Projekt MNUM
Zadanie 2.19
Karol Mocniak

Zadanie 1.

Program do obliczania wartości własnych metodą QR w wersji bez przesunięć i z przesunięciami dla macierzy symetrycznej oraz z przesunięciami dla macierzy niesymetrycznej.

Główna funkcja zadania wywołująca funkcje rozkładów QR oraz generująca statystyki:

Algorytm obliczania wartości własnych metodą QR bez przesunięć ma nastąpującą postać:

$$A^{(k)} = Q^{(k)} R^{(k)} - faktoryzacja$$
$$A^{(k+1)} = R^{(k)} Q^{(k)}$$
$$k \to \infty, \quad A^{(k)} = V^{-1} A V = diag\{\lambda_i\}$$

Algorytm obliczania wartości własnych metodą QR z przesunięciami ma nastąpującą postać:

$$A^{(k)} - p_k I = Q^{(k)} R^{(k)}$$
$$A^{(k+1)} = R^{(k)} Q^{(k)} + p_k I$$

Za pk przyjmuje się najbliższą wartość własną podmacierzy 2x2 z prawego dolnego rogu macierzy A^(k). Po wyzerowaniu pozostałych elementów ostatniej kolumny i ostatniego wiersza otrzymuje się wartość własną znajdującą się w prawym dolnym roku macierzy. W następnym kroku bierze się pod uwagę macierz pomniejszoną o ostatnią kolumnę i wiersz.

Program

```
function [] = zadanie1()
%zadanie1 MNUM

SIZE = [5 10 20]'; %rozmiar macierzy
LOOPS = 30;     %ilosc zadanych wykonan algorytmow

for k=1:3
    ItersQR = 0;    %liczba iteracji algorytmem QR bez przesuniecia
    ItersQRS = 0;   %liczba iteracji algorytmem QR z przesunieciem (QRS)
    Teig = 0;       %czas wykonania algorytmu wbudowanego
    TQR = 0;        %czas wykonania algorytmu QR
    TQRS = 0;       %czas wykonania algorytmu QRS
    loopsQR = 0;    %ilosc poprawnie wykonanych iteracji alg. QR
    loopsQRS = 0;   %ilosc poprawnie wykonanych iteracji alg. QRS

    for i=1:LOOPS
       %DANE:
       A = rand(SIZE(k));
       %A = A'+A; %macierz symetryczna
       tol = 0.00001;
       imax = 2000;

       %funkcja matlabowa
       t_start = tic;
       [~, D] = eig(A);
       t_elapsed = toc(t_start);
       Teig = Teig + t_elapsed;
       eigens = diag(D);
       %disp(sort(eigens));
```

```
        %QR Z PRZESUNIECIAMI
        [eigens, iters, t_elapsed, valid] = eigvalqrshifts(A, tol,imax);
        if valid == 1
            loopsQRS = loopsQRS + 1;
            ItersQRS = ItersQRS + iters;
            TQRS = TQRS + t_elapsed;
        end

        %disp(sort(eigens));

        %QR BEZ PRZESUNIEC
        [eigens, iters, t_elapsed, valid] = eigvalqrnoshift(A, tol, imax);
        if valid == 1
             loopsQR = loopsQR + 1;
             ItersQR = ItersQR + iters;
             TQR = TQR + t_elapsed;
        end

        %disp(sort(eigens));
    end

    MeanQRI = ItersQR / loopsQR;
    MeanQRSI = ItersQRS / loopsQRS;
    MeanTeig = Teig / LOOPS;
    MeanTQR = TQR / loopsQR;
    MEANTQRS = TQRS / loopsQRS;

    fprintf(1, '++++++++++++++++++\n');
    fprintf(1, 'Rozmiar macierzy: %d\n', SIZE(k));
    fprintf(1, 'QR bez przesuniec:\n');
    fprintf(1, 'Srednia ilosc iteracji: %g\n', MeanQRI);
    fprintf(1, 'Udane wykonania: %d z %d\n', loopsQR, LOOPS);
    fprintf(1, 'Sredni czas: %g\n', MeanTQR);
    fprintf(1, 'QR z przesunieciami:\n');
    fprintf(1, 'Srednia ilosc iteracji: %g\n', MeanQRSI);
    fprintf(1, 'Udane wykonania: %d z %d\n', loopsQRS, LOOPS);
    fprintf(1, 'Sredni czas: %g\n', MEANTQRS);
    fprintf(1, 'Matlab eig() sredni czas: %g\n', MeanTeig);

end
end
```

Funkcja obliczająca wartości własne metodą QR z przesunięciami (na podstawie skryptu):

```
function [eigenvalues, iters, t_elapsed, valid] = eigvalqrshifts(A,tol,imax)
% A - macierz
% tol - tolerancja (górna granica wartoœci) elementów zerowanych
% imax - maksymalna liczba iteracji
% eigenvalues - wektor wartosci wlasnych
% iters - liczba wszystkich wykonanych iteracji algorytmu
% t_elapsed - czas dzialania funkcji
% valid - wykonanie funkcji bez przekroczenia imax
t_start = tic;
valid = 1;

n = size(A, 1);              %rozmiar macierzy
eigenvalues = diag(zeros(n));    %alokacja pamieci na wartosci wlasne

iters = 0;
INITIALsubmatrix = A;
for k=n:-1:2
    DK = INITIALsubmatrix(1:k, 1:k); %macierz poczatkowa dla wyznaczenia
```

```
    %jednej wartosci wl.
    i = 0;
    while i <= imax && max(abs(DK(k,1:k-1))) > tol
      ev = roots([1, -(DK(k-1,k-1)+DK(k,k)), DK(k-1,k-1)*DK(k,k)-DK(k,k-1)*DK(k-
1,k)]);
      % MAT = [a b]
      %       [c d]
      % 1*x^2 -(a+d)*x +a*d-c*b  <- tak wyglada rownanie dla macierzy 2x2
      %gdy trzeba wyznaczyc wartosc wlasna, ktora jest pierwiastkami
      %zerowymi tego rownania

      if abs(ev(1)-DK(k,k)) <= abs(ev(2)-DK(k,k))
          shift = ev(1); %przesuniecie - najblizsza DK(k,k) wartosc wlasna
                         %analizowanej macierzy 2x2
      else
          shift = ev(2);
      end
      DK = DK - eye(k)*shift;   %macierz przesunieta
      [Q1, R1] = qrmgs(DK);      %faktoryzacja QR
      DK = R1*Q1 + eye(k)*shift;%macierz przeksztalcona
      i = i+1;
      iters = iters + i;
    end

    if i > imax
          %display('qrshifts imax exceeded program terminated');
          valid = 0;
          break;
    end

    eigenvalues(k) = DK(k,k);
    if k>2
        INITIALsubmatrix=DK(1:k-1,1:k-1); %deflacja macierzy
    else
        eigenvalues(1) = DK(1,1);    %ostatnia wartosc wlasna
    end
end

t_elapsed = toc(t_start);
end
```

Rozkład QR bez przesunięć (również na podstawie skryptu MNUM):

```
function [eigenvalues, i, t_elapsed, valid] = eigvalqrnoshift(D, tol, imax)
% tol   - tolerancja (górna granica wartoœci) elementów zerowanych
% imax  - maksymalna liczba iteracji
% eigenvalues  - wektor wartosci wlasnych
% i - liczba wykonanych iteracji algorytmu
% t_elapsed   - czas dzialania funkcji
% valid - wykonanie funkcji bez przekroczenia imax
t_start = tic;
valid = 1;
i = 1;
while i <= imax && max(max(D-diag(diag(D)))) > tol
    [Q1, R1] = qrmgs(D);
    D = R1*Q1;  %macierz przekszta³cona
    i = i + 1;
end

if i > imax
    %error('imax exceeded program terminated');
    %display('qrnoshifts imax exceeded program terminated');
```

```
    valid = 0;
end

eigenvalues = diag(D);
t_elapsed = toc(t_start);
end
```

Rozkład QR zmodyfikowanym algorytmem Grama-Schmidta:

```
function [ Q, R ] = qrmgs( A )
%QRmGS wyznacza rozklad QR (waski) zmodyfikowany algorytmem Grama-Schmidta
%dla macierzy mxn (m>=n) o pelnym rzedzie, rzeczywistej lub zespolonej

[m n] = size(A);
Q = zeros(m,n);
R = zeros(n,n);
d = zeros(1,n);

%rozklad z kolumnami Q ortogonalnym (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 rozkladu (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
```

