Metody Numeryczne Projekt 2

Wojciech Grabias

Styczeń 2023

Rozwiązywanie równania macierzowego $AX = B$, gdzie $A \in \mathbb{R}^{n \times n}$ - macierz symetryczna i dodatnio określona, $B \in \mathbb{R}^{n \times m}$, $m \geq 1$ metodą Cholesky'ego-Banachiewicza

Spis treści

1 Wstęp teoretyczny

Metoda Cholesky'ego-Banachiewicza w zamyśle dotyczy rozkładu macierzy $A \in \mathbb{R}^{n \times n}$, A - dodatnio określona i symetryczna na iloczyn macierzy LL^T, przy czym L jest macierzą dolnotrójkątną z dodatnimi współczynnikami na diagonali. Umożliwia to rozwiązywanie równania macierzowego

$$AX = B$$

poprzez zastąpienie A wcześniej ustaloną postacią:

$$LL^T X = B$$

następnie traktując wyrażenie $L^T X$ jako nowo powstałą macierz - Y, otrzymując ostatecznie wyrażenie:

$$LY = B$$

Ponieważ jednak L i L^T są macierzami odpowiednio dolno i górnotrójkątnymi, w znaczny sposób usprawnia to znajdywanie rozwiązań odpowiednich podukładów.
Ze względu na fakt, iż rozważane układy są układami macierzowymi, rozwiązywanie ich traktowane jest jako znajdowanie szeregu rozwiązań liczbowych układów liniowych, których ilość zależna jest od liczby kolumn macierzy B.

2 Implementacja metody

2.1 Rozkład macierzy

Aby dokonać rozkładu macierzy $A \in \mathbb{R}^{n \times n}$ korzystamy z poniższego algorytmu, który jest konieczną konsekwencją podejścia odwrotnego - przy założeniu, że macierze L oraz L^T istnieją współczynniki l_{ij}, $i, j \in [n]$ po wymnożeniu tych macierzy można wyznaczyć w następujący sposób:

for $k = 1, 2, ..., n$

$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$

for $i = k + 1, k + 2, ..., n$

$l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj})/l_{kk}$

end

end

Implementacja w języku MATLab wygląda następująco:

```
1      function [L] = cholesky_decomposition(A)
2
3  % Funkcja cholesky_decomposition() jako argument przyjmuje
4  % macierz A o wspolczynnikach z R, ktora jest jednoczesnie ...
       symetryczna
5  % i dodatnio okreslona.
6
7  % Funkcja zwraca macierz dolnotrojkatna L wedlug dekompozycji
8  % Cholesky'ego-Banachiewicza taka, ze LL^T = A
9
10 n = size(A, 1);
11
12 % Deklarujemy L dla wiekszej efektywnosci algorytmu
13 L = zeros(n);
14
15 % Stosujemy algorytm podany na wykladzie
16 for k = 1:n
17     L(k,k) = sqrt(A(k,k) - sum(L(k,1:k-1).^2));
18     wektor_sum = squeeze(sum(L(k+1:n,1:k-1).*L(k,1:k-1), 2));
19     L(k+1:n,k) = (A(k+1:n,k) - wektor_sum)/L(k,k);
20 end
21 end
```

2.2 Rozwiązywanie układów liniowych dla macierzy dolnotrójkątnej

Zaimplementowana funkcja rozwiązywania układów liniowych uszczególniona została wyłącznie do macierzy dolnotrójkątnej. Ponieważ jednak macierz L^T jest górnotrójkątna, w dalszej części projektu permutowane będą wiersze macierzy wejściowych.

```
1      function [X] = lower_triangular_solver(L, B)
2  % Funkcja lower_triangular_solver(L,B) ma za zadanie zwrocic
3  % macierz X bedaca rozwiazaniem ukladu macierzowego LX=B, przy czym
4  % wymagane jest, by macierz L byla macierza dolnotrojkatna
5
6  % L - Macierz kwadratowa dolnotrojkatna o wymiarach nxn,
7  %     dla ktorej rozwiazane ma byc rownanie LX=B
8  % B - Macierz o wymiarach nxm, bedaca prawa strona rownania
9
10 % Ustalamy rozmiar macierzy wynikowej i deklarujemy ja
11 % w celach przyspieszenia algorytmu
12 n = size(L,1);
13 m = size(B,2);
14
15 X = zeros(n, m);
16 % Wektorowo bierzemy pod uwage odpowiednia dlugosc danego wiersza
17 % tak, by operowac tylko na trojkatnej czesci macierzy
18 for i = 1:n
19     column_vector = 1:i-1;
20
21     % Tworzymy zmienna tymczasowa reprezentujaca wartosc liczbowa
22     % iloczynu L oraz X do elementu na diagonali
23     temp = L(i,column_vector) * X(column_vector,:);
24
25     % Po odjeciu temp od B(i,:) wystarczy podzielic przez ...
           wspolczynnik na
26     % diagonali
27     X(i,:) = (B(i,:) - temp) / L(i,i);
28 end
29 end
```

