Metody Numeryczne - projekt II

Aleks Kapich

Grudzień 2022

Spis treści

1 Opis zadania

Tematem projektu jest rozwiązywanie równania macierzowego $AX = B$, gdzie $A \in R^{n\times n}, B \in R^{n\times m}$, $m \geq 1$, metodą eliminacji Gaussa z pełnym wyborem elementu głównego (GECP) oraz wyznaczanie $\det(A)$.

2 Opis metody

Metoda eliminacji Gaussa z pełnym wyborem elementów głównych (GECP) jest rozszerzeniem bazowego wariantu metody eliminacji Gaussa. Podobnie, składa się z dwóch etapów:

1. Etap eliminacji – sprowadzenie macierzy układu do postaci trójkątnej.
2. Rozwiązanie układów z macierzą trójkątną.

Eliminację wykonujemy zaczynając od pierwszej kolumny – zerujemy w niej wszystkie elementy z wyjątkiem pierwszego, odejmując od wierszy 2-go, 3-go, . . . , n-go odpowiednie wielokrotności pierwszego wiersza. Następnie, w drugiej kolumnie zerujemy wszystkie elementy z wyjątkiem dwóch pierwszych, odejmując od wierszy 3-go, 4-go, . . . , n-go odpowiednie wielokrotności drugiego wiersza. W każdej z kolumn zerujemy elementy znajdujące się poniżej głównej przekątnej zgodnie ze schematem, wszystkie operacje wykonując na macierzy rozszerzonej układu.

Niech $A^{(k)} \in R^{n\times n}$ będzie macierzą A po k krokach eliminacji. Wcześniej wspomniane rozszerzenie polega na wybraniu elementu głównego przed przystąpieniem do eliminacji elementów z kolumny. Na początku k-tego kroku przeszukujemy podmacierz macierzy $A^{(k-1)}$, uwzględniając jedynie wiersze k, k+1, ..., n oraz kolumny k, k+1, ..., n. Szukamy takich indeksów p i q, dla których:

$$a_{pq}^{(k-1)} = \underset{i=k,\ldots,n, j=k,\ldots,n}{max} |a_{ij}^{(k-1)}|$$

a następnie zamieniamy miejscami wiersze p-ty i k-ty oraz kolumny k-tą i q-tą w macierzy $(A^{(k-1)}|b^{(k-1)})$. W szczególności, przed pierwszym krokiem eliminacji, elementu o największym module szukamy w całej macierzy układu, a następnie zamieniamy wiersze i kolumny tak, aby ten element znalazł się w pierwszym wierszu na głównej przekątnej.

Istotnym jest, że zmiana kolejności kolumn w macierzy układu równań liniowych zmienia kolejność niewiadomych. Należy uwzględnić to przy wyznaczaniu rozwiązania. Ponadto zamiana w macierzy dwóch wierszy lub dwóch kolumn miejscami zmienia znak jej wyznacznika. Zatem, wykorzystując metodę GECP do obliczenia wyznacznika, musimy uwzględnić łączną liczbę zamian wierszy i kolumn.

Drugi etap polegający na rozwiązaniu układu z macierzą trójkątną sprowadza się do użycia poniższego algorytmu dla każdego z powstałych układów:

$x_n := \frac{b_n}{a_{nn}}$
for $k = n-1, n-2, ..., 1$

$x_k := \frac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}}$
end

3 Kod

Implementując metodę GECP służącą do rozwiązywania równań macierzowych i obliczania wyznacznika macierzy, stworzyłem w Matlabie 3 funkcje:

1. Funkcja główna **GECP** - rozwiązuje ona równanie macierzowe korzystając z metody eliminacji Gaussa z pełnym wyborem elementów głównych.
2. Funkcja **determinant** - oblicza wyznacznik macierzy, przekształcając macierz wejściową w macierz trójkątną górną z użyciem metody GECP. Zlicza w trakcie tego procesu liczbę zamian wierszy oraz kolumn, która wpływa na znak wyznacznika.