Wyniki:

Dla macierzy symetrycznych:

Rozmiar macierzy: 5
QR bez przesuniec:
Srednia ilosc iteracji: 44.1
Udane wykonania: 30 z 30
Sredni czas: 0.0062635
QR z przesunieciami:
Srednia ilosc iteracji: 12.8333
Udane wykonania: 30 z 30
Sredni czas: 0.00222852
Matlab eig() sredni czas: 7.62015e-005

Rozmiar macierzy: 10
QR bez przesuniec:
Srednia ilosc iteracji: 219.033
Udane wykonania: 30 z 30
Sredni czas: 0.0792535
QR z przesunieciami:
Srednia ilosc iteracji: 23.7667

Udane wykonania: 30 z 30
Sredni czas: 0.00567859
Matlab eig() sredni czas: 9.14711e-005

Rozmiar macierzy: 20
QR bez przesuniec:
Srednia ilosc iteracji: 558.615
Udane wykonania: 26 z 30
Sredni czas: 0.76092
QR z przesunieciami:
Srednia ilosc iteracji: 45.4667
Udane wykonania: 30 z 30
Sredni czas: 0.024997
Matlab eig() sredni czas: 0.000234243

Wyniki dla macierzy niesymetrycznych:

Rozmiar macierzy: 5
QR bez przesuniec:
Srednia ilosc iteracji: NaN
Udane wykonania: 0 z 30
Sredni czas: NaN
QR z przesunieciami:
Srednia ilosc iteracji: 25.9667
Udane wykonania: 30 z 30
Sredni czas: 0.00374033
Matlab eig() sredni czas: 0.000110391

Rozmiar macierzy: 10
QR bez przesuniec:
Srednia ilosc iteracji: NaN
Udane wykonania: 0 z 30
Sredni czas: NaN
QR z przesunieciami:
Srednia ilosc iteracji: 48.8667
Udane wykonania: 30 z 30
Sredni czas: 0.0109829
Matlab eig() sredni czas: 0.000156216

Rozmiar macierzy: 20
QR bez przesuniec:
Srednia ilosc iteracji: NaN
Udane wykonania: 0 z 30
Sredni czas: NaN
QR z przesunieciami:
Srednia ilosc iteracji: 92.8333
Udane wykonania: 30 z 30
Sredni czas: 0.0578659
Matlab eig() sredni czas: 0.000435633

Wnioski:
Użyte algorytmy uzyskały takie same wyniki co matlabowa fukncja eig. Rozkład QR bez

przesunięć nie poradzil sobie jednak z macierzami niesymetrycznymi, jak i z nie wszystkimi macierzami symetrycznymi. Rozkład QR z przesunięciami jest o wiele bardziej uniwersalny, jest również szybszy mimo większego nakładu obliczeniowego na jeden krok ze względu na większą zbieżność.

Zadanie 2.
Wyznaczyć metodą najmniejszych kwadratów funkcję wielomianową najlepiej aproksymującą dane. Wykorzystać rozwiązanie układu równań normalnych oraz rozkład macierzy QR.

Rozwiązania układu równań normalnych:

$$A^T A a = A^T y$$

LZNK metodą QR:

$$A^{(k)} = Q^{(k)} R^{(k)} - faktoryzacja$$
$$A^{(k+1)} = R^{(k)} Q^{(k)}$$
$$k \to \infty, \quad A^{(k)} = V^{-1} A V = diag\{\lambda_i\}$$

Program:

Algorytm rozwiązujący równanie i zwracający rozmiar błędu:

```
function [a, residue] = equasystem(n, x, y, method)
%equasystem
% n - zadany stopien wielomianu
% x - wektor argumentow
% y - wektor wartosci
% a - wektor wyznaczonych wspolczynnikow
% ren_norm - norma reszt

[m, ~] = size(x); %pobranie rozmiarow

A = zeros(m, n+1);
%display (A);
for i=1:m  %wiersze
    for j=0:n %kolumny
        A(i,n+1-j) = x(i)^(j); %wype³niamy odpowiednimi wartoœciami potegi x
    end
end

if strcmp(method,'qr')
    [Q, R] = qrmgs(A); %rozklad qr metoda g-s z poprzedniego zadania
    a = R \ (Q'*y);
elseif strcmp(method,'normal')
    a = (A'*A)\(A'*y); %rozwiazanie ukaadu równan
else
    error('no method selected');
end

res = zeros(m,1);
%obliczanie reszt
for i=1:m
    sum = 0;
    for j=1:n
        sum = sum + a(j)*x(i)^(n+1-j); % suma a*x^potega
    end
    sum = sum + a(n+1); %plus sta³a
    res(i) = y(i) - sum;
```

```
end
residue = norm(abs(res),Inf);
end
```

Wyniki:

Stopień wielomianu: 1
Residuum w układzie równań normalnych 20.7909
Residuum w rozkładzie qr 20.7909

Stopień wielomianu: 2
Residuum w układzie równań normalnych 6.75924
Residuum w rozkładzie qr 6.75924

Stopień wielomianu: 3
Residuum w układzie równań normalnych 0.633177
Residuum w rozkładzie qr 0.633177

Stopień wielomianu: 4
Residuum w układzie równań normalnych 0.659345
Residuum w rozkładzie qr 0.659345

Stopień wielomianu: 5
Residuum w układzie równań normalnych 0.467668
Residuum w rozkładzie qr 0.467668

Stopień wielomianu: 6
Residuum w układzie równań normalnych 0.560987
Residuum w rozkładzie qr 0.560987

Stopień wielomianu: 7
Residuum w układzie równań normalnych 0.560987
Residuum w rozkładzie qr 0.560987

Stopień wielomianu: 8
Residuum w układzie równań normalnych 0.452665
Residuum w rozkładzie qr 0.452665

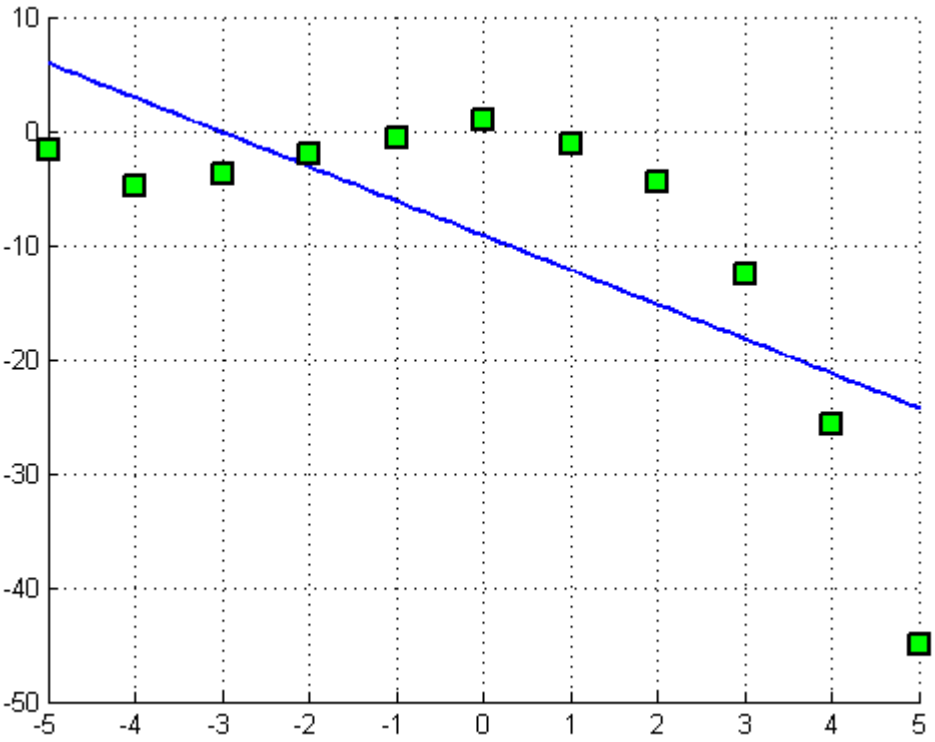
Stopień wielomianu: 9
Residuum w układzie równań normalnych 0.452665
Residuum w rozkładzie qr 0.452665

