MetodyNumeryczne - Projekt 2

Krzysztof Sawicki

Styczeń 2023

Celem projektu jest rowiązywanie równania macierzowego $AX = B$, gdzie $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$, $m \geq 1$ metodą Crouta oraz obliczenie $det(A)$ na podstawie wyznaczonego rozkładu.

Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Spis treści

1 Rozkład Crouta

1.1 Opis matematyczny

Rozkład Crouta to rozkład macierzy na czynniki w którym macierz A zapisuje się jako iloczyn macierzy trójkątnej dolnej L oraz trójkątnej górnej U, przy czym na głównej przekątnej U znajdują się wyłącznie jedynki.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} * \begin{bmatrix} 1 & u_{12} & \dots & u_{1n} \\ 0 & 1 & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

1.2 Implementacja w języku Matlab

Funkcja crout(A) przyjmuje macierz kwadratową A i zwraca macierze L i U pochodzące z rokładu Crouta macierzy A.

```
function [L, U] = crout(A)

%Funkcja crout(A) sluzy znajdowaniu rozk adu crouta macierzy A
%[L, U] = crout(A) zwraca macierze L i U takie, ze L*U = A gdzie
%A: macierz kwadratowa
%L: macierz trojkatna dolna
%U: macierz trojkatna gorna z jedynkami na glownej przekatnej

sizeA = size(A);
if sizeA(1) ≠ sizeA(2)
    warning("Macierz nie jest kwadratowa")
end

n = sizeA(1);
U = eye(n);
L = zeros(n,n);
L(:,1) = A(:,1);
U(1, :) = A(1, :) / L(1, 1);

for i = 2:n
    for j = 2:i
        L(i, j) = A(i, j) - L(i, 1:j - 1) * U(1:j - 1, j);
    end

    for j = i + 1:n
        U(i, j) = (A(i, j) - L(i, 1:i - 1) * U(1:i - 1, j)) / ...
            L(i, i);
    end
end

end
```

1.3 Prezentacja działania implementacji

W celu zaprezentowania działania metody weźmy macierz:

$$A = \begin{bmatrix} 1 & 4 & 1 & 0 \\ 1 & 9 & 4 & 5 \\ 1 & 9 & 1 & 8 \\ 1 & 9 & 2 & 0 \end{bmatrix}$$

Rozkład crouta dla tej macierzy wygląda następująco:

```
>> crout(A)

L =

     1     0     0     0
     1     5     0     0
     1     5    -3     0
     1     5    -2    -7


U =

    1.0000    4.0000    1.0000         0
         0    1.0000    0.6000    1.0000
         0         0    1.0000   -1.0000
         0         0         0    1.0000
```

Rzeczywiście macierze L i U spełaniją założenia rozkładu crouta oraz $A = LU$

```
>> L*U

ans =

     1     4     1     0
     1     9     4     5
     1     9     1     8
     1     9     2     0

>> L*U == A

ans =

  4 4  logical array

   1   1   1   1
   1   1   1   1
   1   1   1   1
   1   1   1   1
```

1.4 Ograniczenia metody

Nie wszystkie macierze kwadratowe dają się rozłożyć za pomocą rozkładu crouta. Weźmy macierz A i spróbujmy rozłożyć ją na macierz L i U:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} * \begin{bmatrix} 1 & u_{12} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} l_{11} & l_{11} * u_{12} \\ l_{21} & l_{21} * u_{12} + l_{22} \end{bmatrix}$$

Stąd otrzymujemy, że $l_{11} = 0$ oraz $l_{11} * u_{12} = 0$ co sprzeczne. Gdy spróbujemy macierz A podać funkcji crout otrzymujemy:

```
>> crout(A)

L =

     0     0
     1  -Inf


U =

   NaN   Inf
     0     1
```

