \cdot u_{k,k} \tag{2.3.2}$$
$$a_{k,i} - \sum_{s=1}^{k-1} l_{k,s} \cdot u_{s,i} = l_{k,k} \cdot u_{k,i} \tag{2.3.3}$$

Lewe strony równości, na mocy założenia, są znane. Onacza, to że wartość $l_{i,k}$ określa jednoznacznie resztę elementów w tej kolumnie. Również wartość $u_{k,i}$ określa jednoznacznie wartość reszty elementów w tym wierszu. Ponadto na podstawie $l_{i,k}$ można wyznaczyć $u_{k,k}$. Wnioskujemy stąd, że znajomość dowolnego elementu z k-tej kolumny L lub k-tego wiersza, wyznacza jednoznacznie resztę elementów w tej kolumnie/wierszu. Ustalając z góry n wartości, jesteśmy w stanie, o ile istnieje, łatwo wyznaczyć rozkład LU korzystając z (2.3.2) i (2.3.3). Przykładowo, dla $l_{i,i} = 1$ rozkład ten jest znany jako rozkład Doolittle'a.

2.3.1. Problemy

Rozumowanie przedstawione w poprzednim podrozdziale owocuje stosunkowo prostym sposobem na obliczenie rozkładu LU, lecz jest jeden oczywisty problem, który nie został dokładnie przeanalizowany. Nie wiadomo, czy rozkład LU w ogóle istnieje, a jeśli istnieje, to jak dobrać początkowe n współczynników, żeby przy obliczaniu kolejnych elementów korzystając z (2.3.2) lub (2.3.3) nie podzielić przez zero.

Rozważmy macierz $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right]$. Pokażemy, że nie istnieje rozkład LU. Poniżej układ równań wynikający z (2.3.1)

$$0 = l_{11} \cdot u_{11}$$
$$1 = l_{11} \cdot u_{22}$$
$$1 = l_{21} \cdot u_{11}$$
$$1 = l_{21} \cdot u_{12} + l_{22} \cdot u_{22}$$

Układ ten jest oczywiście sprzeczy, co oznacza, że dla tej macierzy nie istnieje żaden rozkład LU

3. Rozkład PLU

Jak wynika z poprzedniego rozdziału, rozkład LU nie zawsze istnieje. Okazuje się jednak, że dla dowolnej macierzy możemy spermutować jej wiersze tak, żeby rozkład LU istniał. Jeśli zachodzi równość

$$PA = LU$$

to zachodzą równoważności

$$A\boldsymbol{x} = \boldsymbol{b} \iff PA\boldsymbol{x} = P\boldsymbol{b} \iff LU\boldsymbol{x} = P\boldsymbol{b}$$

metoda na rozwiązywanie układu równań wynikającego z ostatniej równości, została opisana na początku rozdziału (2.3). Oznacza to, że znalezienie rozkładu z macierzą permutacji również daje prosty sposób na rozwiązywanie układów liniowych.

3.1. Eliminacja Gaussa

Następująca procedura jest równoważna rozkładowi opisanemu w (2.3) (czyli nie działa w ogólnym przypadku), lecz stosunkowo łatwa modyfikacja pozwoli osiągnąć cel jakim jest rozkład PLU. W k-tym kroku zerujemy wszystkie elementy k-tej kolumny pod główną przekątną. Niech $A^{(k)}$ oznacza macierz uzyskaną po k krokach. Zaczynając od pierwszego wiersza macierzy

$$A^{(1)} = \begin{bmatrix} a_{11} & a_{12} & \dots & \dots & a_{1,n} \\ a_{21} & a_{22} & \dots & \dots & a_{2,n} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & \dots & a_{nn} \end{bmatrix}$$

Niech $l_{i1} = \frac{a_{i1}}{a_{11}}$

$$A^{(2)} = \begin{bmatrix} a_{11} & a_{12} & \dots & \dots & a_{1,n} \\ 0 & a_{22} - l_{21} * a_{22} & \dots & \dots & a_{2,n} - l_{21} * a_{2,n} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & a_{n,2} - l_{n,1} * a_{n,2} & \dots & \dots & a_{n,n} - l_{n,1} * a_{n,n} \end{bmatrix} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & \dots & a_{1,n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & \dots & a_{2,n}^{(2)} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & a_{n,2}^{(2)} & \dots & \dots & a_{nn}^{(2)} \end{bmatrix}$$