Stopień wielomianu: 10
Residuum w układzie równań normalnych 1.44736e-010
Residuum w rozkładzie qr 1.53845e-011

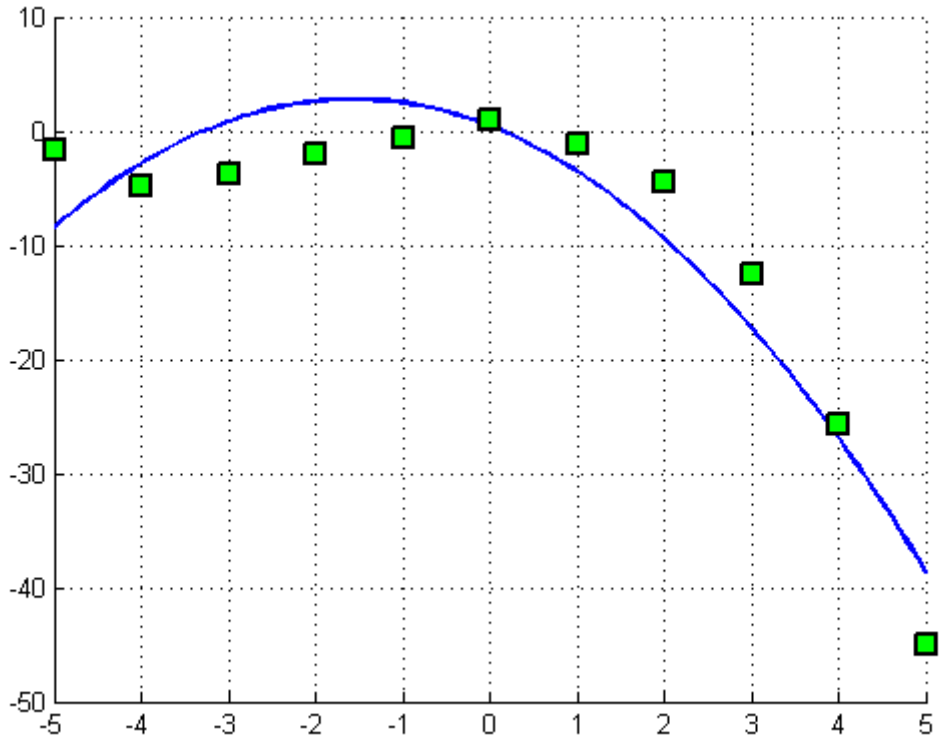
Stopień wielomianu: 11
Residuum w układzie równań normalnych 1.35455e-010
Residuum w rozkładzie qr 33.0655

Wykresy dla metody równań normalnych:

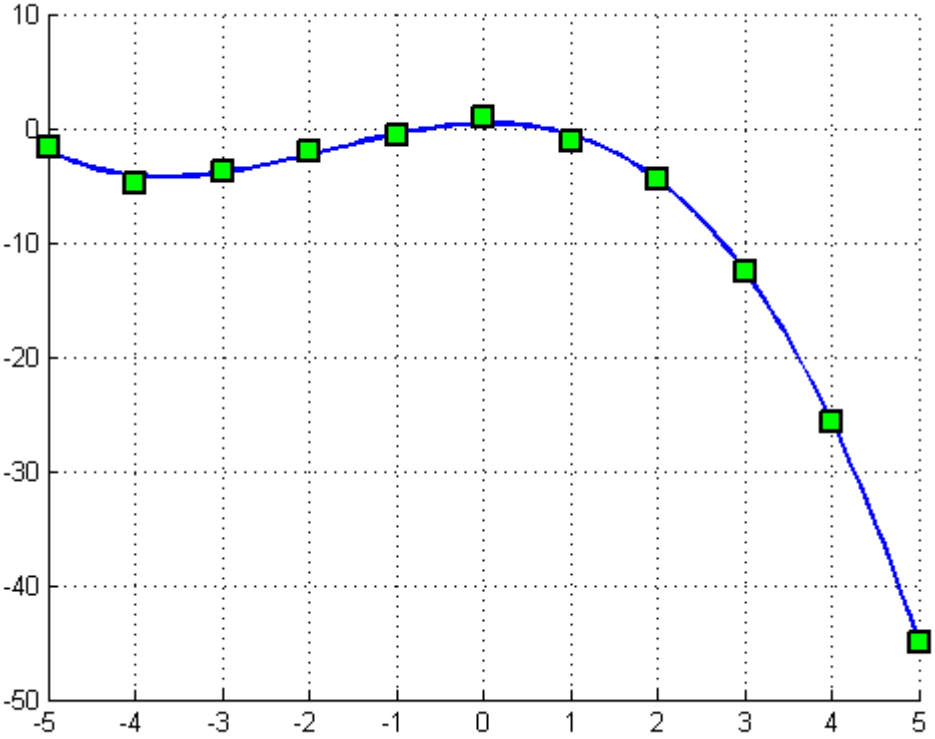
n=1



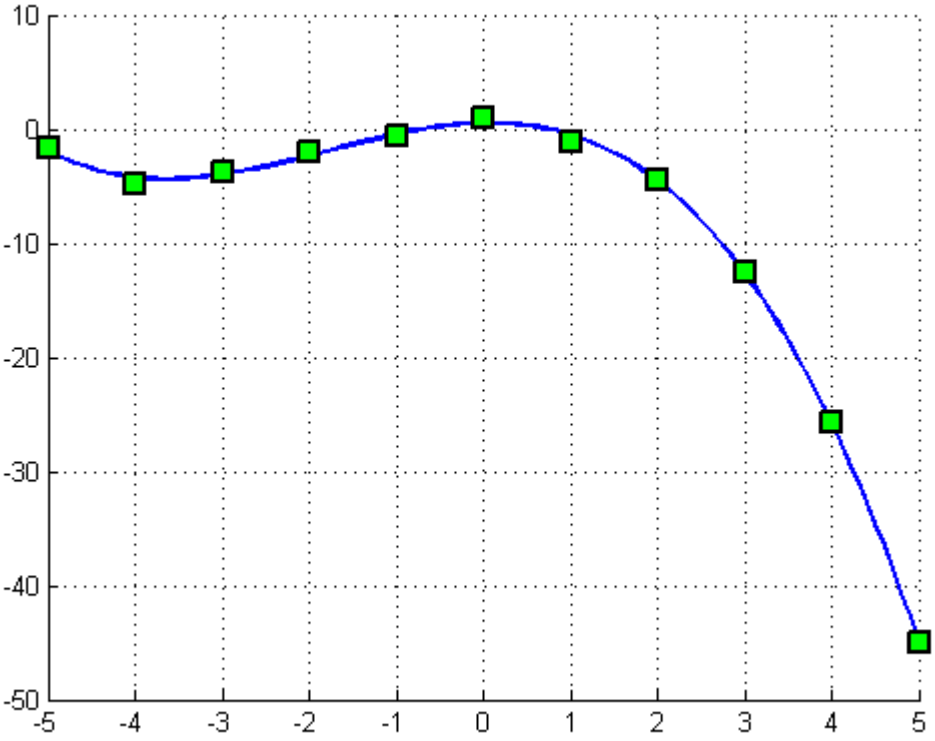
n=2



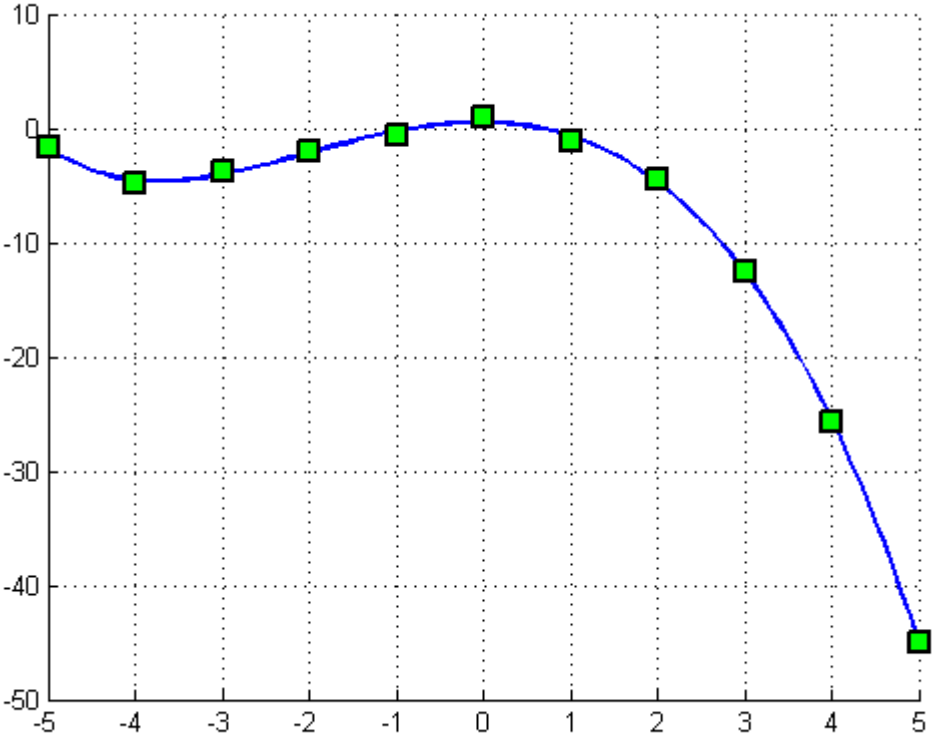
n=3



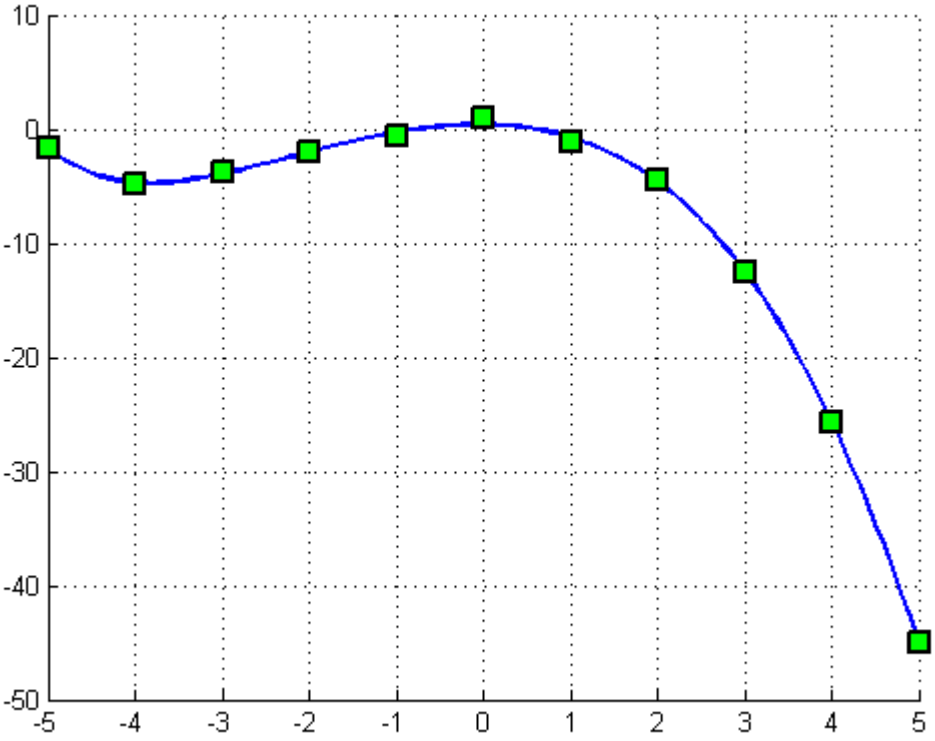
n=4



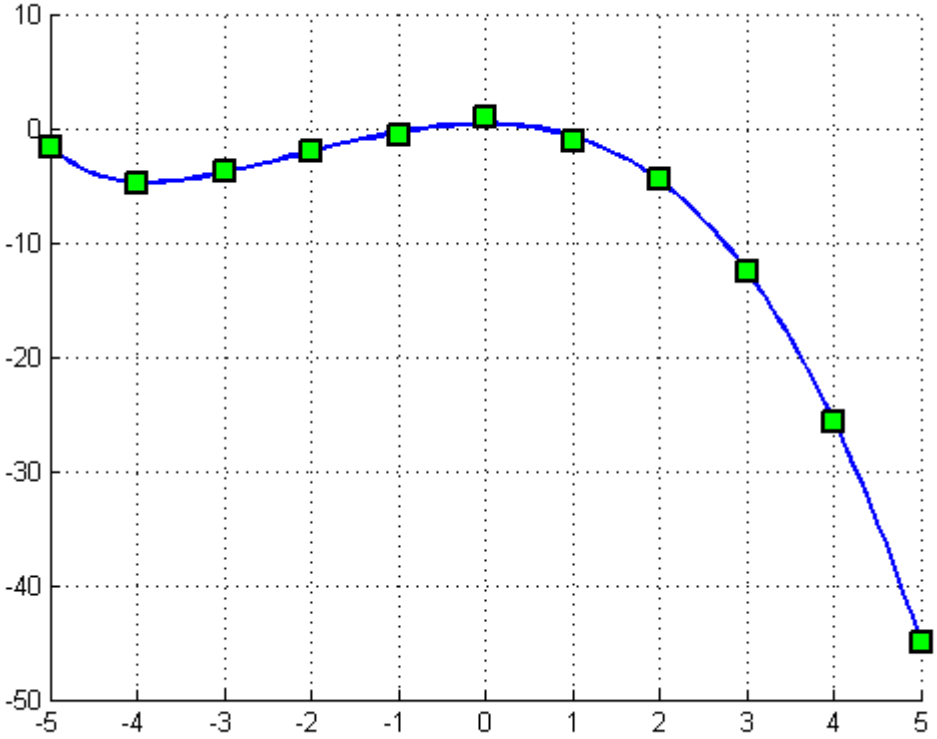
n=5



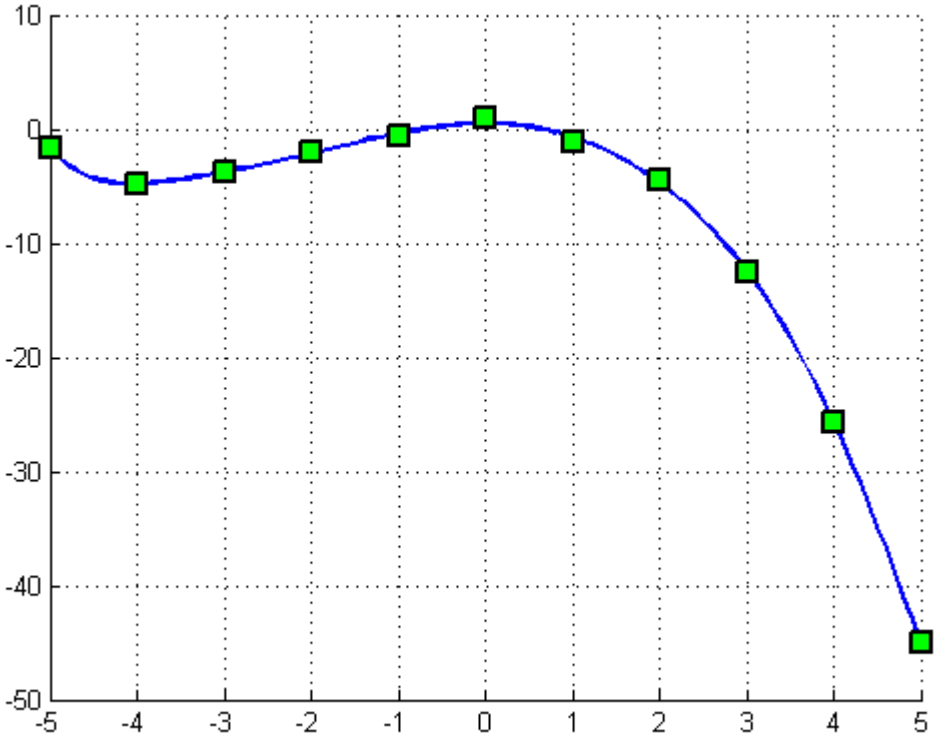
n=6



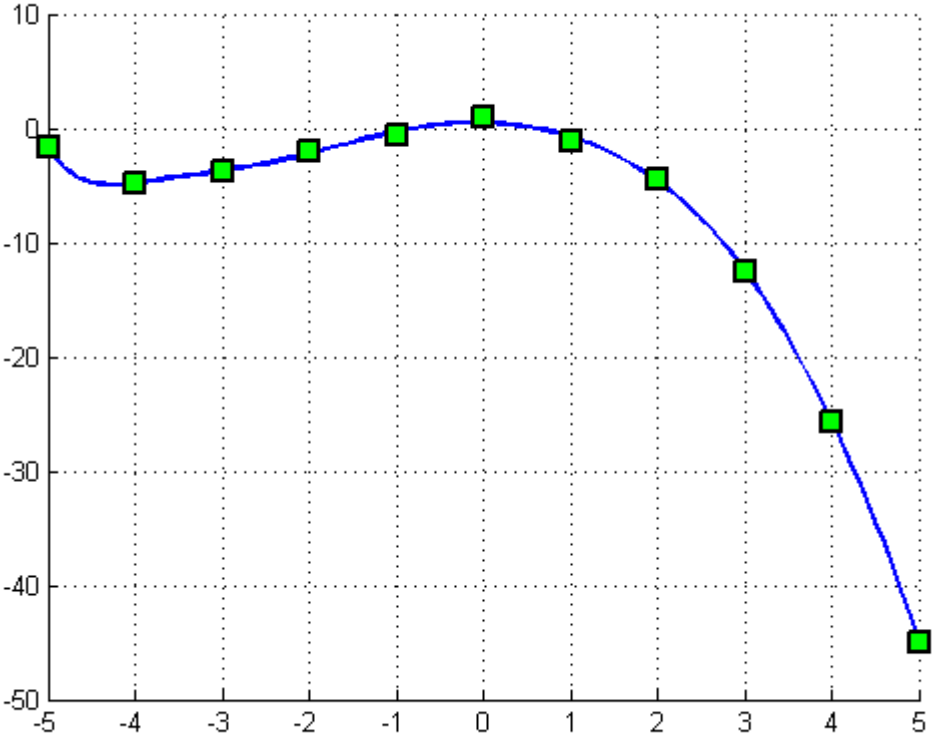
n=7



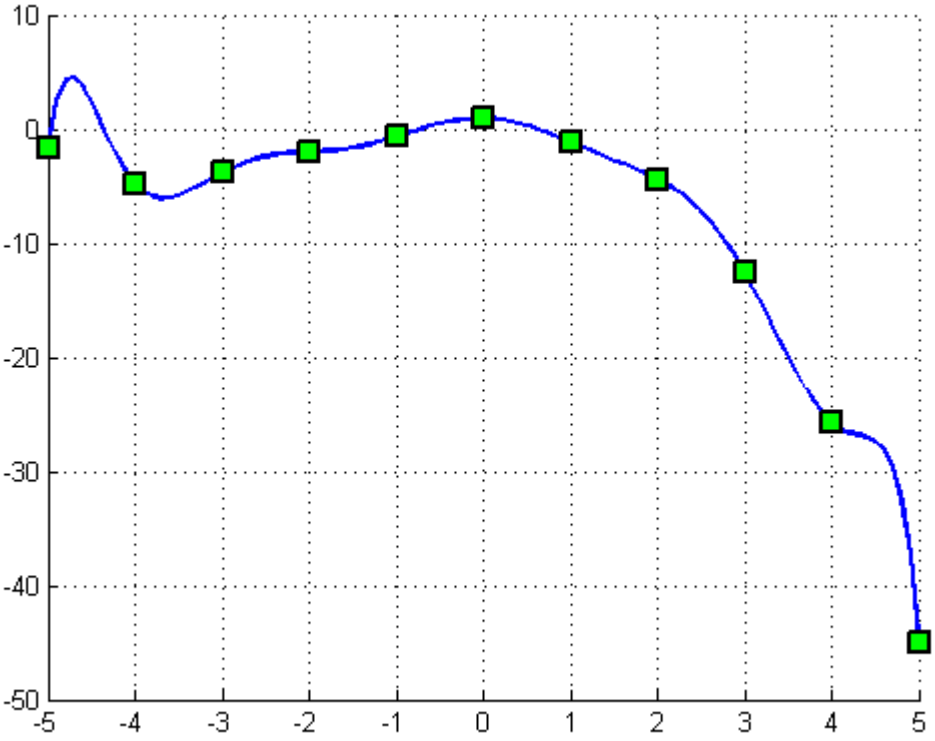