2 Obliczanie wyznacznika macierzy za pomocą rozkładu crouta

2.1 Opis matematyczny

Przy obliczaniu wyznacznika macierzy A, dzięki rozkładowi crouta, przedstawiamy macierz A jako iloczyn macierzy L i U, a następnie korzystamy z własności wyznacznika otrzymując:

$$det(A) = det(L) * det(U)$$

Zauważmy, że macierze L i U to macierze trójkątne zatem ich wyznacznik to iloczyn elementów na głównej przekątnej zatem:

$$det(A) = det(L) * det(U) = \Pi_{i=1}^{n} l_{ii} * \Pi_{i=1}^{n} u_{ii}$$

jednakże w rozkładzie crouta macierz U na głównej przekątnej posiada same jedynki, więc $det(U) = 1$. Z tego powodu równananie możemy uprościć do:

$$det(A) = det(L) = \Pi_{i=1}^{n} l_{ii}$$

2.2 Implementacja w języku Matlab

```
function [res] = detCrout(A)

%Funkcja detCrout(A) sluzy do obliczenia wyznacznika macierzy A
%wykorzystujac rozklad crouta macierzy A (LU = A)
%Funkcja korzysta z wlasnosci wyznacznika det(A) = det(L)*det(U)
%oraz tego,  e  L i U to macierze trojk tne zatem ich ...
    wyznacznik to iloczyn
%elementow na przekatnej. Dodatkowo macierz U na glownej ...
    przekatnej ma same
%1 zatem jej wyznacznik jest r wny 1

[L,U] = crout(A);
n = size(L,1);

%obliczamy wyznacznik det(L), det(U) jest pomijany gdyz jest ...
    r wny 1
res = L(1,1);

for i = 2:n
    res = res*L(i,i);
end

end
```

2.3 Prezentacja działania implementacji

Weźmy wcześniej rozważaną macierz A i obliczmy jej wyznacznik za pomocą funkcji detCrout:

$$A = \begin{bmatrix} 1 & 4 & 1 & 0 \\ 1 & 9 & 4 & 5 \\ 1 & 9 & 1 & 8 \\ 1 & 9 & 2 & 0 \end{bmatrix}$$

```
>> detCrout(A)

ans =

   105
```

Porównajmy wynik z wbudowaną funkcją det:

```
>> det(A)

ans =

   105
```

2.4 Ograniczenia metody

Metoda oczywiście nie zadziała, gdy weźmiemy macierz która nie posiada rozkładu crouta. Zatem dla wcześniej rozpatrywanej macierzy:

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

```
>> detCrout(A)

ans =

   NaN
```

Porównując wynik z wbudowaną funkcją det:

```
>> det(A)

ans =

    -1
```

3 Rozwiązywanie równania AX = B metodą Crouta

3.1 Opis matematyczny metody

W celu rozwiązania równania $AX = B$ macierz A, przy pomocy metody crouta, rozkładamy na iloczyn macierzy L i U otrzymując równanie:

$$LUX = B$$

Następnie stosujemy podstawienie $Y = UX$ otrzymując równanie $LY = B$. Dzięki temu otrzymujemy dwa równania $LY = B$ oraz $UX = Y$, w których macierze L i U są macierzami trójkątnymi co znacznie upraszcza rozwiązywanie takiego układu. Po rozwiązaniu $LY = B$ rozwiązujemy $UX = Y$ otrzymując szukanego X.