3. Funkcja pomocnicza **findMaxSubmatrix**. Służy ona do wykonania pełnego wyboru elementu głównego w odpowiedniej podmacierzy macierzy podawanej w argumencie. Ma ona zastosowanie wewnątrz dwóch wymienionych wcześniej funkcji.

Ponadto stworzyłem również dwie niewielkie funkcje pomocnicze, użyteczne przy analizie - służą do wyznaczania kolejno współczynnika poprawności oraz błędu względnego obliczeń.

Funkcja główna:

```
function [X] = GECP(A, B)
% funkcja rozwiazujaca rownanie macierzowe AX = B metoda eliminacji
% Gaussa z pelnym wyborem elementu glownego (GECP)

% upewnic sie ze macierz A jest wymiarow nxn
if size(A, 1) ~= size(A, 2)
    warning("Macierz A nie jest kwadratowa!")
end

if size(A, 2) ~= size(B, 1)
    warning("Macierze A i B maja niepasujace wymiary!")
end


X = zeros(size(A, 1), size(B, 2)); % macierz wynikowa ma tyle wierszy
% ile macierz A oraz tyle kolumn ile macierz B

p = 1:size(A, 2); % wektor zapamietujacy kolejnosc niewiadomych

joint_matrix = [A B]; % macierz A|b
% przeprowadzamy eliminacje Gaussa
for k = 1:(size(A, 2)-1)
    % kazdy krok nalezy poprzedzic wyborem elementu glownego
    [max_row, max_col] = findMaxSubmatrix(joint_matrix(1:size(A,1),1:size(A,2)), k);
    if (max_row ~= k)
        joint_matrix([k max_row],:) = joint_matrix([max_row k],:);
    end
    if (max_col ~= k)
        joint_matrix(:,[k max_col]) = joint_matrix(:,[max_col k]);
        p(:,[k max_col]) = p(:,[max_col k]);
    end

    % zerujemy elementy pod elementem a_kk
    for i = k+1:size(joint_matrix,1)
        l = joint_matrix(i, k)/joint_matrix(k,k);
            joint_matrix(i,:) = joint_matrix(i,:)-l*joint_matrix(k,:);
    end
end


% obliczenie X
for m = 1:size(B,2)
    n = size(joint_matrix, 1);
    % mamy obecnie macierz trojkatna z kolumna wyrazow wolnych A|b
    % od razu mozemy obliczyc ostatni x, x_n
    X(n, m) = joint_matrix(n, n+m)/joint_matrix(n,n);
    % obliczamy kolejno x_k
    for k = n-1:-1:1
        sum = 0;
        for j = k+1:n
            sum = sum + joint_matrix(k, j)*X(j, m);
        end
        X(k, m) = (joint_matrix(k, n+m) - sum)/joint_matrix(k,k);
    end

    % wlasciwa kolejnosc
    X(p',m)=X(1:size(X,1), m);
end
```

Funkcja licząca wyznacznik:

```
function [det] = determinant(A)
% funkcja obliczajaca wyznacznik macierzy A z uzyciem metody eliminacji
% Gaussa z pelnym wyborem elementu gwnego (GECP)

% upewnic sie ze macierz A jest nxn
if size(A, 1) ~= size(A, 2)
    warning("Macierz A nie jest kwadratowa!")
end

alterations = 0; % ile razy zamiana wierszy lub kolumn
for k = 1:(size(A, 2)-1)
    % kazdy krok nalezy poprzedzic wyborem elementu glownego
    [max_row, max_col] = findMaxSubmatrix(A(1:size(A,1),1:size(A,2)), k);
    if (max_row ~= k)
        alterations = alterations + 1;
        A([k max_row],:) = A([max_row k],:);
    end
    if (max_col ~= k)
        alterations = alterations + 1;
        A(:,[k max_col]) = A(:,[max_col k]);
    end

    % zerujemy elementy pod elementem a_kk
    for i = k+1:size(A,1)
        l = A(i, k)/A(k,k);
        A(i,:) = A(i,:)-l*A(k,:);
    end
end

% Policzenie wyznacznika
det = (-1)^alterations;
for k = 1:size(A,1)
    det = det*A(k,k);
end
```