Załóżmy, że $k-1$ kroków procedury zostało wykonanych. Od wierszy $k+1, k+2, \dots n$ odejmujemy wielokrotność k-tego wiersza tak, żeby elementy w k-ej kolumnie pod $a_{kk}^{(k)}$ się wyzerowały. Przyjmujemy $l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$ dla $i > k$. Oznacza to, że elementy macierzy $A^{(k+1)}$ są następujące:

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & \text{dla } i \leqslant k, \\ 0 & \text{dla } i \geqslant k+1, j \leqslant k \\ a_{ij}^{(k-1)} - l_{ik} a_{ij}^{(k-1)} & \text{dla } i \geqslant k+1, j \geqslant k+1 \end{cases}$$

Po $n-1$ krokach otrzymana macierz jest górnotrójkątna. Przyjmijmy $U := A^{(n)}$, oraz $L = [l_{ij}]_{n \times n}$. Gdzie l_{ij} zdefiniowane jest jak poprzednio, tzn.

$$l_{ij} = \begin{cases} 0 & \text{dla } j > i, \\ 1 & \text{dla } i = j \\ \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} & \text{dla } i > j \end{cases}$$

Przy takim doborze L i U, zachodzi równość $A = LU$, o czym można się przekonać badając iloczyn LU według powyższych definicji.

3.2. Eliminacja Gaussa z wyborem wierszy głównych

Jeśli dla pewnego k, $a_{kk}^{(k)} = 0$, opisana powyżej procedura zawiedzie. Można ten problem jendak łatwo rozwiązać. W k-tym kroku możemy wybrać i-ty ($i > k$) wiersz taki że, zachodzi $a_{ik}^{(k)} \neq 0$ i zamienić te wiersze miejscami, po czym kontynuować procedurę. I-ty wiersz nazywamy wtedy *wierszem głównym*, a $a_{ik}^{(k)}$ elementem głównym a zamienione wiersze zapisujemy w macierzy permutacji P. Jeśli żaden wiersz nie spełnia tego wymagania, to znaczy że wszystkie są wyzerowane, można więc przejść $k+1$-ego kroku algorytmu.

Korzystając z tej zmodyfikowanej wersji eliminacji Gaussa dla dowolnej macierzy, otrzymamy rozkład $PA = LU$. Dowód poprawności algorymtu został opisany bardziej szczegółowo w [1, str. 167]

3.3. Wybór wierszy głównych

W teorii wybór wiersza głównego nie ma znaczenia, o ile element główny jest niezerowy. W praktyce dobra strategia doboru wierszy głównych może drastycznie wpłynąć na stabliność algorytmu. W skrypcie *program.jl* został zaimplementowany skalowany wybór wierszy głównych. Dodatowko, można wykorzystać to, że macierz L można zapisywać na miejscu zerowanych elementów w A i tym sposobem wynik rozkładu przechować w jednej macierzy zamiast dwóch (kosztem oryginalnej macierzy). Ponadto nie zmieniamy miejscami wierszy macierzy A w pamięci, lecz tworzymy tablicę permutacji, w której zapisujemy zmiany wierszy.

4. Test algorymtu

W programie *program.jl* zostały zaimplementowane dwie omówione wersje eliminacji Gaussa, ze skalowanym wyborem wierszy głównych i bez. Testy zostały przeprowadzone na ręcznie dobranej macierzy, macierzach **Hilberta** $H_n = [h_{ij}] \in ML_n(\mathbb{R})$

$$h_{ij} := \frac{1}{i+j-1}$$

macierzach **Pei**

$$P_{n,d} := \begin{bmatrix} d & 1 & \dots & \dots & 1 \\ 1 & d & \dots & \dots & 1 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 1 & 1 & \dots & \dots & d \end{bmatrix}$$

W ramach kontroli wyników, dla każdej macierzy testowej A podano maksymalny element macierzy $AB - I$ lub $BA - I$ ($B = fl(A^{-1})$). Tę wartość oznaczymy poprzez R(A).