2.3 Otrzymanie macierzy wynikowej

Do uzyskania macierzy wynikowej wymagane jest skorzystanie z funkcji lower_triangular_solver dwukrotnie. Najpierw, w celu policzenia podukładu $LY = B$, następnie układu $L^T X = Y$. Uważny czytelnik może zauważyć, że macierz L^T nie jest wymaganą macierzą dolnotrójkątną. W tym celu funkcja lower_triangular_solver wywoływana jest na macierzy L, natomiast macierz Y przekazywana jest jako jej permutacja Y' w ten sposób by dla dowolnego $i = 1, ..., n$, $y'_{ij} = y_{(n-i+1)j}$

```
1      function [X] = cholesky(A, B)
2  % Funkcja cholesky(A, B) ma za zadanie zwrocic macierz wynikowa
3  % bedaca rozwiazaniem ukladu macierzowego AX = B korzystajac z ...
       poprzednio
4  % zdefiniowanych funkcji cholesky_decomposition oraz
5  % lower_triangular_solver
6
7  % Rozkladamy macierz A metoda Cholesky'ego-Banachiewicza
8  L = cholesky_decomposition(A);
9
10 % Rozwiazujemy poduklad LY = B
11 Y = lower_triangular_solver(L, B);
12 % Macierz wynikowa permutujemy tak, by i-ty wiersz stal sie ...
       (n-i+1)-tym
13 Y = flip(Y,1);
14
15 % Rozwiazujemy poduklad L^{T}X=Y (argumentem nadal jest L, bo ...
       wymagana jest
16 % macierz dolnotrojkatna, co umozliwia wczesniejsze odwrocenie Y
17 X = lower_triangular_solver(L, Y);
18
19 % Otrzymana macierz wynikowa rowniez ma odwrocone wiersze wzgledem
20 % poprawnego wyniku
21 X = flip(X,1);
22
23 end
```

3 Prezentacja działania implementacji

W poniższych przykładach stosowana będzie następująca notacja:

1. wskaźnik uwarunkwania macierzy A - $cond(A)$
2. błąd rozkładu - e_{dec}
3. błąd względny - e_{rel}
4. współczynnik stablilności - wsp_{stab}
5. współczynnik poprawności - wsp_{popr}

Kluczowym założeniem jest również, że za dokładne rozwiązanie równania macierzowego uznajemy funkcję matlab_inv, która korzysta z wbudowanej w język MATLab funkcji inv():

```
1      function [X] = matlab_inv(A, B)
2
3  % Funkcja matlab_inv oblicza r wnanie macierzowe AX=B za pomoc
4  % wbudowanej w j zyk funkcji inv()
5
6  % A - Macierz odwracalna nxn
7  % B - Macierz rozmiar w nxm
8
9  X = inv(A)*B;
10
11 end
```

3.1 Przykład 1

Jako pierwszy i najbardziej podstawowy przykład roważmy macierz z kartki ćwiczeniowej J (z dokładnością do rozszerzenia wektora b o jedną kolumnę tak, by otrzymać równanie macierzowe):

$$A = \begin{pmatrix} 4 & -2 & -2 \\ -2 & 5 & -1 \\ -2 & -1 & 6 \end{pmatrix}$$

$$B = \begin{pmatrix} -2 & 2 \\ 1 & -1 \\ 9 & -9 \end{pmatrix}$$

```
>>    cholesky(A, B)

ans =

       1    -1
       1    -1
       2    -2
```