Funkcja pomocnicza użyta w powyższych funkcjach:

```
function [max_row, max_col] = findMaxSubmatrix(matrix, k)
% funkcja zwraca element maksymalny w podmacierzy A(k:n, k:n)
% macierzy A (o wymiarach n x n) oraz indeksy wiersza i kolumny w
% ktorych znajduje sie element maksymalny

if size(matrix, 1) ~= size(matrix, 2)
    warning("Macierz A nie jest kwadratowa!")
end

max = abs(matrix(k,k));
max_row = k;
max_col = k;

for i = k:size(matrix, 1) % liczba wierszy
    for j = 1:size(matrix, 2) % liczba kolumn
        if abs(matrix(i,j)) > max
            max = matrix(i,j);
            max_row = i;
            max_col = j;
        end
    end
end
```

4 Analiza przykładów

Treści przedstawione poniżej znajdują się w pliku **examples.m** załączonym do sprawozdania. Na potrzeby obliczeń zakładać będziemy, że wyniki prawidłowe są to te uzyskane za pomocą funkcji wbudowanych.

Przykład 1.

Na początek pokażemy, że funkcja działa prawidłowo - zaczniemy od prostego równania macierzowego, które pojawiło się na wykładzie.

$$A_1 = \begin{bmatrix} 2 & 1 & -1 \\ -4 & -1 & 3 \\ 6 & 1 & -3 \end{bmatrix}, b_1 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Rozwiążmy równanie $A_1 x = b_1$;
Rozwiązanie otrzymane za pomocą funkcji **GECP**:

$$x = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

Rozwiązanie otrzymane za pomocą funkcji wbudowanej **linsolve**:

$$x = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

Jak widać, rozwiązania są identyczne. Poniżej tabela zawierająca dane dotyczące wykonanych obliczeń.

Współczynnik poprawności	Błąd względny
6.1679e-18	0

Sprawdźmy teraz, jak funkcja **determinant** poradzi sobie z obliczeniem wyznacznika macierzy A_1 w porównaniu z funkcją wbudowaną **det**.

Wynik **determinant**	Wynik **det**	Błąd względny
4	4	0

Przykład 2.

Posłużymy się teraz drugim przykładem z wykładu, uwzględniającym nieznacznie większe macierze.

$$A_2 = \begin{bmatrix} -2 & 2 & 1 & 0 \\ 4 & -2 & 0 & 0 \\ 0 & 1 & 2 & 5 \\ 0 & 0 & 5 & 20 \end{bmatrix}, b_2 = \begin{bmatrix} -1 \\ 4 \\ 0 \\ -5 \end{bmatrix}$$

Rozwiążmy równanie $A_2 x = b_2$;
Rozwiązanie otrzymane za pomocą funkcji **GECP**:

$$x = \begin{bmatrix} 2 \\ 2 \\ -1 \\ 0 \end{bmatrix}$$

Rozwiązanie otrzymane za pomocą funkcji wbudowanej **linsolve**:

$$x = \begin{bmatrix} 2 \\ 2 \\ -1 \\ 0 \end{bmatrix}$$

Ponownie, rozwiązania są identyczne.

Współczynnik poprawności	Błąd względny
0	0

Porównajmy funkcję **determinant** z funkcją wbudowaną **det**.

Wynik **determinant**	Wynik **det**	Błąd względny
20	20	0

Operując na liczbach całkowitych otrzymaliśmy idealny wynik pozbawiony błędu.

Przykład 3.