3.2 Implementacja w języku Matlab

Funkcja solve_Crout wykorzystuje funkcje pomocnicze do rozwiązywania układu równań, gdzie macierz jest trójkątna. Poniżej znajdują się implementacje tych funkcji.

```
function [X] = solve_Crout(A,B)

%Funkcja solve_Crout(A,b) s u y do rozwi zywania r wnania ...
    macierzowego
%AX = B, gdzie A     R^nxn, B     R^nxm metod  Crouta.
%Funkcja korzysta z rozk adu Crouta dla macierzy A otrzymuj c ...
    uk ad  r wna
%LUX = B. Nast pnie podstawia Y za UX i rowi zuje uk ad ...
    r  w n a   LY = B, aby
%w ko cowym rozrachunku r o z w i  z a   uk ad UX = Y i  z w r  c i   ...
    wynikowy X,  kt ry
%spe nia pierwotne r wnanie AX = B.

sizeA = size(A);
sizeB = size(B);

% sprawdzanie, czy macierze A i B  s   poprawnych wymiar w

if sizeA(1) ≠ sizeA(2)
    ME = MException("solve_Crout:wrongInput", "Macierz A nie ...
        jest kwadratowa");
    throw(ME)
end
if sizeA(2) ≠ sizeB(1)
    ME = MException("solve_Crout:wrongInput", "Macierze A i B ...
        m a j  nieodpowiednie wymiary");
    throw(ME)
end

[L,U] = crout(A);
```

```
Y = lower_triangular_solver(L,B);
X = upper_triangular_solver(U,Y);

end
```

```
function [X] = upper_triangular_solver(A, B)

%Funkcja upper_triangular_solver rozwiazuje rownanie macierzowe ...
    AX=B, gdzie A
%jest macierza gornotrojkatna.
%Funkcja przyjmuje:
%Kwadratowa macierz A o wymiarach nxn
%Macierz B o wymiarach nxm
%Funkcja zwraca:
%Macierz X spelniajaca rownanie AX = B

n = size(A,1);
m = size(B,2);
X = zeros(n, m);

for i = n:-1:1

    v = n:-1:i+1;
    X(i,:) = (B(i,:) - A(i, v)*X(v,:)) / A(i,i);

end

end
```

```
function [Y] = lower_triangular_solver(A, B)

%Funkcja lower_triangular_solver rozwiazuje rownanie macierzowe ...
    AX=B, gdzie A
%jest macierza dolnotrojkatna.
%Funkcja przyjmuje:
%Kwadratowa macierz A o wymiarach nxn
%Macierz B o wymiarach nxm
%Funkcja zwraca:
%Macierz X spelniajaca rownanie AX = B

n = size(A,1);
m = size(B,2);
Y = zeros(n, m);

for i = 1:n

    v = 1:i-1;
    Y(i,:) = (B(i,:) - A(i, v)*Y(v,:)) / A(i,i);

end

end
```

3.3 Prezentacja działania implementacji

W celu prezentacji rozpatrzmy układ równań AX=B:

$$\begin{bmatrix} 1 & 4 & 1 & 0 \\ 1 & 9 & 4 & 5 \\ 1 & 9 & 1 & 8 \\ 1 & 9 & 2 & 0 \end{bmatrix} * \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 1 & 0 \\ 1 & 2 \\ 5 & 3 \end{bmatrix}$$

```
>> X = solve_Crout(A,B)

X =

   -0.2857    1.3810
    0.7143    0.3810
   -0.5714   -0.9048
   -0.5714   -0.2381

>> A*X

ans =

    2.0000    2.0000
    1.0000   -0.0000
    1.0000    2.0000
    5.0000    3.0000
```

Zatem X równa się:

$$\begin{bmatrix} -0.2857 & 1.380 \\ 0.7143 & 0.3810 \\ -0.5714 & -0.9048 \\ -0.5714 & -0.2381 \end{bmatrix}$$