Otrzymana macierz dokładnie pokrywa się z wynikiem obliczonym zarówno w ramach zajęć ćwiczeniowych, jak i poprzez funkcję matlab_inv. Tabela wskaźników, błędów i współczynników bez zaskoczeń otrzymuje więc postać:

Tablica wskaźników dla Przykładu 1				
$cond(A)$	e_{dec}	e_{rel}	wsp_{stab}	wsp_{popr}
5.2376	0	1.3914×10^{-17}	0	0

Powyższy przykład potwierdza więc poprawne działanie implementacji numerycznej i na jego podstawie możemy wnioskować, iż dla macierzy niewielkich rozmiarów i o nieróżniących się znacząco współczynnikach, metoda sprawdza się bez wad.

3.2 Przykład 2

Użyjmy tej samej macierzy A, by sprawdzić, czy różnica we współczynnikach macierzy B wpłynie na jakoś wyniku implementacji numerycznej metody Cholsky'ego-Banachiewicza. Ustalmy więc teraz macierz B:

$$B = \begin{pmatrix} 1000 & 10000 & 100000 & 1000000 & 10000000 & 100000000 & 1000000000 \\ 0 & 0.01 & 0.001 & 0.0001 & 0.00001 & 0.000001 & 0.0000001 \\ -10 & -0.001 & -100 & -0.0001 & -1000 & -0.00001 & -10000 \end{pmatrix}$$

```
>> matlab_inv(A,B)

ans =

   1.0e+08 *

    0.0000    0.0000    0.0005    0.0045    0.0453    0.4531    ...
        4.5312
    0.0000    0.0000    0.0002    0.0022    0.0219    0.2187    ...
        2.1875
    0.0000    0.0000    0.0002    0.0019    0.0187    0.1875    ...
        1.8750

>> cholesky(A,B)

ans =

   1.0e+08 *

    0.0000    0.0000    0.0005    0.0045    0.0453    0.4531    ...
        4.5312
    0.0000    0.0000    0.0002    0.0022    0.0219    0.2187    ...
        2.1875
    0.0000    0.0000    0.0002    0.0019    0.0187    0.1875    ...
        1.8750
```

Ponownie otrzymany wynik nie rózni się jednak w ogóle od wyniku przyjętego za dokładny, z dokładnością do wyświetlanych przez konsolę przybliżeń. Spójrzmy więc na tabelę wskaźników błędu dla tego przykładu:

Tablica wskaźników dla Przykładu 2				
$cond(A)$	e_{dec}	e_{rel}	wsp_{stab}	wsp_{popr}
5.2376	0	1.3914×10^{-17}	2.6566×10^{-18}	3.5429×10^{-17}

Otrzymane współczynniki stabilności i poprawności zadania numerycznego są pomijalnie małe mimo, że współczynniki macierzy B różniły się od siebie o kilkanaście rzędów wielkości. Burzy to więc teorię, że duża róznica w wartościach współczynników negatywnie wpływa na jakość wyniku zastosowanego algorytmu.

3.3 Przykład 3

Przykład 3 poświęcony będzie zbadaniu wpływu rozmiaru macierzy B na poprawność i stabilność zadania numerycznego realizowanego implementowany algorytmem. Rozważmy więc pewną losowo wygenerowaną macierz B o rozmiarach 3×1000000, o współczynnikach z zakresu $[0, 100]$ rozlosowanych z rozkładu jednostajnego:

```
1       >> a = 1;
2       >> b = 100;
3       >> B = a + (b-a) * rand(3, 1000000);
```

Korzystając ponownie z macierzy A z Przykładu 1, zbadajmy wskaźniki dla implementowanego algorytmu:

Tablica wskaźników dla Przykładu 3.1				
$cond(A)$	e_{dec}	e_{rel}	wsp_{stab}	wsp_{popr}
5.2376	0	7.6287×10^{-17}	1.4565×10^{-17}	3.4945×10^{-17}

Po raz kolejny rozmiary współczynników są pomijalnie małe, co burzy również teorię o wpływie ilości kolumn na poprawność i stabilność zaimplementowanego zadania numerycznego.
Rozważmy jednak macierz B o większej ilości wierszów, konkretniej o wymiarach 1500×1000. By zbadać pozostałe współczynniki macierz A będzie losowo wygenerowaną macierzą symetryczną i dodatnio określoną, zaś jej wskaźnik uwarunkowania zbliżony będzie do 1.

