MNUM – PROJEKT, zadanie 2.11

Bartosz Cywiński

Spis treści

Aproksymacja metodą najmniejszych kwadratów

Opis algorytmu

Zadanie aproksymacji: Niech $f(x)$ przyjmuje na pewnym zbiorze punktów $x_0, x_1, \dots, x_N$ znane wartości $y_j = f(x_j), j = 0,1, \dots, N$.

Wyznaczyć wartości współczynników $a_0, a_1, \dots, a_n$ określających funkcję aproksymującą $F(x) = \sum_{i=0}^{n} a_i \phi_i(x)$ tak, aby zminimalizować błąd średniokwadratowy:

$$H(a_0, \dots, a_n) \triangleq \sum_{j=0}^{N} \left[f(x_j) - \sum_{i=0}^{n} a_i \phi_i(x_j) \right]^2$$

Wyznaczymy poszukiwane współczynniki $a_0, \dots, a_n$ z warunków koniecznych minimum:

$$\frac{\partial H}{\partial a_k} = -2 \sum_{j=0}^{N} \left[f(x_j) - \sum_{i=0}^{n} a_i \phi_i(x_j) \right] \cdot \phi_k(x_j) = 0$$

Warunek ten wyznacza układ równań normalnych, który można zapisać w zwięzłej postaci:

$$\langle \phi_i, \phi_k \rangle \stackrel{\text{def}}{=} \sum_{j=0}^{N} \phi_i(x_j) \phi_k(x_j)$$

Układ równań przyjmuje wówczas postać:

$$\begin{bmatrix} \langle \phi_0, \phi_0 \rangle & \cdots & \langle \phi_n, \phi_0 \rangle \\ \vdots & \ddots & \vdots \\ \langle \phi_0, \phi_n \rangle & \cdots & \langle \phi_n, \phi_n \rangle \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \langle \phi_0, f \rangle \\ \langle \phi_1, f \rangle \\ \vdots \\ \langle \phi_n, f \rangle \end{bmatrix}$$

Zdefiniujmy macierz $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} \phi_0(x_0) & \cdots & \phi_n(x_0) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_n(x_N) \end{bmatrix}$$

Oraz wektory zmiennych decyzyjnych i wartości funkcji oryginalnej jako:

$$\boldsymbol{a} = [a_0 \quad \dots \quad a_n]^T$$
$$\boldsymbol{y} = [y_0 \quad \dots \quad y_n]^T, y_j = f(x_j), j = 0,1, \dots, N$$

Wówczas funkcję celu można zapisać w postaci:

$$H(\boldsymbol{a}) = \left(||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{a}||_2 \right)^2$$

Wówczas układ równań normalnych można zapisać w postaci:

$$\boldsymbol{A^T A\, a} = \boldsymbol{A^T y}$$

Natomiast macierz $\boldsymbol{A^T A}$ może być źle uwarunkowana – kiedy macierz $\boldsymbol{A}$ jest słabo uwarunkowana, ponieważ wskaźnik uwarunkowania macierzy $\boldsymbol{A^T A}$ jest kwadratem wskaźnika uwarunkowania macierzy $\boldsymbol{A}$. Wtedy zalecanym sposobem rozwiązania zadania aproksymacji jest wykorzystanie rozkładu $\boldsymbol{QR}$ macierzy $\boldsymbol{A}$:

$$\boldsymbol{R\,a = Q^T y}$$

Przy aproksymacji wielomianem jego stopień jest najczęściej dużo mniejszy od liczby punktów, na podstawie których dokonujemy aproksymacji, jednak dla sprawdzenia efektywności algorytmów przetestuję dokładność rozwiązania również dla wielomianu, którego stopień będzie równy liczbie próbek danych.

Funkcję wielomianową $y = f(x)$ wyznaczę dla następujących danych pomiarowych:

x_i	y_i
-5	–79,1639
-4	-40,7900
-3	-18,7814
-2	–6,3530
-1	–0,4392
0	0,8270
1	0,0585
2	–1,7477
3	–3,4384
4	–6,3580
5	–9,3875