Przyjrzyjmy się równaniu macierzowemu gdzie macierz B składa się z więcej niż 1 kolumny.

$$A_3 = \begin{bmatrix} 2 & 1 \\ 3 & 7 \end{bmatrix}, b_2 = \begin{bmatrix} 4 & 16 \\ 6 & 46 \end{bmatrix}$$

Rozwiążmy równanie $A_3 x = b_3$;
Rozwiązanie otrzymane za pomocą funkcji **GECP**:

$$x = \begin{bmatrix} 2 & 6 \\ 0 & 4 \end{bmatrix}$$

Rozwiązanie otrzymane za pomocą funkcji wbudowanej **linsolve**:

$$x = \begin{bmatrix} 2 & 6 \\ 0 & 4 \end{bmatrix}$$

Po raz kolejny, rozwiązania są identyczne.

Współczynnik poprawności	Błąd względny
0	0

Porównanie funkcji obliczających wyznacznik:

Wynik **determinant**	Wynik **det**	Błąd względny
11	11	1.6149e-16

Jak widzimy powyżej, tym razem błąd względny przy obliczaniu wartości wyznacznika nie jest równy 0 - mimo tego, że jest do 0 bardzo zbliżony. Tutaj widać minimalną "wyższość"funkcji **determinant** używającej metody eliminacji Gaussa z pełnym wyborem elementu głównego, bowiem użycie polecenia **isequal(11, determinant(A3))** w Matlabie zwraca wynik TRUE w przeciwieństwie do polecenia **isequal(11, det(A3))**, które zwraca FALSE. Funkcja wbudowana ze względu na błędy działań arytmetycznych nie zwraca dokładnie prawidłowej wartości. Błąd jest jednak na tyle mały, że można pominąć tę niedokładność.

Przykład 4.

Sprawdźmy teraz działanie funkcji, gdy rozpatrywać będziemy macierze zawierające ułamki.

$$A_4 = \begin{bmatrix} 0.1111 & 0.2604 \\ 0.2137 & 0.2669 \end{bmatrix}, b_4 = \begin{bmatrix} 0.2584 & 0.2103 \\ 0.6696 & 0.2505 \end{bmatrix}$$

Rozwiążmy równanie $A_4 x = b_4$;
Rozwiązanie otrzymane za pomocą funkcji **GECP**:

$$x = \begin{bmatrix} 4.0545 & 0.3501 \\ -0.7375 & 0.6582 \end{bmatrix}$$

Rozwiązanie otrzymane za pomocą funkcji wbudowanej **linsolve**:

$$x = \begin{bmatrix} 4.0545 & 0.3501 \\ -0.7375 & 0.6582 \end{bmatrix}$$

Dla macierzy zawierającej liczby inne niż całkowite również otrzymaliśmy poprawne rozwiązanie. Poniżej tabela zawierająca dane dotyczące wykonanych obliczeń.

Współczynnik poprawności	Błąd względny
2.1968e-17	5.3797e-17

Jak widać poniżej, funkcja wbudowana i funkcja **determinant** zwróciły dokładnie ten sam wynik, błąd względny wynosi 0.

Wynik **determinant**	Wynik **det**	Błąd względny
-0.026	-0.026	0

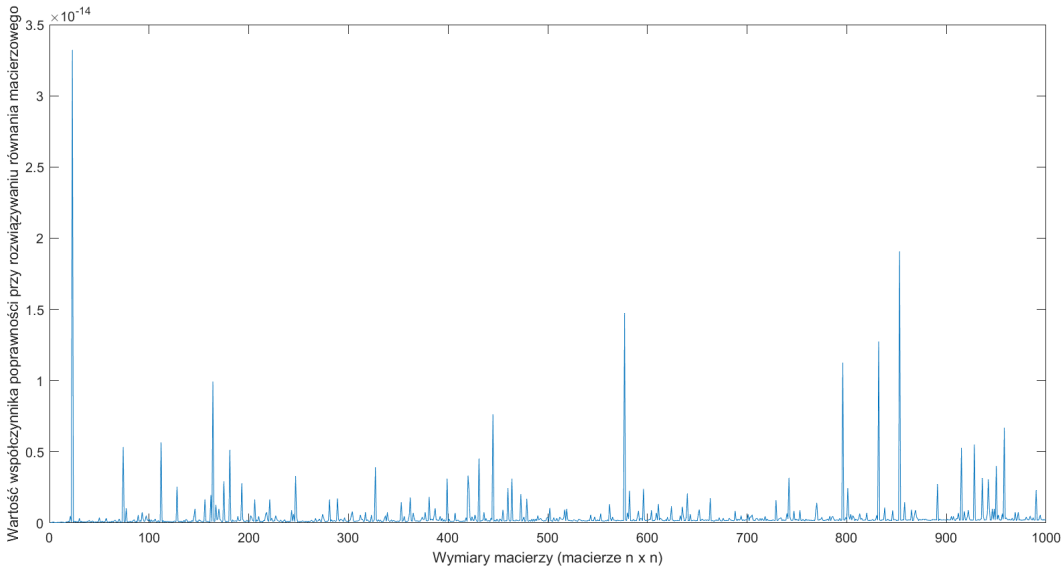
Przykład 5.