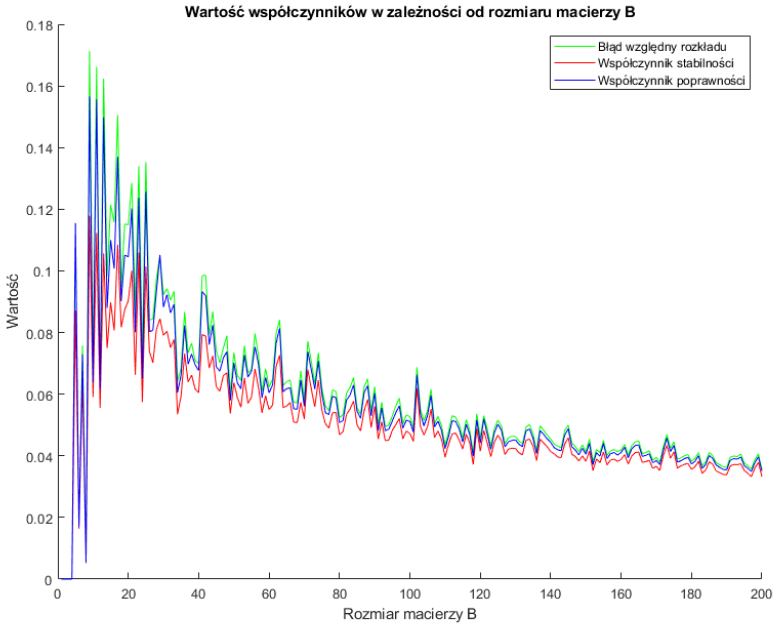
```
1       >> B = a + (b-a) * rand(1500, 1000);
```

Tablica wskaźników dla Przykładu 3.2				
$cond(A)$	e_{dec}	e_{rel}	wsp_{stab}	wsp_{popr}
1.1083	3.2113×10^{-16}	0.0256	0.0231	0.0243

Na tym przykładzie widać widoczną zmianę w rozmiarach błędów. Nadal są one bardzo małe, w porównaniu do wcześniejszych przykładów wzrosły jednak o conajmniej 14 rzędów. Świadczy to niemal jednoznacznie o tym, że ilość wierszów macierzy A i B rozważanych w równaniu macierzowym $AX = B$ negatywnie wpływa na stabilność i poprawność implementowanego algorytmem zadania numerycznego.

3.4 Przykład 4

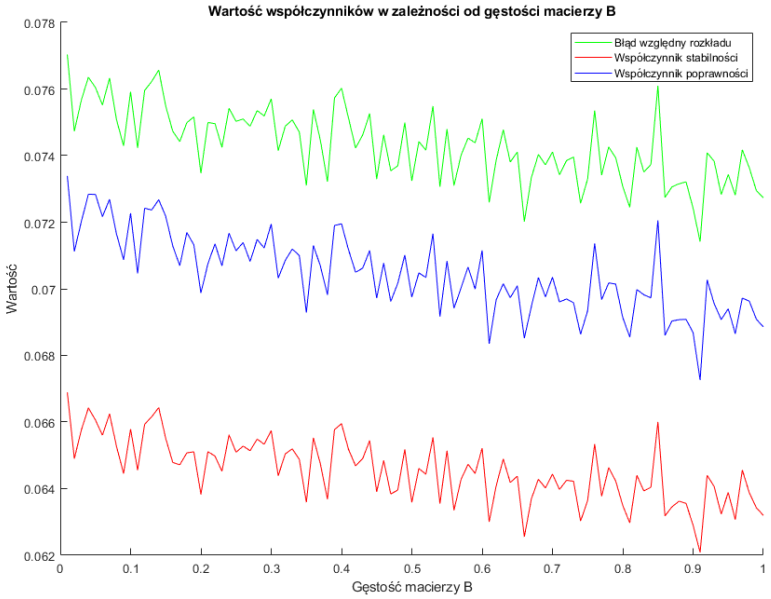
Przykład 4 dotyczy zmiany wartości trzech z pięciu wcześniej przedstawianych wartości współczynników i błędu w zależności od wielkości n dla macierzy $B = I_{n \times n}$. Dla ustalenia uwagi kolejne macierze A będą miały bardzo zbliżony wskaźnik uwarunkowania.
Macierze A generowane będą funkcją generatesparseSPDmatrix, dzięki czemu poprzez ustalenie stałej gęstości otrzymamy porównywalne do siebie macierze A pod względem wskaźnika uwarunkowania.