Rozwiązanie przy pomocy układu równań normalnych

Kod programu

```
function [euclidean_errors, chebyshev_errors] = approxNormalEquations(n, x, y)
%APPROXNORMALEQUATIONS Aproksymacja metodą najmniejszych kwadratów z
%wykorzystaniem układu równań normalnych
A = zeros(size(x,1), max(n)+1);
euclidean_errors = zeros(size(n,1),1);
chebyshev_errors = zeros(size(n,1),1);

% Konstruuję macierz A, której kolumny to elementy wektora x podnoszone do
% kolejnych potęg
for i = 1:size(x,1)
    for k = 1:max(n)+1
        A(i,k) = x(i)^(k-1);
    end
end

if size(n,1) > 1
    subplot(2,1,1);
end

lin = linspace(min(x), max(x), 100)';
scatter(x,y, "DisplayName","dane");
```

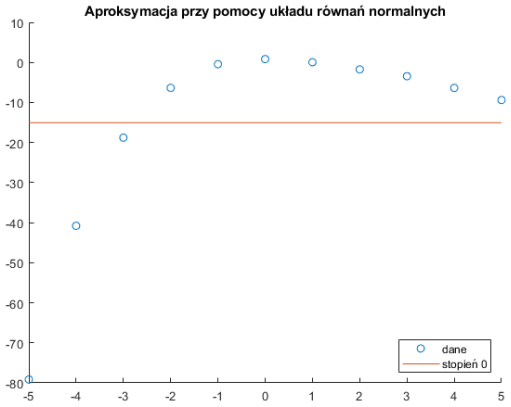
```
legend('Location', 'southeast');
title("Aproksymacja przy pomocy układu równań normalnych");
hold on;

for i = 1:size(n,1)
    A_i = A(:,1:n(i)+1);

    % Rozwiązuje równanie normalne przy pomocy rozkładu LU
    a_i = LUDecomposition(A_i'*A_i, A_i'*y, n(i)+1);

    % Liczę błąd aproksymacji w normie euklidesowej
    euclidean_errors(i) = norm(polynomialValue(a_i, x)-y);

    % Liczę błąd aproksymacji w normie Czebyszewa
    chebyshev_errors(i) = norm(polynomialValue(a_i, x)-y, "inf");

    plot(lin, polynomialValue(a_i, lin),'DisplayName', "stopień " + n(i));
end
hold off;

if size(n,1) > 1
    subplot(2,1,2);
    plot(x, euclidean_errors, "DisplayName","Błąd w normie euklidesowej");
    title("Błędy");
    legend('Location', 'southeast');
    hold on;
    plot(x, chebyshev_errors, "DisplayName","Błąd w normie Czebyszewa");
    hold off;
end
end
```

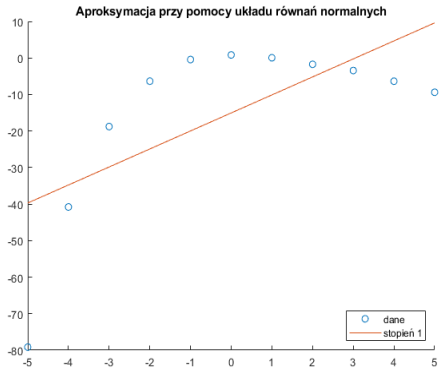
Do rozwiązania układu równań normalnych wykorzystałem zaimplementowany przeze mnie w ramach poprzedniego projektu rozkład LU.

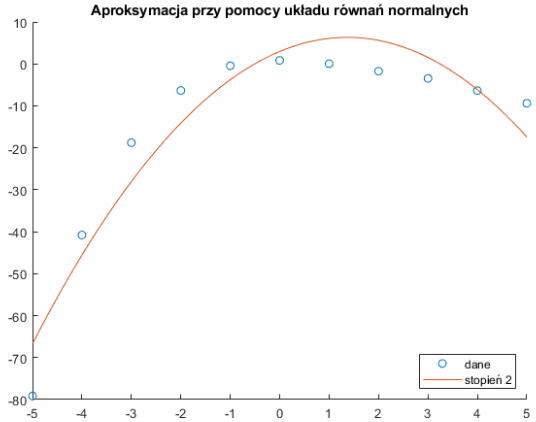
Wartości wielomianu obliczam pomocniczą funkcją:

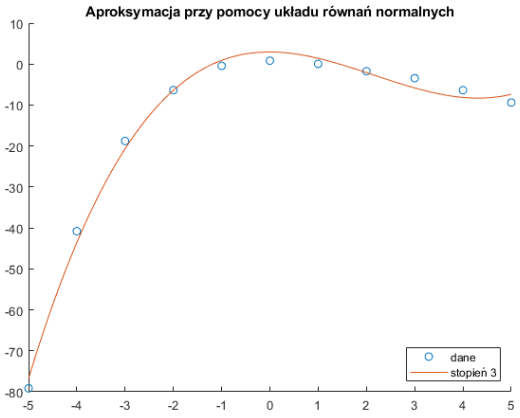
```
function [values] = polynomialValue(a, x)
%POLYNOMIALVALUE Liczy wartość wielomianu o danych współczynnikach
values = zeros(size(x,1),1);
for i = 1:size(x,1)
    for j = 1:size(a,1)
        values(i) = values(i) + a(j) * (x(i)^(j-1));
    end
end
end
```

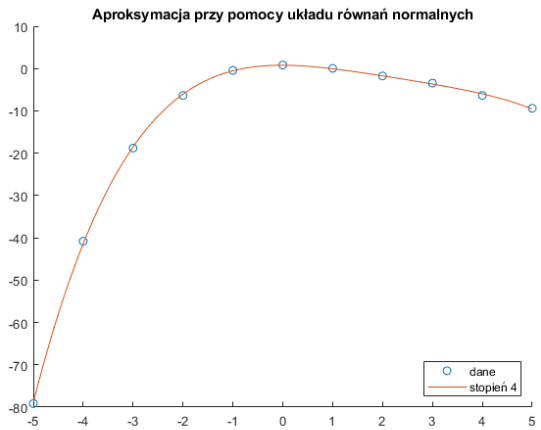
Prezentacja wyników

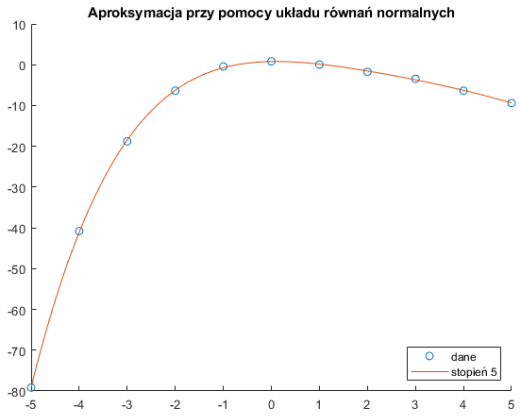


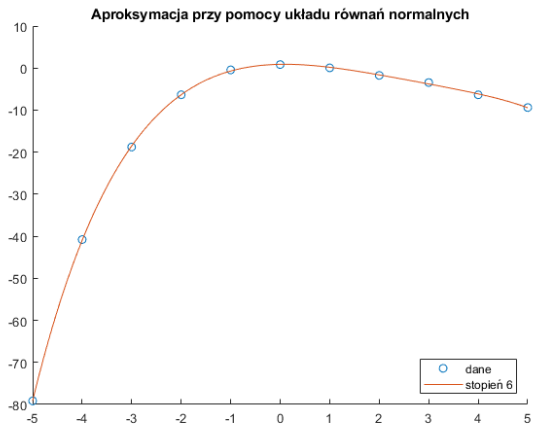


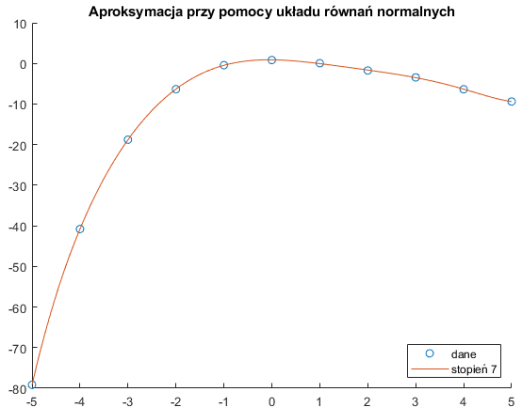


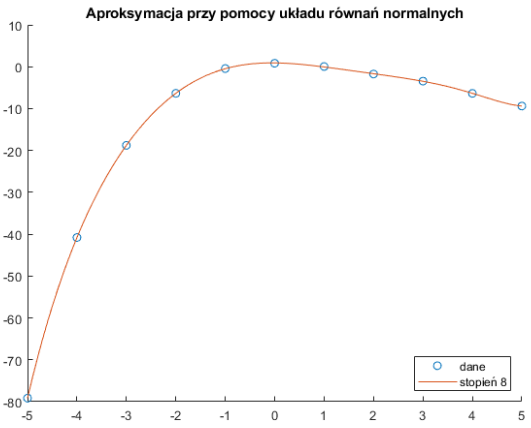


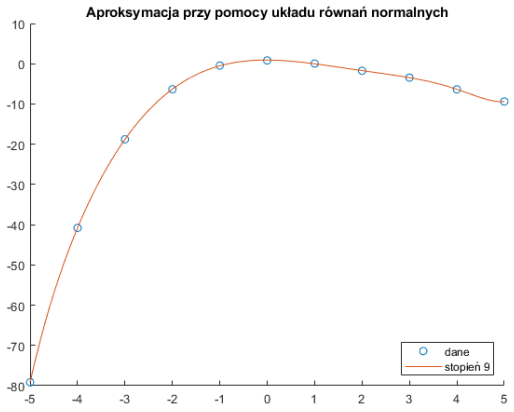


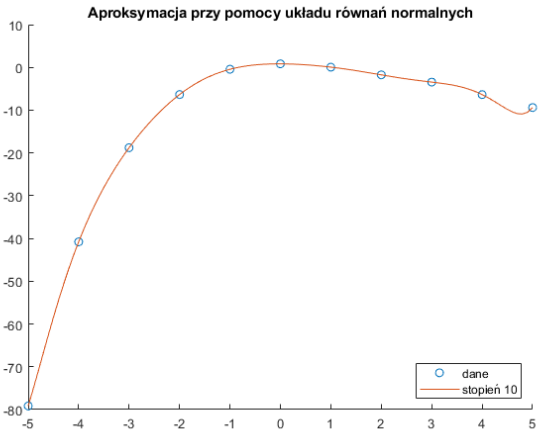


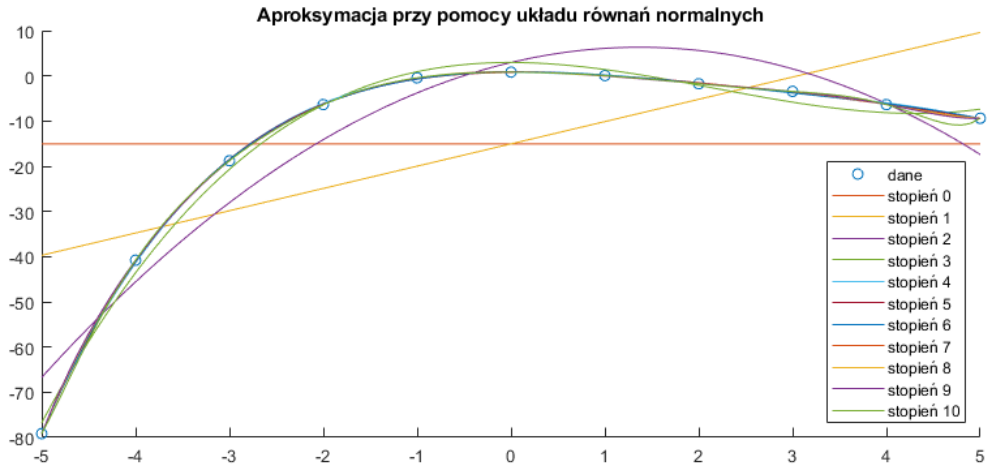


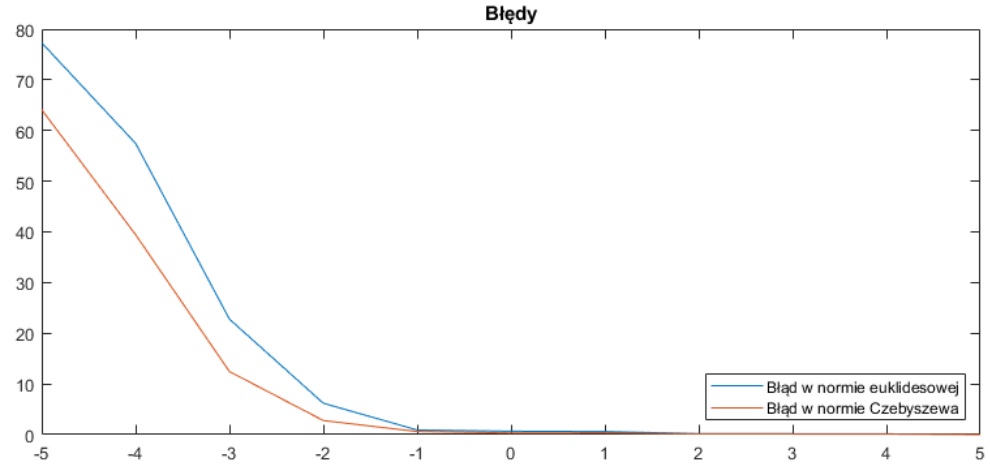



Aproksymacja przy pomocy układu równań normalnych
dane
stopień 8


Aproksymacja przy pomocy układu równań normalnych
dane
stopień 9


Aproksymacja przy pomocy układu równań normalnych
dane