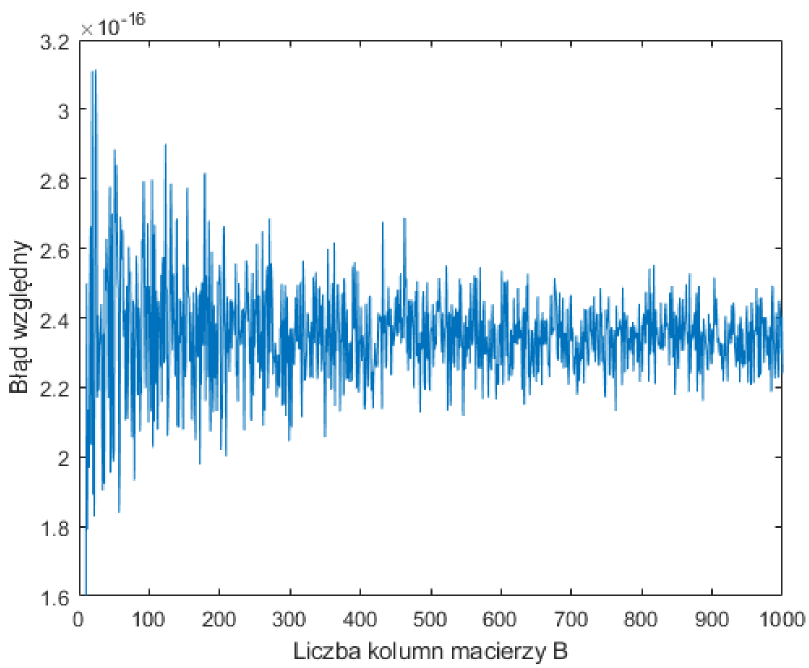
4 Przykłady

Przykład 1

Rozpatrzymy jak metoda crouta radzi sobie z układami równań $AX = B$ w których A to macierz Wilkinsona (100×100). Dla takiej macierzy:

- wskaźnik uwarunkowania macierzy $cond(A) = 236.2716$
- błąd rozkładu przy wykorzystaniu metody crouta wynosi $e_{dec} = 2.2096e-18$
- wyznacznik wyznaczony przy pomocy rozkładu crouta wynosi $-3.9880e+126$
- wyznacznik wyznaczony przy pomocy wbudowanej funkcji *det* również wynosi -3.9880e+126

Sprawdźmy jak zmienia się błąd względny w zależności od ilości kolumn w macierzy B. Przy tej analizie macierz B posiadała 100 wierszy oraz składała się z losowych liczb naturalnych z mniejszych bądź równych 10.

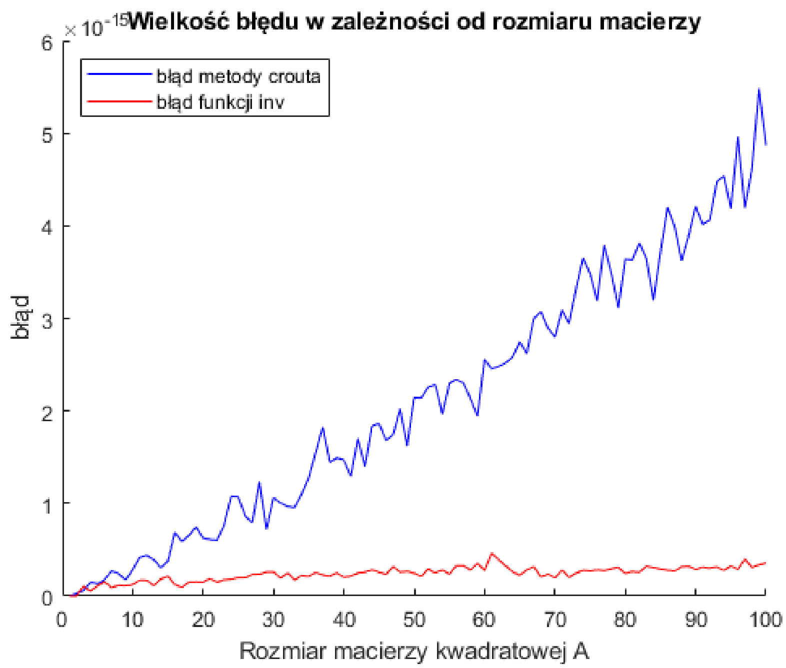


Rysunek 1: Wykres zależności błędu względnego od ilości kolumn w macierzy B

Jak możemy zauważyć im więcej kolumn posiada macierz tym mniejsze wahania błędu względnego wynikającego z rozwiązania równiania $AX = B$.Dla wiekszej ilości kolumn błąd ten zawiera się w przedziale $(2.2 \times 10^{-16}, 2.6 \times 10^{-16})$.