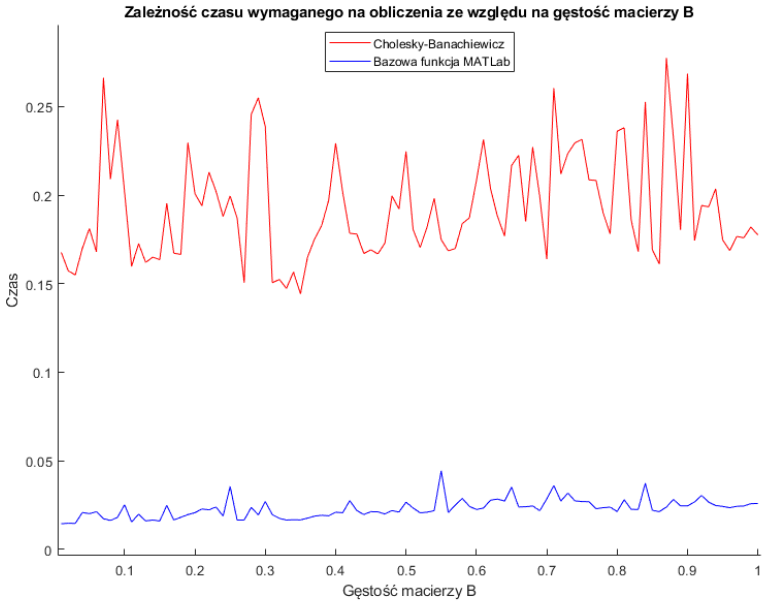
Pierwszą istotną obserwacją może być fakt, że wszystkie z mierzonych wielkości są nieporównywalnie większe od tych otrzymywanych w poprzednich przykładach dla tych samych wymiarów macierzy. Różnica ta wynosi ponad kilkanaście rzędów wielkości. Oznacza to, że B jako macierz jednostkowa bez zaskoczenia stanowi duży problem dla implementowanej metody. Rozwiązaniem takiego równania macierzowego jest bowiem macierz odwrotna macierzy A. Ciekawym zjawiskiem jest jednak fakt, że dla dowolnego rozmiaru macierzy B, błąd względny rozkładu jest ostro większy od współczynnika poprawności, który zawsze jest z kolei ostro większy od współczynnika stabilności. Kolejnym istotnym spostrzeżeniem jest to, że wraz z wzrostem rozmiaru macierzy B, wartość mierzonych wielkości maleje. Brak spójności z poprzednim przykładem może być uargumentowany tym, że więcej elementów w każdej kolumnie macierzy B umożliwia dokładniejszy dobór współczynników macierzy wynikowej X tak, by wymnożona macierz miała w odpowiednich miejscach wartość jak najbliższą 0.

3.5 Przykład 5

Przykład 5 będzie dotyczyć macierzy A oraz B zmieniającej się gęstości dotyczącej ilości zer jako współczynników. Wymiary macierzy zarówno A, jak i B ustalone zostaną na 400×400, jednak gęstość zmieniana będzie jedynie w przypadku macierzy B. Macierz A posiadać będzie ustaloną gęstość równą 0.5.