stopień 10





Stopień wielomianu	Błąd w normie euklidesowej	Błąd w normie Czebyszewa
0	77.3041	64.1118
1	57.4623	39.4596
2	22.7857	12.4460
3	6.1916	2.7805
4	0.9143	0.6079
5	0.6673	0.3270
6	0.6057	0.3411
7	0.1891	0.1121
8	0.1773	0.1123
9	0.1553	0.0911
10	3.5907e-11	2.2230e-11

Rozwiązanie przy pomocy rozkładu QR

Kod programu

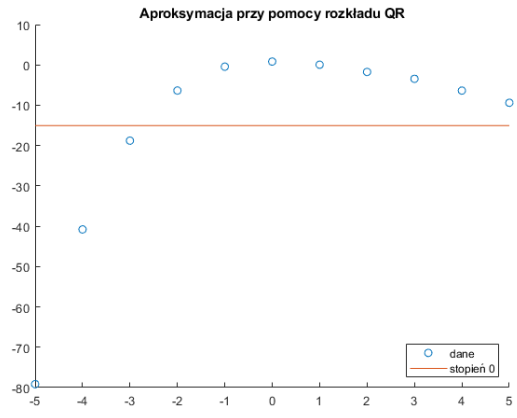
```
function [euclidean_errors, chebyshev_errors] = approxQR(n, x, y)
%APPROXQR Aproksymacja metodą najmniejszych kwadratów z wykorzystaniem
%rozkładu QR
A = zeros(size(x,1), max(n)+1);
euclidean_errors = zeros(size(n,1),1);
chebyshev_errors = zeros(size(n,1),1);

% Konstruuję macierz A, której kolumny to elementy wektora x podnoszone do
% kolejnych potęg
for i = 1:size(x,1)
    for k = 1:max(n)+1
        A(i,k) = x(i)^(k-1);
    end
end

if size(n,1) > 1
    subplot(2,1,1);
end

lin = linspace(min(x), max(x), 100)';
scatter(x,y, "DisplayName","dane");
legend('Location', 'southeast');
title("Aproksymacja przy pomocy rozkładu QR");
hold on;

for i = 1:size(n,1)
    A_i = A(:,1:n(i)+1);

    % Rozkład QR
    [Q, R] = qrmgs(A_i);

    % Rozwiązuje równanie normalne przy pomocy rozkładu LU
    a_i = LUDecomposition(R, Q'*y, n(i)+1);

    % Liczę błąd aproksymacji w normie euklidesowej
    euclidean_errors(i) = norm(polynomialValue(a_i, x)-y);

    % Liczę błąd aproksymacji w normie Czebyszewa
    chebyshev_errors(i) = norm(polynomialValue(a_i, x)-y, "inf");

    plot(lin, polynomialValue(a_i, lin),'DisplayName', "stopień " + n(i));
end
hold off;

if size(n,1) > 1
    subplot(2,1,2);
    plot(x, euclidean_errors, "DisplayName","Błąd w normie euklidesowej");
    title("Błędy");
    legend('Location', 'southeast');
    hold on;
    plot(x, chebyshev_errors, "DisplayName","Błąd w normie Czebyszewa");
    hold off;
```