n=8



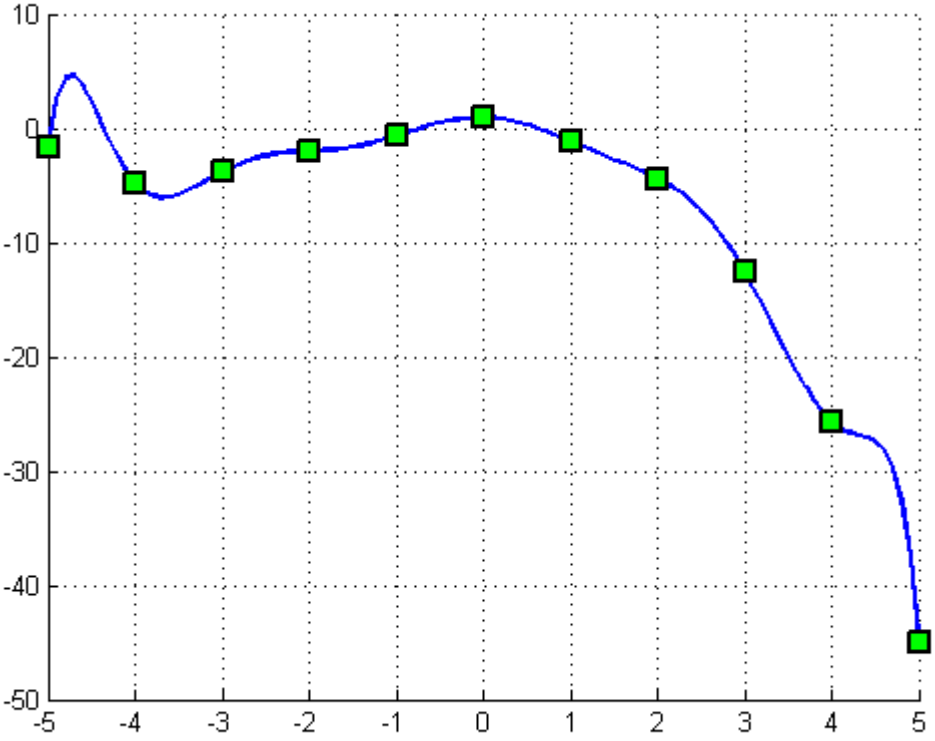
n=9



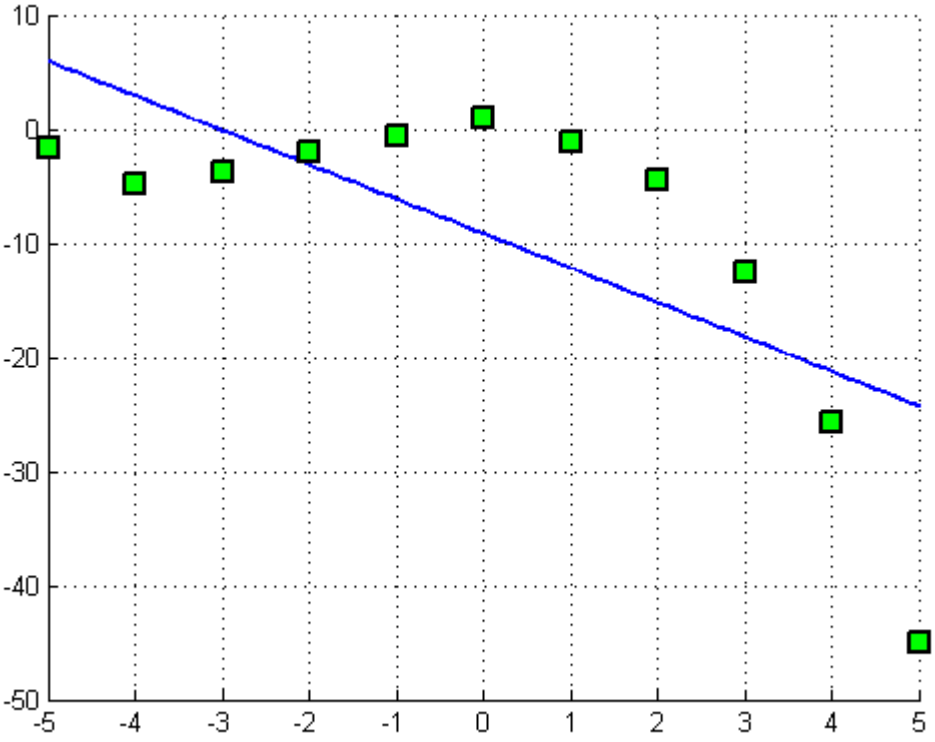
n=10



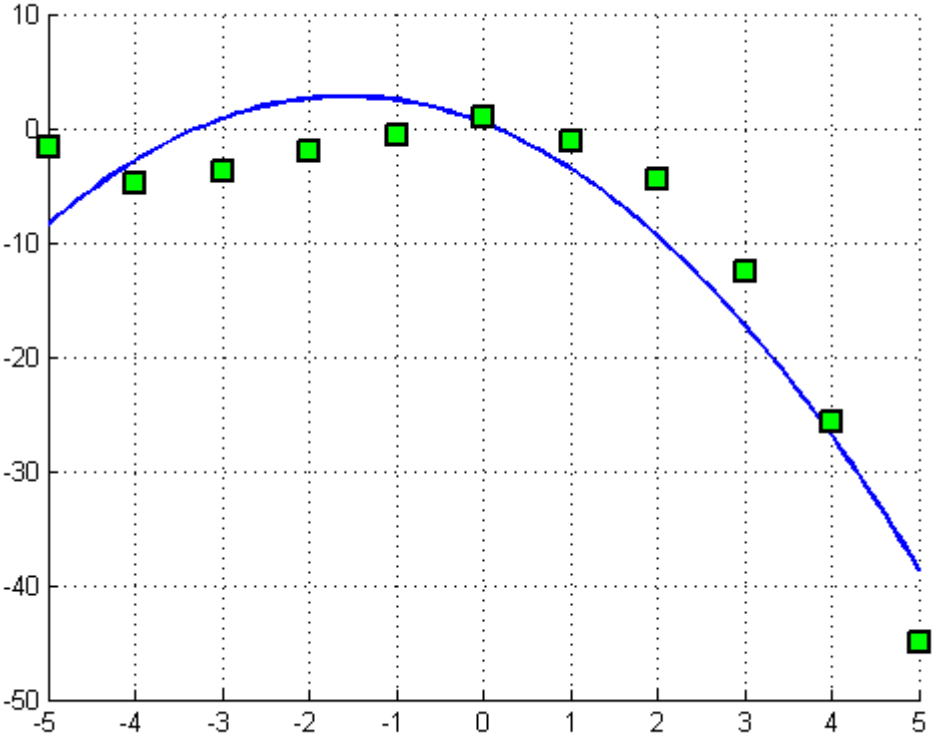
n=11



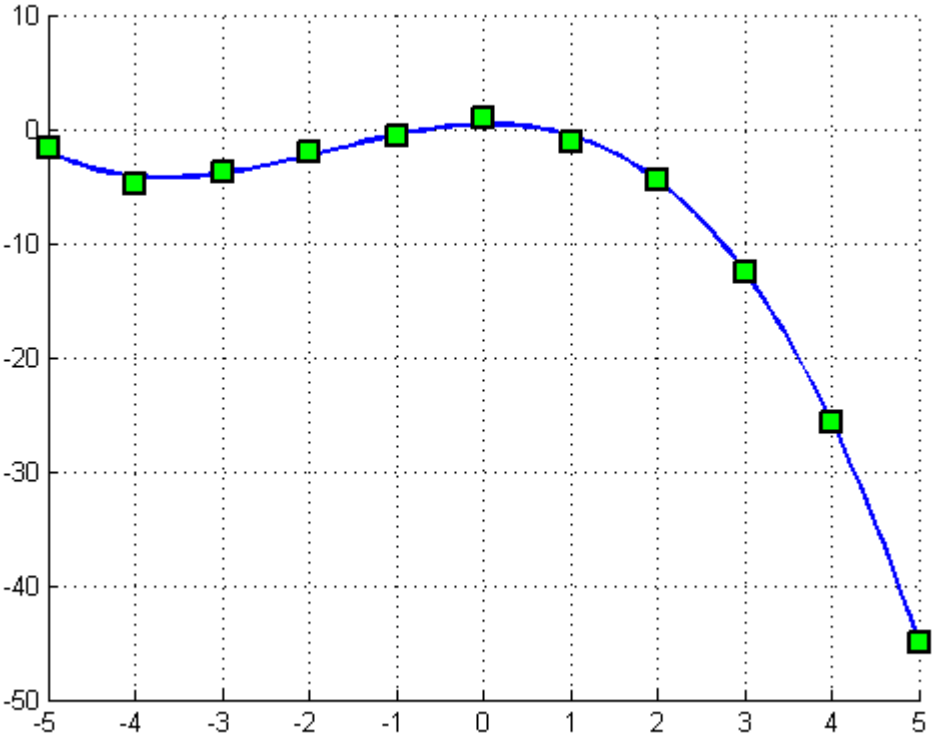
Metoda rozkładu QR
n=1



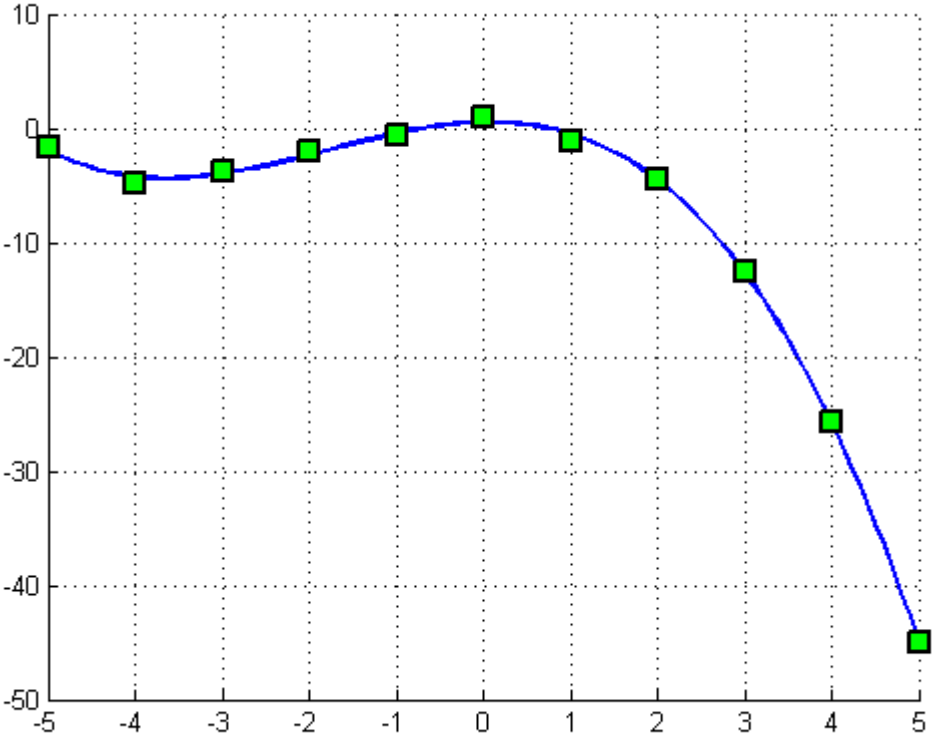
n=2



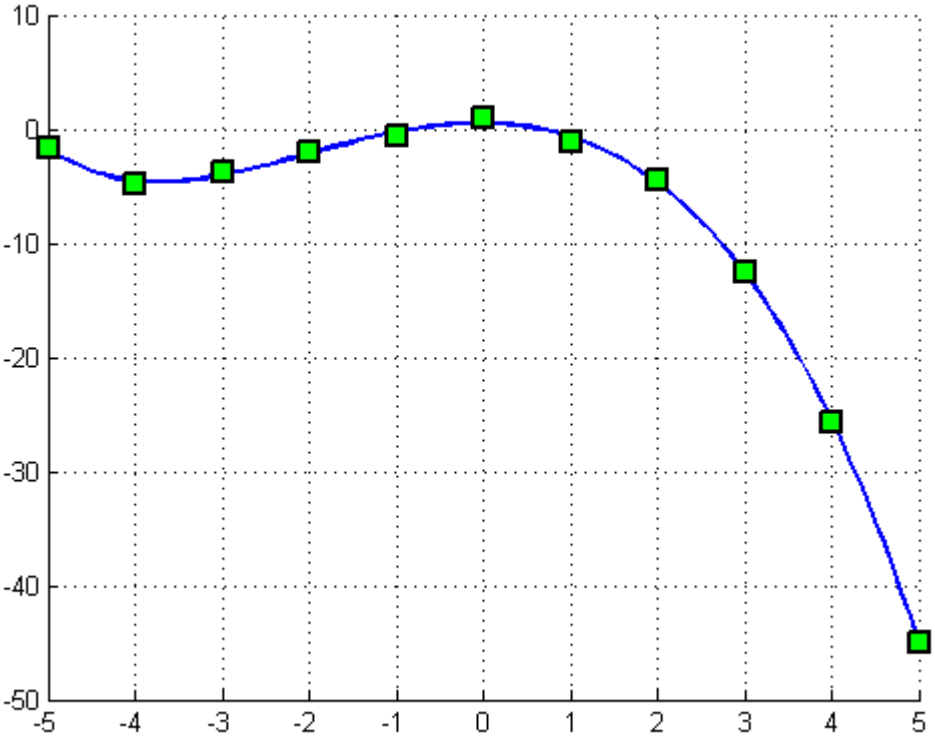
n=3



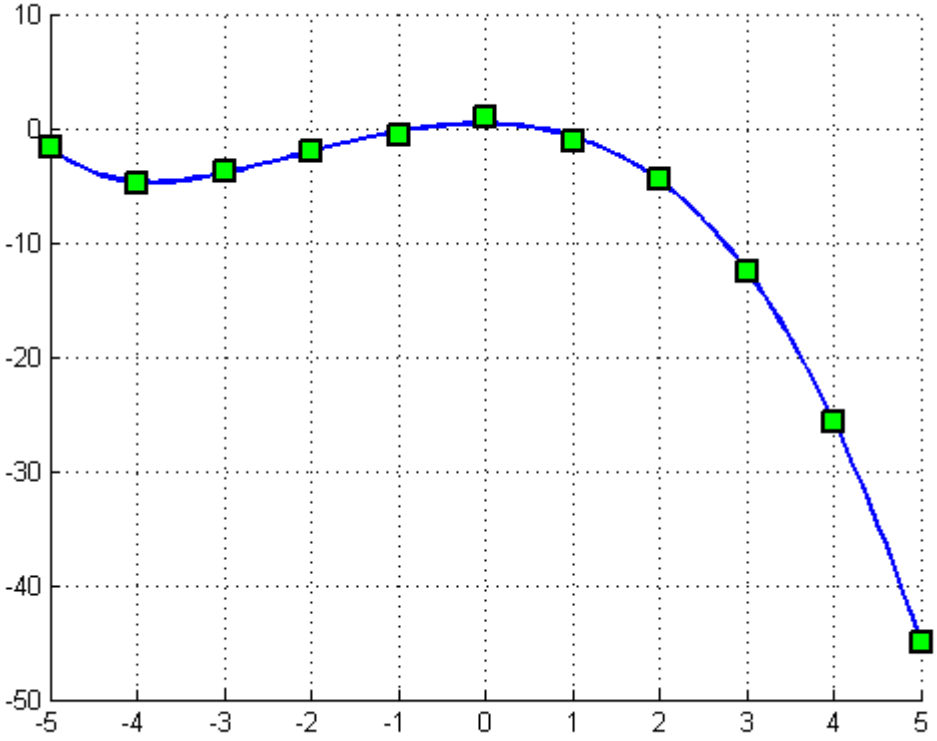