Interesującym spostrzeżeniem jest fakt, iż zachowana została relacja pomiędzy badanymi wartościami względem przykładu 4. Istotną obserwacją jest również to, że wychylenia od linii trendu są znaczące nawet na przestrzeni niewielkich różnic w gęstości, co spowodowane jest losowością we współczynnikach macierzy zarówno A, jak i B. Tendencja jest jednak widocznie spadkowa, z czego natychmiast wnioskować można, iż dla macierzy z jak najmniejszą liczbą zerowych współczynników, metoda implementowana algorytmem będzie najbardziej poprawna i stabilna. Zmiany te są jednak nieznaczące i tym bardziej tłumione przez liczne wychylenia. Skoro jednak ustalić można, iż gęstość macierzy ma wpływ na wartość współczynników stabilności i poprawności implementacji w języku MATLab, spójrzmy na wpływ gęstości na czas potrzebny do zrealizowania algorytmu:

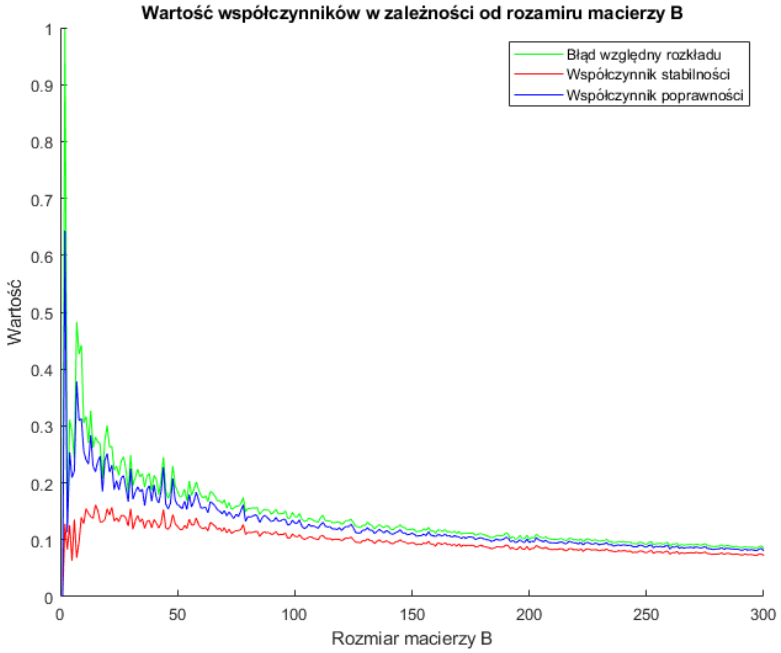


Najistotniejszym wnioskiem z powyższego wykresu powinno być to, jak dużo wolniejsza jest stosowana implementacja, względem wbudowanej funkcji inv(). Podobnie jak w przypadku błędów, wychylenia również występują, w tym wypadku są jednak dużo bardziej znaczące. Śmiało można z kolei przyjąć, że czas potrzebny do wykonania obu algorytmów niezależny jest od gęstości macierzy, czego przyczyną może być niezależność wymaganej liczby obliczeń od liczby zer w macierzy B. Istnieją więc podstawy do przypuszczenia, iż wymagany czas zależeć będzie od rozmiaru macierzy B.