Po zaprezentowaniu działania funkcji na pojedynczych przykładach, czas na przeprowadzenie bardziej obszernej analizy. Wygenerujemy 1000 losowych par macierzy, A_i, B_i dla $i = 1, 2, ..., 1000$, gdzie dla każdego i macierze A_i i B_i będą wymiaru $i \times i$.

Zbadamy, jak zmieniają się współczynnik poprawności oraz błędy względne w zależności od wymiarów macierzy, na jakiej operujemy.

5.1

Na poniższym wykresie widać, że wymiar macierzy nie wpływa na współczynnik poprawności przy rozwiązywaniu równania macierzowego. Zazwyczaj wartości współczynnika są rzędu 10^{-17} lub 10^{-16}. Spośród 1000 obserwacji wyróżnia się kilka przypadków, gdzie współczynnik osiągał odstające wartości rzędu 10^{-14}, jednakże wciąż są to bardzo małe liczby.

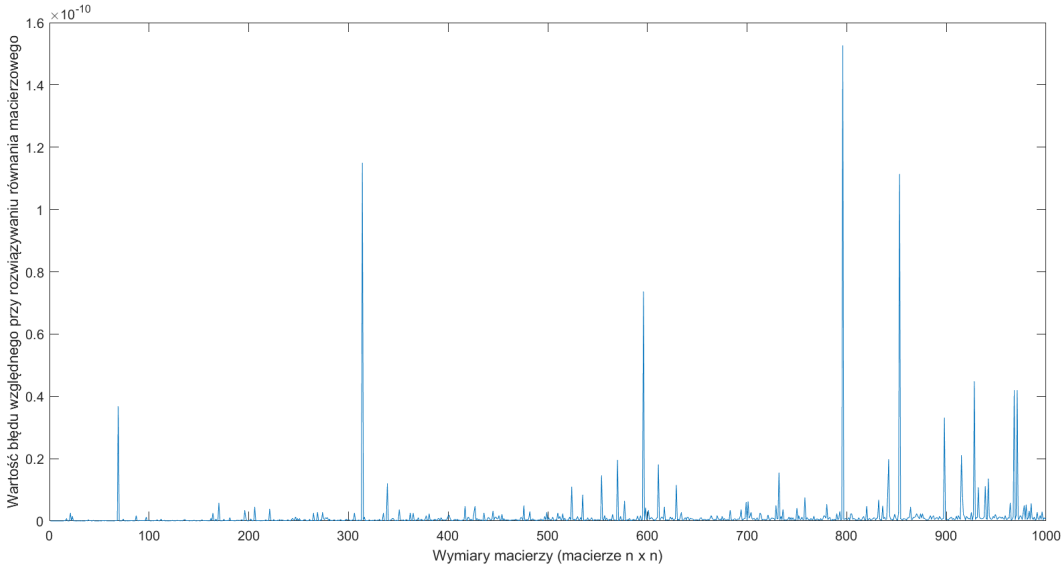


Rysunek 1: Współczynnik poprawności w zależności od wymiaru macierzy

Średnia arytmetyczna	Mediana
4.4529e-16	1.8131e-16

5.2

W wypadku błędu względnego rozwiązywania równania macierzowego, wymiary macierzy mają znaczenie. Wraz ze wzrostem wymiaru macierzy wartości błędu również wzrastają, częściej występują również ponadprzeciętnie wysokie wartości. Niemniej, nawet dla większych macierzy błąd jest zazwyczaj rzędu 10^{-11} lub 10^{-12}, czyli relatywnie niewielki.

