MNUM – Projekt 2.9

KrissKry xd

Spis treści

Zadanie 1.

Polecenie:

Metodą najmniejszych kwadratów należy wyznaczyć funkcję wielomianową y= f(x) najlepiej aproksymującą te dane (proszę przetestować wielomiany różnych stopni).

x_i	y_i
–5	–32,9591
–4	–20,7011
–3	–12,6986
–2	–5,1508
–1	–1,6893
0	0,1266
1	0,0743
2	–0,8709
3	–1,7371
4	–3,9952
5	–4,8987

Do rozwiązania proszę wykorzystać:

a) Układ równań normalnych
b) Układ równań liniowych z macierzą R wynikającą z rozkładu QR macierzy układu równań problemu.

Proszę obliczyć błąd aproksymacji w dwóch normach: euklidesowej oraz Czebyszewa(maksimum).

Układ równań normalnych

Opis rozwiązania:

Do rozwiązania problemu korzystam z podrozdziału 4.1.1 z książki prof. Piotra Tatjewskiego „Metody Numeryczne”. Przyjmuję bazę wielomianów naturalną (potęgową):

$$\varphi_0(x) = 1, \varphi_1(x) = x, \quad \dots, \varphi_n(x) = x^n$$

Moim celem jest uzyskanie układu równań normalnych opisanego wzorem:

$$\begin{array}{l} a_0 * g_{00} + a_1 * g_{10} + .. + a_n * g_{n0} = \rho_0 \\ a_0 * g_{01} + a_1 * g_{11} + .. + a_n * g_{n1} = \rho_1 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_0 * g_{0n} + a_1 * g_{1n} + .. + a_n * g_{nn} = \rho_n \end{array} \quad \equiv \quad G \cdot a = \rho$$

Dla którego:

$$g_{ik} \stackrel{\text{def}}{=} \sum_{j=0}^{N} (x_j)^{i+k}$$

$$\rho_k \stackrel{\text{def}}{=} \sum_{j=0}^{N} f(x_j)(x_j)^k$$

$$N - pożądany\ stopień\ wielomianu$$

Otrzymane współczynniki ρ po rozwiązaniu układu równań są współczynnikami wielomianu dla kolejnych potęg x.

Wielomian stworzony tą drogą będzie miał postać:

$$F(x) = a_0 + a_1 * x + a_2 * x^2 + \dots$$

Kod:

Tworzenie podstawowych macierzy

```
function [X, Y] = get_data()
    X = [ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 ];
    Y = [-32.9591, -20.7011, -12.6986, -5.1508, -1.6893, 0.1266, 0.0743, -
0.8709, -1.7371, -3.9952, -4.8987 ];
end
```

Algorytm liczenia współczynników wielomianu

```
function [coefficients] = get_coefficients_normal(X, Y, degree)
    tic

    %G * a = rho - układ równań normalnych
    rho = zeros(degree + 1, 1);
    G = zeros(degree + 1, degree + 1);
    samples = size(X, 2); % == 11

    %count rho
    for equation = 1:degree + 1
        sum = 0;
        for sample = 1: samples
            sum = sum + Y(sample) * (X(sample)^(equation-1));
        end
        rho(equation, 1) = sum;
    end

    %count G
    for row = 1: degree + 1
        for column = 1: degree + 1
            g = 0;
            for sample = 1: samples
                g = g + X(sample)^(row + column - 2);
            end
            G(row, column) = g;
        end
    end

    coefficients = linsolve(G, rho);
    %is now 1 + x + x^2 ...

    coefficients = flip(coefficients, 1);
    %is now ...x^2 + x + 1

    toc
end
```

Obliczenie odchylenia wielomianu od próbek

```
function [approx_error] = approx_error(X, Y, coefficients)

    coefficients = flip(coefficients, 1);
    %1, x, x^2, ...
    degree = size(coefficients, 1);
    approx_error = zeros( size(X) );

    for element = 1:size(X,2)
```

```
        sum = 0;
        for coeff = 1 : degree

            sum = sum + coefficients(coeff) * X(element)^(coeff - 1);

        end
        approx_error(element) = Y(element) - sum;
    end
end
```

Obliczenie normy euklidesowej

```
function [norm] = euclides_norm(approx_error)

    sum = 0;
    for err = 1:size(approx_error, 2)

        sum = sum + approx_error(err)^2;
    end

    norm = sqrt(sum);
end
```

Obliczenie normy maksymalnej

```
function [norm] = max_norm(approx_error)

    norm = 0;

    for err = 1:size(approx_error, 2)

        if abs(approx_error(err)) > norm
            norm = approx_error(err);

        end
    end
end
```

Rysowanie wykresu dla danych współczynników wielomianu

```
function plot_approximation(X, Y, coefficients)

    %próbkowanie wielomianu = 10x liczba próbek X
    sample_rate = size(X,2) * 10;

    %co jaką wartość x jedna próbka wielomianu
    x_step = X(end) / sample_rate;

    polynomial_x = X(1):x_step:X(end);
    polynomial_y = polyval(coefficients, polynomial_x);


    plot(X, Y, 'o', polynomial_x, polynomial_y);
    ylabel('y(x)');
    xlabel('x');
    title('Próbki');
    grid on;
end
```