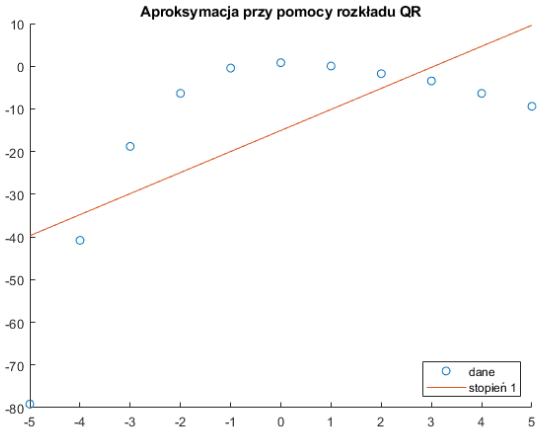
```
end
end
```

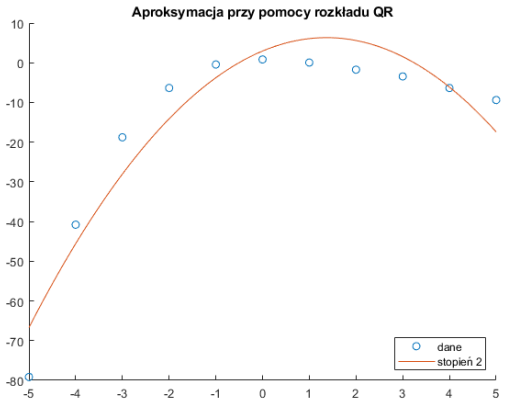
Do rozkładu QR użyłem pomocniczej funkcji, której kod umieszczony jest w podręczniku do przedmiotu:

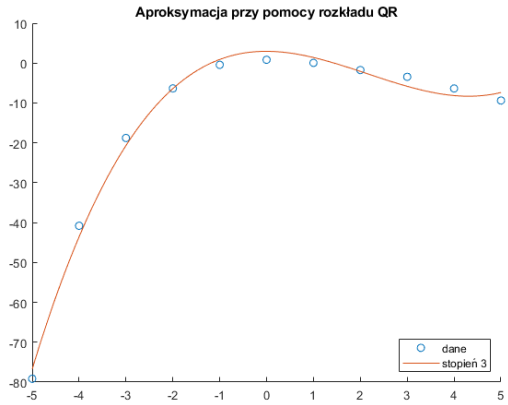
```
function [Q,R]=qrmgs(A)
%rozkład QR (wąski) zmodyfikowanym algorytmem Grama-Schmidta
%dla macierzy prostokątnych rzeczywistych i zespolonych
[m, n]=size(A);
Q=zeros(m,n);
R=zeros(n,n);
d=zeros(1,n);

%rozkład z kolumnami Q ortogonalnymi (nie ortonormalnymi):
for i=1:n
    Q(:,i)=A(:,i);
    R(i,i)=1;
    d(i)=Q(:,i)'*Q(:,i);
    for j=i+1:n
        R(i,j)=(Q(:,i)'*A(:,j))/d(i);
        A(:,j)=A(:,j)-R(i,j)*Q(:,i);
    end
end

%normowanie rozkładu (kolumny Q ortonormalne):
for i=1:n
    dd=norm(Q(:,i));
    Q(:,i)=Q(:,i)/dd;
    R(i,i:n)=R(i,i:n)*dd;
end
```