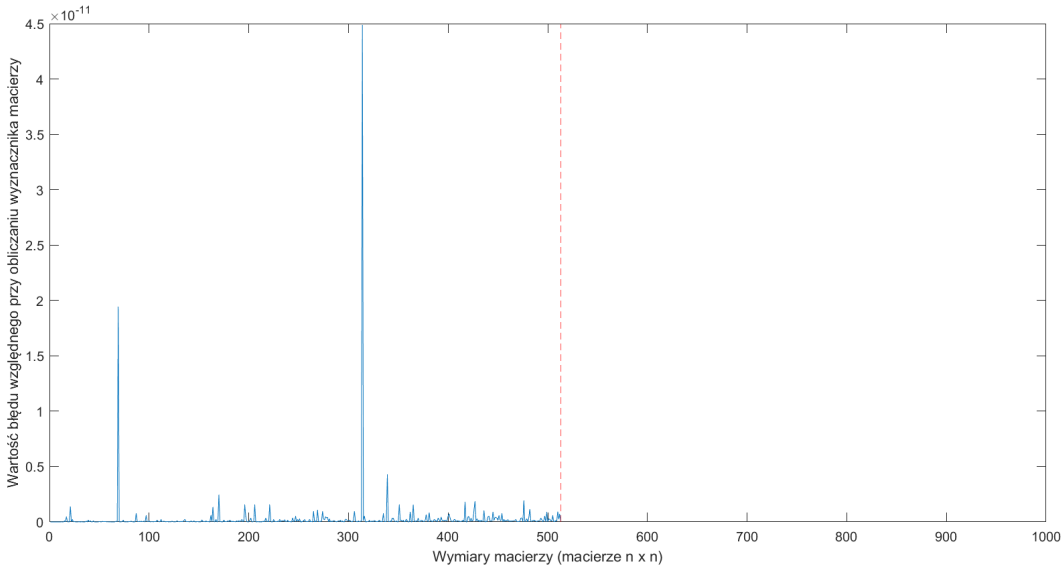


Rysunek 2: Błąd względny rozwiązania równania macierzowego w zależności od wymiaru macierzy

Średnia arytmetyczna	Mediana
1.4871e-12	3.6073e-13

5.3

Przy wyznaczaniu błędu względnego związanego z obliczaniem wyznacznika, rozmiar macierzy również ma znaczenie. Na wykresie nie widać tego klarownie, ponieważ jest on spłaszczony przez jedną odstającą zdecydowanie obserwację. Co rzuca się w oczy, mniej więcej w połowie osi wykres się kończy, mimo iż wykonaliśmy 1000 obserwacji. Miejsce to oznaczone jest czerwoną, przerywaną linią. Dla macierzy rzędu 514×514 oraz większych Matlab nie był w stanie policzyć wyznacznika, zwracał wartość Inf lub -Inf, w skutek tego nie był również w stanie wyznaczyć błędu obliczeń. Porównując wykres błędu względnego obliczania wyznacznika z wykresem błędu względnego rozwiązywania równania macierzowego, widzimy że miejsca w których występują charakterystyczne skoki pokrywają się ze sobą.