n=4



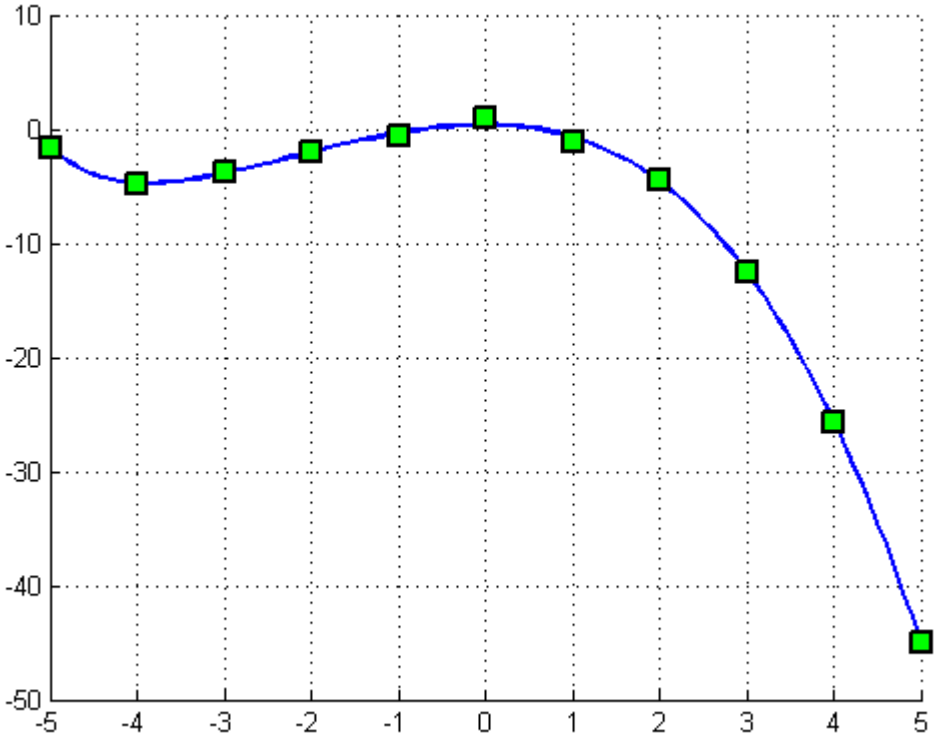
n=5



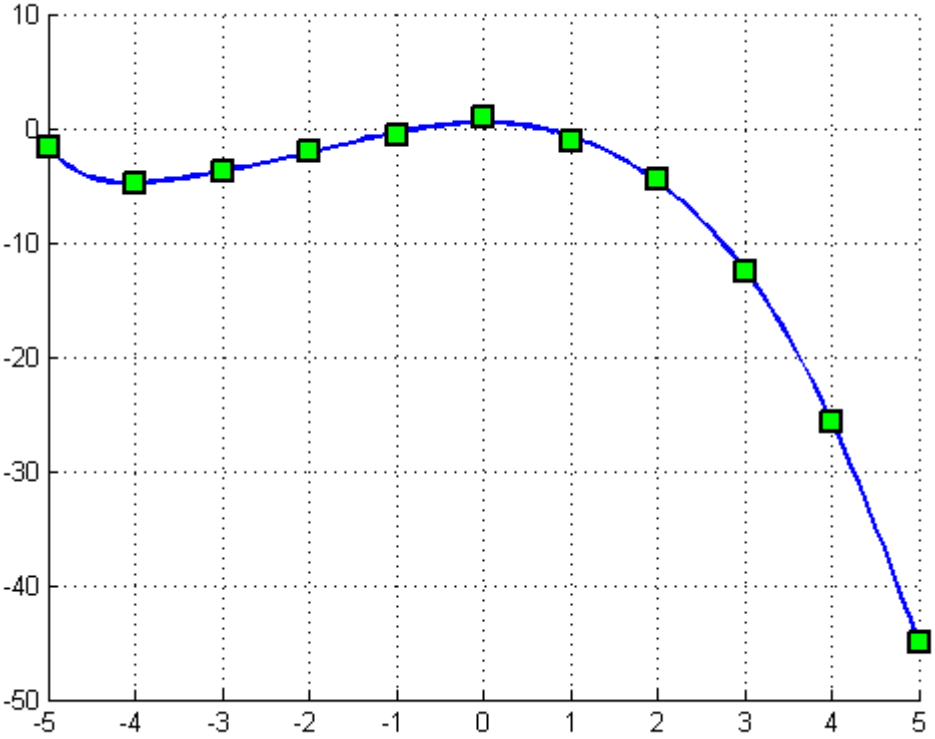
n=6



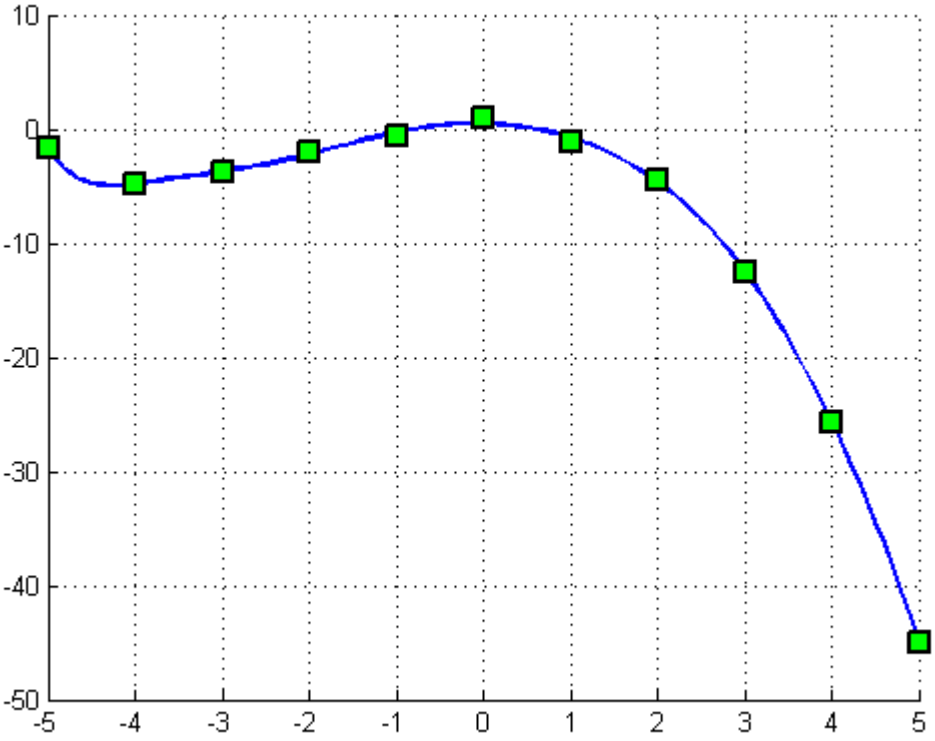
n=7



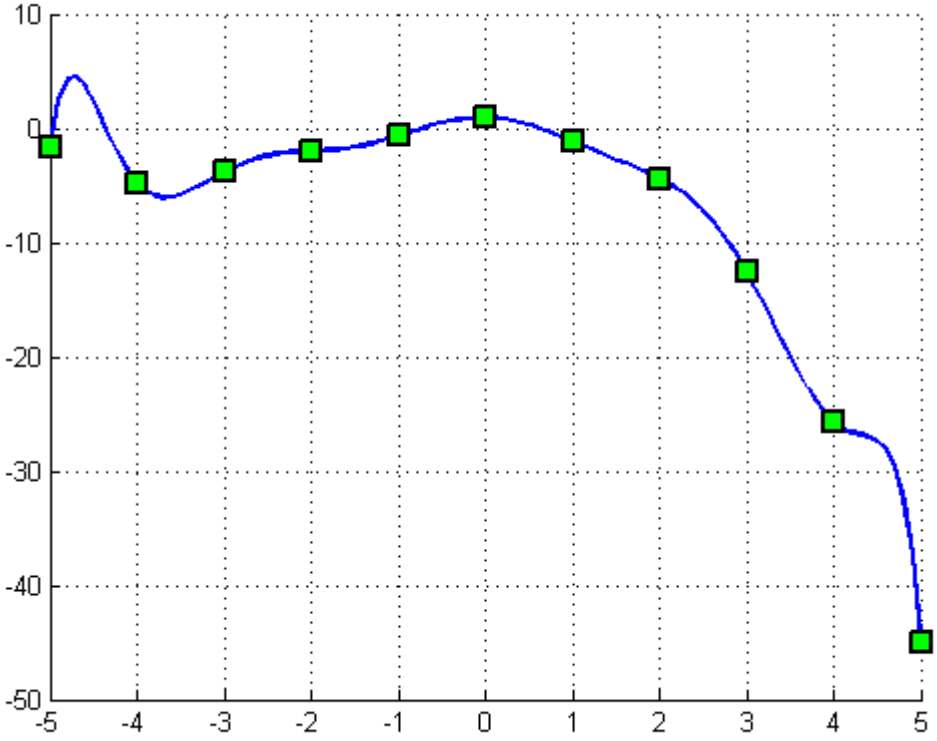
n=8



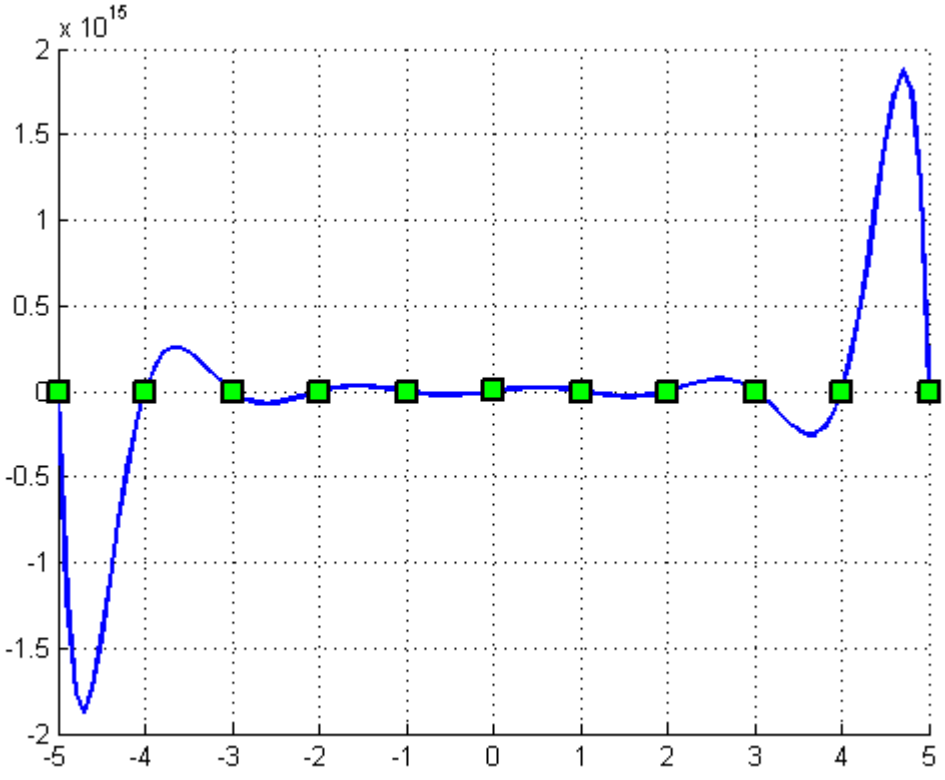
n=9



n=10


10
0
-10
-20
-30
-40
-50
-5
-4
-3
-2
-1
0
1
2
3
4
5

n=11


x 10^15
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-5
-4
-3
-2
-1
0
1
2
3
4
5

Wnioski:
Zarówno metoda równań normalnych jak i metoda rozkładu QR dobrze odwzoruje wielomian znany na podstawie próbek. Jednak razem ze wzrostem rzędu wielomianu powiększają się błędy numeryczne (macierz A szybko traci dobre uwarunkowanie), co jest widoczne w przypadku rozkładu QR dla n=11. Biorąc pod uwagę, że dane wejściowe zostały obarczone pewnym błędem, trzeba zauważyć, że zwiększając rząd wielomiany w pewnym momencie przestaje się aproksymować oryginalną funkcję, a zaczyna tę, która najbardziej pasuje do zebranych danych. Zarówno w przypadku metody układu równań normalnych jak i qr dla rozwiązań rzędu >= 10 obliczona funkcja znacznie odbiega od poprzednio uzyskanych przebiegów.