3.6 Przykład 6

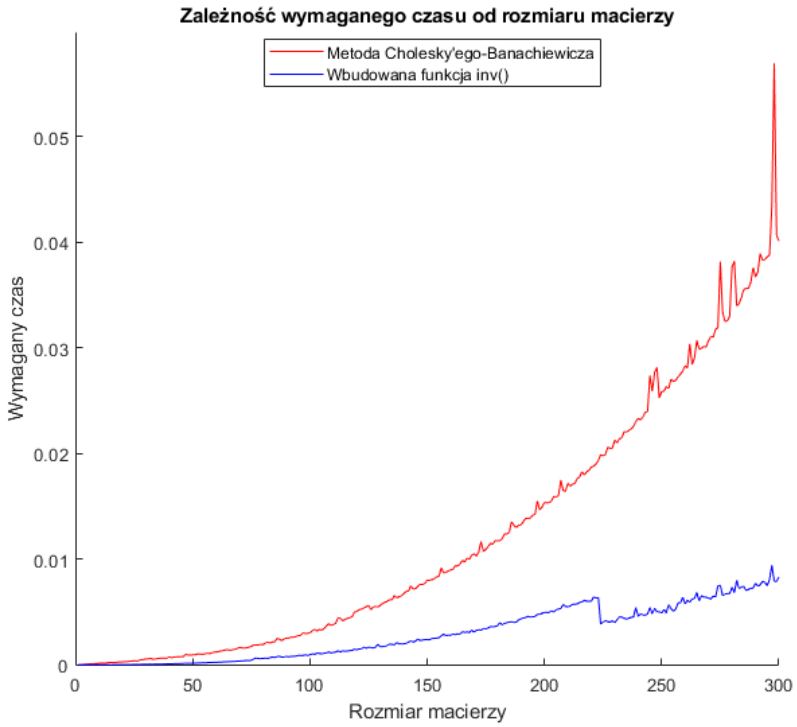
Sprawdźmy wcześniej przyjętą hipotezę na podstawie równań macierzowych macierzy róznych rozmiarów. Dla ustalenia uwagi, niech macierz A będzie miała zbliżony wskaźnik uwarunkowania w każdej z iteracji, natomiast macierz B będzie miała postać w pewnym sensie odwrotną do macierzy jednostkowej, tj. (dla $n = 5$):

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Tendencja spadkowa badanych wartości jest analogiczna do tej zauważonej w przykładzie 4, jednak istotnym spostrzeżeniem jest to, że sama wartość błędów jest nawet większa od tych otrzymanych w przypadku macierzy jednostkowych. Same wartości zbiegają wręcz asymptotycznie do wartości około 0.1, przez co można wnioskować, że dalsze zwiększanie rozmiaru macierzy B będzie powodowało coraz mniejsze spadki w badanych wartościach. (Projekt nie przewiduje sprawdzenia tej hipotezy ze względu na ograniczenia sprzętowe)

Spójrzmy jednak na sedno tego przykładu, czyli czas wymagany na zrealizowanie algorytmu implementującego metodę Cholesky'ego-Banachiewicza i porównanie go z czasem wymaganym do uzyskania dokładnego wyniku używając wbudowaną funckją inv():



Zgodnie z oczekiwaniami, czas wymagany na wykonanie zadania niezależnie od wybranego sposobu wzrasta wraz z zwiększeniem rozmiaru macierzy. Istotna jest też różnica w czasie wymaganym do osiągnięcia wyniku przez oba sposoby. Cholesky-Banachiewicz nie tylko jest wolniejszy dla każdego z rozmiarów, ale ma też szybszy wzrost wymaganego czasu wraz z wzrostem rozmiarów macierzy. Potwierdza to wcześniej postawioną tezę, iż to właśnie rozmiar macierzy wejściowych wpływa na wymagany czas na obliczenie wyniku implementowaną metodą.

4 Podsumowanie

Implementowana metoda Cholesky'ego-Banachiewicza do rozwiązywania macierzowego $AX = B$ powyżej opisanym algorytmem sprawdza się zaskakująco dobrze. W przypadku małych macierzy mierzone kryteria poprawności, stabilności i błędów są znikomo małe zarówno dla zbliżonych, jak i znacząco oddalonych od siebie pod względem wartości bezwzględnej współczynników. Większe odchylenia od dokładnego wyniku można zaobserować w ramach większej liczbie wierszów rozważanych macierzy. Liczba kolumn z kolei nie wpływa niemal w ogóle na badane wartości, ze względu na specyfikę implementacji - zwiększa się jedynie liczba wykonywanych podukładów.
Istotną cechą charakteryzującą analizowaną metodę jest również gorsza poprawność w przypadku, gdy macierz B jest macierzą jednostkową. Wówczas metoda pełni rolę znajdywania macierzy odwrotnej, co z perspektywy praktycznej może dyskwalifikować ten sposób jako uniwersalnie używany. Znajdywanie macierzy odwrotnej pełni bowiem kluczową rolę w operacjach macierzowych w przeróżnych dziedzinach nauki, takich jak fizyka, czy mechanika. Za wadę implementowanej metody można zdecydowanie uznać czas wymagany na zrealizowanie implementacji szczególnie, jeżeli porównany zostanie on do czasu realizacji funkcji inv().