4.1. Test

Następujący przykłady ma na celu pokazanie, czemu implementacja eliminacji Gaussa bez wyboru wierszy głównych jest złym pomysłem. Rozważmy macierz

$$A := \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix}$$

Gdzie $\epsilon = 10^{-19}$. Wtedy

$$A^{-1} = \frac{1}{1-\epsilon} \begin{bmatrix} -1 & 1 \\ 1 & -\epsilon \end{bmatrix}$$

Algrotym z wyborem wierszy głównych daje poprawną odpowiedź, a w wyniku zwykłej eliminacji Gaussa otrzymujemy

$$fl(A^{-1}) = \begin{bmatrix} 0 & 1 \\ 1 & -\epsilon \end{bmatrix}$$

Przy zastosowaniu eliminacji bez wyboru czynników głównych otrzymujemy następujący rozkład

$$A = \begin{bmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & 1-\epsilon^{-1} \end{bmatrix}$$

Chcąc znaleźć pierwszą kolumnę macierzy odwrotnej rozwiązujemy podstawianiem w tył/przód układ równań

$$\begin{bmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & 1-\epsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

I otrzymujemy

$$x_2 = \epsilon^{-1}(1-\epsilon^{-1})^{-1}$$

Dla małego ϵ w arytmetyce maszynowej $(1-\epsilon^{-1})^{-1} = \epsilon^{-1}$, stąd $x_2 \approx 1$. Z kolei

$$x_1 = \frac{1-x_2}{\epsilon} \approx 0$$

W skutek przybliżeń podczas dzielenia przez bardzo małe wartości, algorytm prowadzi do mocno nieprecyzyjnych wyników. Algorytm z wyborem wierszy głównych zamieni wiersze miejscami i wtedy dokona rozkładu LU.

$$\begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-\epsilon \end{bmatrix}$$

W tym przypadku nie pojawiają się żadne problematyczne operacje, więc otrzymana macierz odwrotna będzie obliczona z dużo większą precyzją.

4.2. macierze Pei

Testy podzielone są względem parametru d.

n	d	$R(P_{n,d})$
100	2	1.4654943925052066e-14
200	2	5.129230373768223e-14
300	2	1.4249712521063884e-13
400	2	1.7724710588140624e-13
500	2	1.8851586958135158e-13
600	2	4.156675004196586e-13
700	2	7.687184222504584e-13
800	2	7.303324611740436e-13
900	2	1.2645301472602455e-12
1000	2	1.170619157164765e-12

(a) duże wartości n

n	d	$R(P_{n,d})$
3	$1 + 10^{-5}$	1.4551915228366852e-11
6	$1 + 10^{-5}$	1.5232926031671923e-11
9	$1 + 10^{-5}$	7.09405867382884e-11
12	$1 + 10^{-5}$	3.2741809263825417e-11
3	$1 + 10^{-12}$	0.0001220703125
6	$1 + 10^{-12}$	0.000274658203125
9	$1 + 10^{-12}$	0.00018310546875
12	$1 + 10^{-12}$	0.0012054443359375
9	$1 + 10^{-15}$	0.375
12	$1 + 10^{-15}$	0.921875

(b) Wartości d zbliżone do 1

Tabela 1: Wartości $R(P_{n,d})$ dla różnych wartości n i d

4.3. macierze Hilberta

Poniższe tabele przedstawiają wartości $R(H_n)$ otrzymane kolejno przy użyciu funkcji **inverse** z *program.jl* i funkcji bibliotecznej **inv** z programu *Julia*

n	$R(H_n)$
2	0.0
4	2.2737367544323206e-13
6	2.9050325108380774e-10
8	7.152557373046875e-7
10	0.0014044910684864705
12	1.6429351532121341
14	214.45860104634392
16	925.556640625

(a) implementacja własna

n	$R(H_n)$
2	0.0
4	4.051996833303335e-13
6	1.9727101433659785e-10
8	5.215406417846668e-7
10	0.0005514722872238115
12	1.4664803307934204
14	141.4375
16	6357.0

(b) funkcja **inv** z biblioteki standardowej

Tabela 2: wartość $R(H_n)$ dla różnych n

5. Wnioski

Uzyskany algorytm daje dobre wyniki dla macierzy, dobrze uwarunkowanych (np. macierz Pei dla $d >> 1$). Z kolei, odwracanie macierzy Pei dla $d \approx 1$ i macierzy Hilberta dla $n > 10$ dało nieprecyzyjne wyniki. Jak pokazuje Tabela 2, nie jest to kwestia implementacji. Jest tak, ponieważ macierze te są źle uwarunkowane, co oznacza, że obliczając ich odwrotności trzeba znaleźć metody specificzne dla danej macierzy, lub liczyć się z tym, że otrzymany wynik nie będzie dokładny. O ile więc nie jest to konieczne, macierzy nie należy odwracać, w szczególności po to żeby rozwiązywać układy równań, ponieważ jak pokazano, mając rozkład LU, jest się w stanie robić to efektywnie ($O(n^2)$), bez znajdowania odwrotności.

Literatura

[1] David Kincaid, Ward Cheney (2006), *Analiza numeryczna*