Przykład 2

Zbadajmy jak metoda crouta radzi sobie ze znajdowaniem macierzy odwrotnej do A czyli rozwiązywaniem układu $AX = B$, gdzie B to macierz jednostkowa. W celu analizy za macierz A przyjmijmy macierze typu Circulant.



Rysunek 2: Wykres zależności błędu od rozmiaru macierzy A

Jak możemy zauważyć metoda crouta poradziła sobie gorzej niż wbudowana funkcja inv().

Przykład 3

Rozkład Crouta nie zadziała dla macierzy o bardzo dużym uwarunkowaniu. Przykładowo dla równania:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} * X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Otrzymujemy następujące podsumowanie:

Uwarunkowanie	Wyznacznik	Wyznacznik (detCrout)
$1.4155e+17$	$6.2123e-30$	NaN

Błąd rozkładu	Błąd względny	Wsp stabilności	Wsp poprawności
NaN	NaN	NaN	NaN

Dla wcześniej badanej macierzy:

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

Również nie otrzymamy macierzy odwrotnej pomimo niskiego uwarunkowania.

Uwarunkowanie	Wyznacznik	Wyznacznik (detCrout)
2.618	-1	NaN

Błąd rozkładu	Błąd względny	Wsp stabilności	Wsp poprawności
NaN	NaN	NaN	NaN

Przykład 4

W tym przykładzie sprawdźmy jak Crout radzi sobie magią. Aby przestestować jego magiczne zdolności poddajemy go próbie rozwiązania układu równań $AX = B$, gdzie A to macierz magiczna, w której komórki wpisano liczby w ten sposób, że ich suma w każdym wierszu, w każdej kolumnie i w każdej przekątnej jest taka sama, a macierz B to macierz typu Circulant. Wyniki przedstawiają się następująco:

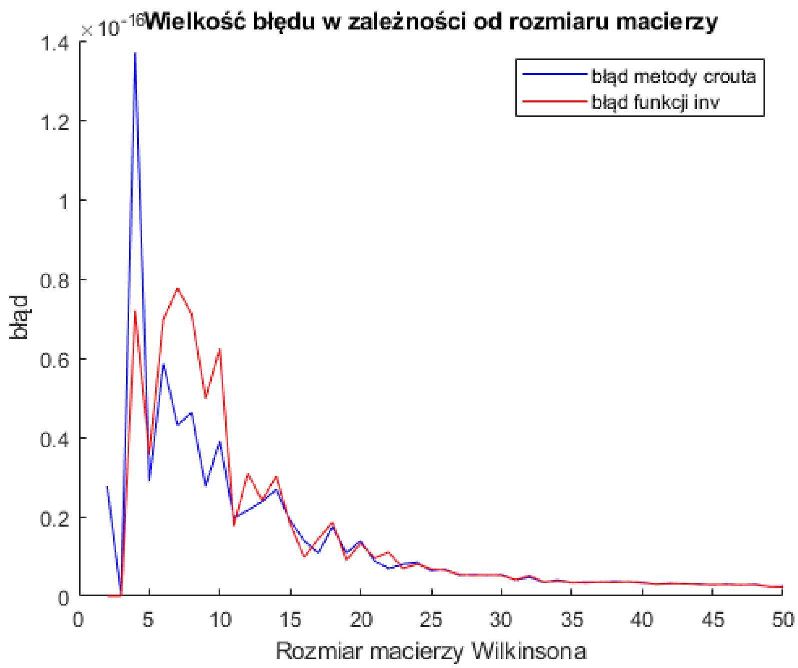
Rozmiar macierzy	Uwarunkowanie	Błąd rozkłądu	Wyznacznik	Wyznacznik (detCrout)
5	5.4618	$5.07e+06$	$5.07e+06$	$1.0931e-16$
6	$4.7002e+16$	$7.8403e-17$	$-3.4498e-09$	$2.7598e-08$
7	7.1113	$1.1187e-16$	$-3.4805e+11$	$-3.4805e+11$
8	$1.3683e+18$	NaN	0	NaN
9	9.1017	$2.7631e-15$	$7.5036e+16$	$7.5036e+16$
10	$7.9107e+17$	$1.3206e-16$	$5.7874e-31$	$-1.2717e-30$