Rysunek 3: Błąd względny obliczania wyznacznika w zależności od wymiaru macierzy

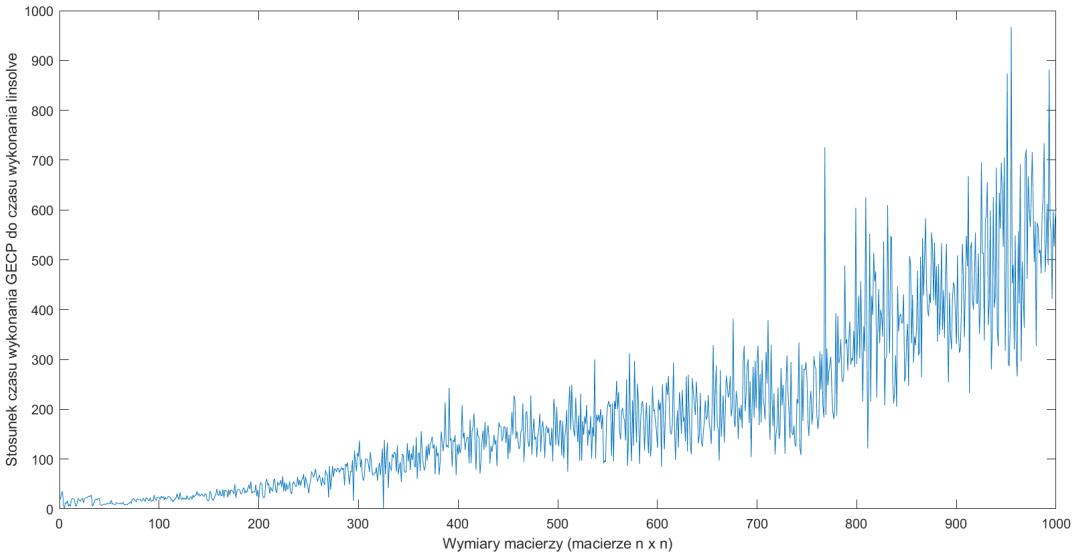
W poniższej tabeli uwzględnione jest jedynie pomyślne 513 pierwszych obserwacji.

Średnia arytmetyczna	Mediana
2.8091e-13	4.6466e-14

Przykład 6.

W tym przykładzie skupimy się na porównaniu wydajności czasowej funkcji **GECP** oraz funkcji wbudowanej **linsolve**. Podobnie jak w przykładzie piątym, będziemy generować 1000 losowych par macierzy, gdzie każda para będzie większa od pary poprzedników. Zbadamy stosunek czasu jaki potrzebny jest na wykonanie funkcji **GECP** do czasu potrzebnego na wykonanie funkcji **linsolve**.

Na poniższym wykresie widać, że im większe macierze składają się na rozpatrywane równanie macierzowe, tym gorzej wypada napisana przeze mnie funkcja w porównaniu z funkcją wbudowaną. Dla małych macierzy różnica jest praktycznie niezauważalna, bowiem metoda **GECP** wypada tylko kilkanaście razy gorzej, a obliczenia wykonywane są w przeciągu ułamków sekund. Dla dużych macierzy jednak różnica staje się zauważalna, bowiem funkcja wbudowana radzi sobie kilkaset razy lepiej.



Rysunek 4: Proporcja czasu wywołania **GECP** do czasu wywołania **linsolve**

5 Podsumowanie

Jak widać, zaimplementowana przeze mnie metoda sprawuje się poprawnie i jest w stanie dokładnie rozwiązywać równania macierzowe, dzięki eliminacji Gaussa z pełnym wyborem elementu głównego można również obliczać wyznacznik macierzy. Jednakże, szczególnie w wypadku macierzy o większych wymiarach, lepiej jest wykorzystywać analogiczne metody wbudowane, bowiem spisują się odczuwalnie szybciej. Nie jest to jednak zaskoczenie, ponieważ gdyby istniała optymalna alternatywa dla funkcji **linsolve** i **det**, najprawdopodobniej osoby odpowiedzialne za rozwój języka Matlab zadbałyby o zwiększenie wydajności i zastąpiły dotychczasowe implementacje metod ich lepszymi wersjami.

Literatura

- Notatki z wykładu z przedmiotu Metody Numeryczne autorstwa dr Iwony Wróbel
- *D. Kincaid, W. Cheney - Analiza numeryczna* Warszawa 2005