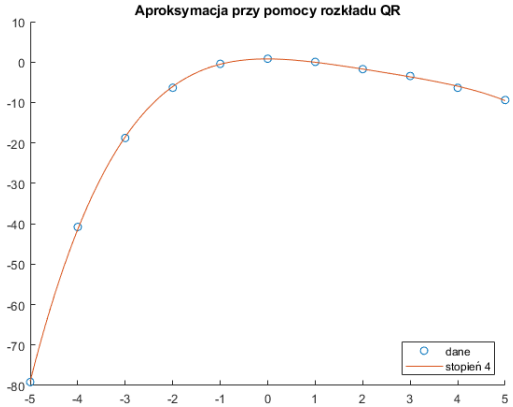
Prezentacja wyników

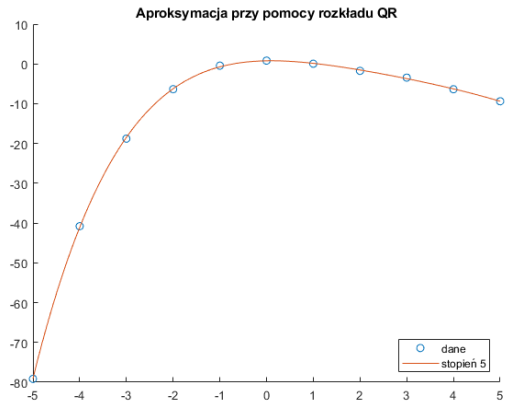


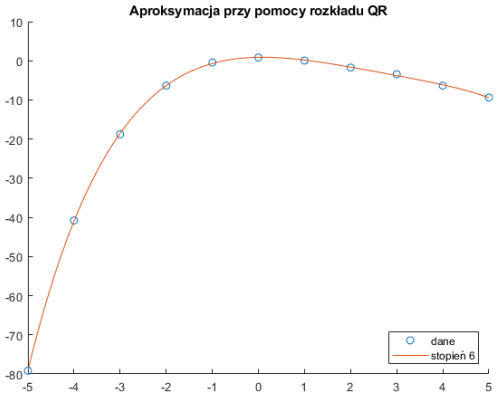


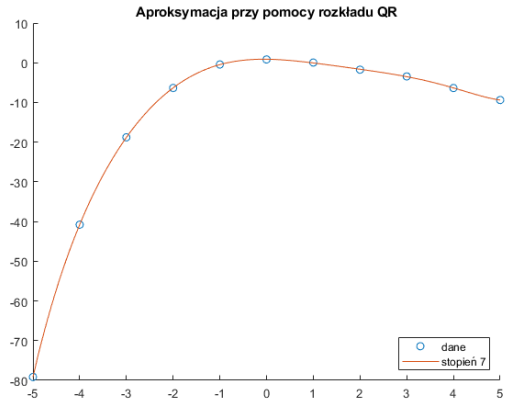

Aproksymacja przy pomocy rozkładu QR
dane
stopień 2

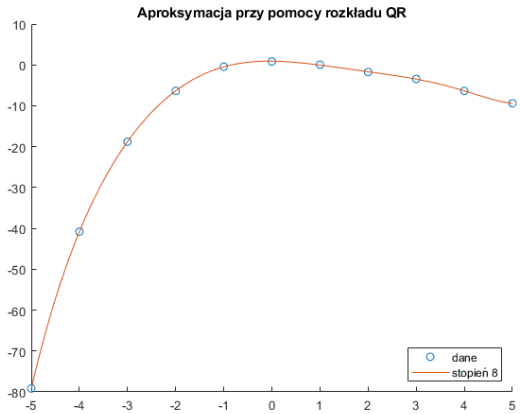

Aproksymacja przy pomocy rozkładu QR
dane
stopień 3

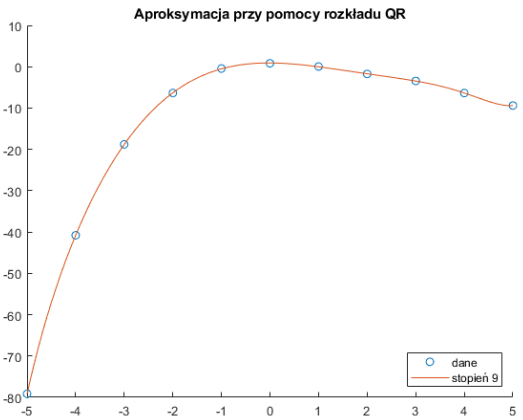

Aproksymacja przy pomocy rozkładu QR
dane
stopień 4

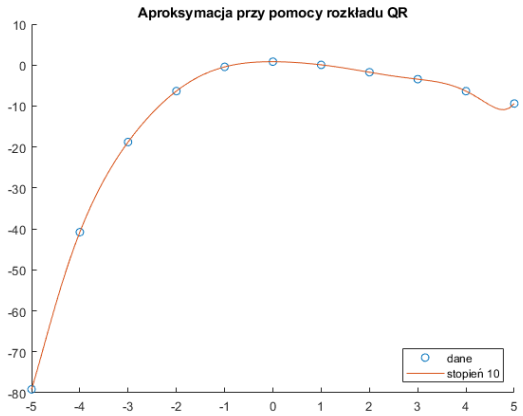

Aproksymacja przy pomocy rozkładu QR
dane
stopień 5

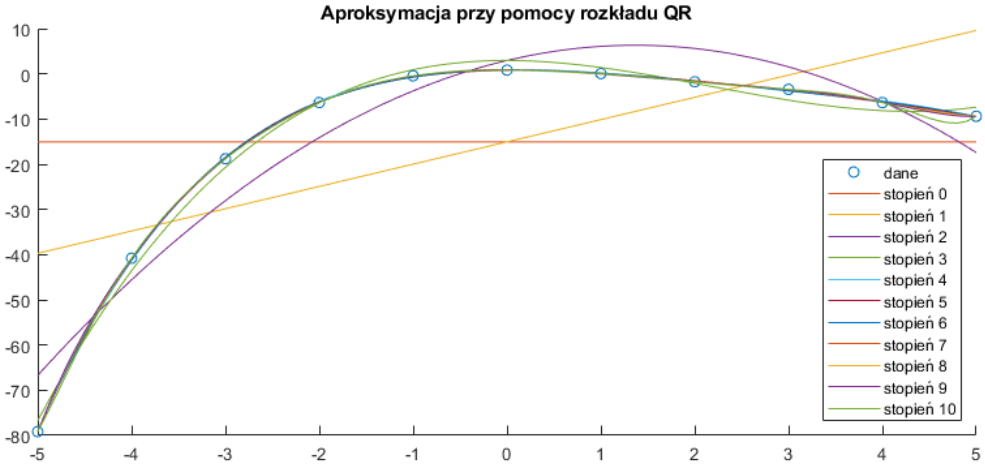

Aproksymacja przy pomocy rozkładu QR
dane
stopień 6

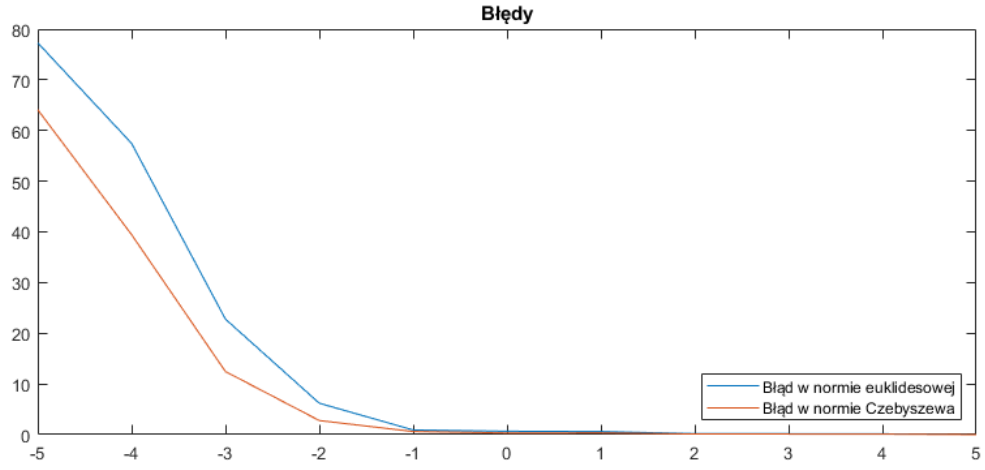

Aproksymacja przy pomocy rozkładu QR
dane
stopień 7


Aproksymacja przy pomocy rozkładu QR
dane
stopień 8


Aproksymacja przy pomocy rozkładu QR
dane
stopień 9


Aproksymacja przy pomocy rozkładu QR
dane
stopień 10


Aproksymacja przy pomocy rozkładu QR
dane
stopień 0
stopień 1
stopień 2
stopień 3
stopień 4
stopień 5
stopień 6
stopień 7
stopień 8
stopień 9
stopień 10


Błędy