Rozmiar macierzy	Błąd względny	Współczynnik stabilności	Współczynnik poprawności
5	$9.2913e-16$	$1.7011e-16$	$2.3083e-16$
6	1.125	$2.3935e-17$	$4.1477e-17$
7	$4.8945e-15$	$6.8826e-16$	$9.5344e-16$
8	NaN	NaN	NaN
9	$1.2925e-14$	$1.4201e-15$	$2.0527e-15$
10	1.3294	$1.6806e-18$	$5.2237e-17$

Jak możemy zauważyć Crout ptawie poradził sobie zadaniem. Zauważmy, że przy dużym uwarunkowaniu macierzy różnica w obliczonych wyznacznikach jest zauważalna. Dla n = 8 nie udało się znaleźc rozwiązania. Macierz magiczna 8x8 nie ma rozkładu crouta.

Przykład 5

Powracając do macierzy wilkisona sprawdzamy jak metoda crouta radzi sobie z rozwiązywaniem równań $AX = B$, gdzie A to macierz wilkinsona a B to macierz jednostkowa.

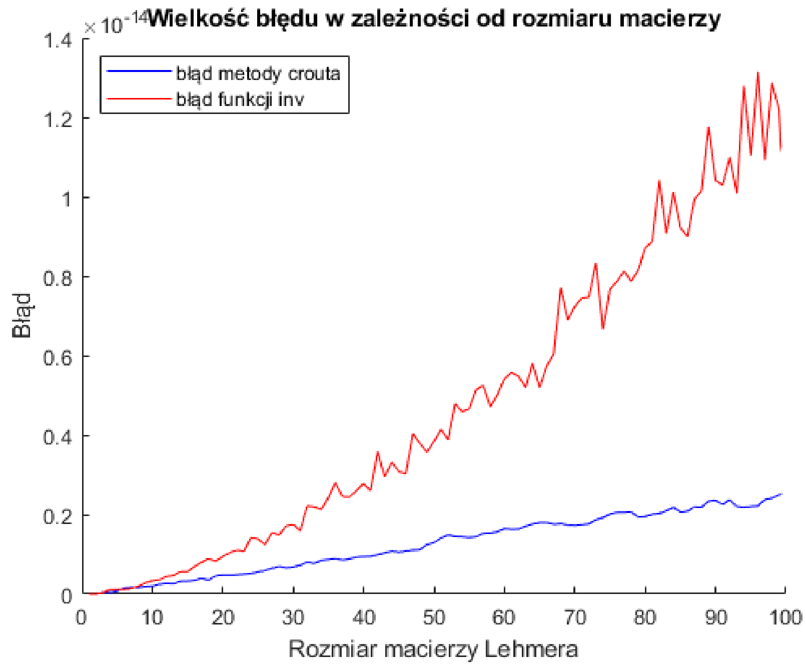


Rysunek 3: Wykres zależności błędu od rozmiaru macierzy Wilkinsona

Jak możemy zauważyć metoda Crouta radzi sobie podobnie jak wbudowana funkcja w matlabie. Im większa macierz A tym bliższe są błędy produkowane przez te metody.

Przykład 6

W ostatnim przykładzie rowiążemy układ $AX = B$, gdzie A to kwadratowa macierz Lehmera natomiast B to macierz jednostkowa.



Rysunek 4: Wykres zależności błędu od rozmiaru macierzy Lehmera

Jest to przykład pokazujący, że w wyszczególnionych przypadkach metoda Crouta radzi sobie zdecydowanie lepiej niż wbudowane funckje w matlabie.