Błąd w normie euklidesowej
Błąd w normie Czebyszewa

Stopień wielomianu	Błąd w normie euklidesowej	Błąd w normie Czebyszewa
0	77.3041	64.1118
1	57.4623	39.4596
2	22.7857	12.4460
3	6.1916	2.7805
4	0.9143	0.6079
5	0.6673	0.3270
6	0.6057	0.3411
7	0.1891	0.1121
8	0.1773	0.1123
9	0.1553	0.0911
10	1.0768e-11	6.3742e-12

Wnioski

W obu przypadkach błędy aproksymacji maleją wraz ze zwiększaniem się stopnia wielomianu. Wynika to z tego, że im większy stopień wielomianu tym funkcja aproksymująca jest w stanie dokładniej dopasować się do danych. Błędy mierzone zarówno w normie euklidesowej jak i w normie Czebyszewa zarówno przy aproksymacji przy pomocy układu równań normalnych, jak i przy pomocy rozkładu QR są bardzo podobne. Różnice dopiero widać przy aproksymacji wielomianem stopnia 10, ponieważ wtedy błędy są już minimalne.

Na przedstawione błędy składają się też błędy numeryczne popełniane przy rozwiązywaniu układów równań, a także wynikające z szumu w danych.

Po wykresach można stwierdzić, że już wielomian stopnia czwartego bardzo dobrze aproksymuje dane.