Rozkład QR

Opis rozwiązania:

Z posiadanych macierzy X muszę najpierw stworzyć macierz A. Będzie to macierz z odpowiednimi potęgami wartości wektora danych x_i. Następnie zostanie ona rozłożona na macierze Q i R (ręcznie lub za pomocą wbudowanych w matlaba funkcji). By uzyskać współczynniki, należy wykorzystać równanie

$$Rx = Q^T \cdot b$$ co w moim przypadku sprowadza się do $$R * wsp = Q^T \cdot Y^T.$$

Macierz Y jest transponowana ze względu na bycie macierzą $1\ x\ n$ aniżeli na odwrót. Przekształcając, otrzymamy końcowy wzór na współczynniki wielomianu:

$$wsp = R^{-1} \cdot Q^t \cdot Y^t$$

Kod:

Algorytm liczenia współczynników dla rozkładu QR

```
function [coefficients] = get_coefficients_qr(X, Y, degree)

    samples = size(X, 2);
    A = zeros(samples, degree + 1);

    for row = 1:samples
        for column = degree + 1:-1:1
            A(row, column) = X(row) ^ (degree + 1 - column);
        end
    end

    % Dekompozycja QR na macierzy A o wymiarach m x n, taka, że A = QR.
    % R to macierz górna trójkątna o wymiarach m x n,
    % Q to macierz ortogonalna o wymiarach m x m.
    [Q, R] = qr(A);
    % [R, Q] = qrmgs(A); %nieużywana implementacja z podręcznika

    % Y = Y' bo moja macierz Y ma wymiar 1 x n
    coefficients = R \ (Q' * Y');
end
```

Wyniki

Wartości błędu w normie euklidesowej dla danych metod:

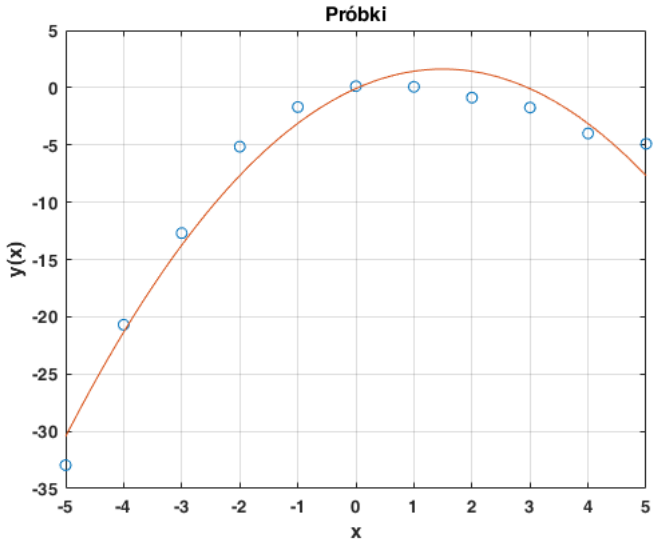
Stopień wielomianu	Układ równań normalnych	Rozkład QR – układ liniowy
1	23.0309	23.0309
2	5.8859	5.8859
3	1.2090	1.2090
4	1.1135	1.1135
5	1.09	1.09
6	1.0895	1.0895
7	0.6709	0.6709
8	0.6332	0.6332
9	0.2069	0.2069
10	1.4778e-11	8.0681e-14

Porównanie norm dla metody układu równań normalnych:

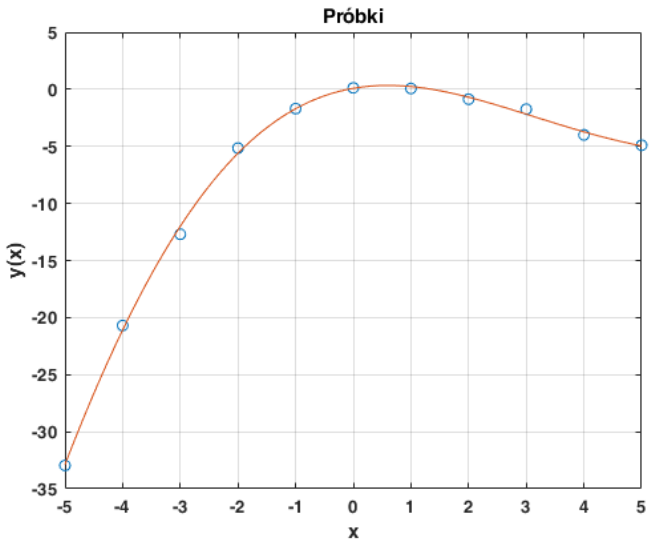
Stopień wielomianu	Norma euklidesowa	Norma maksymalna
1	23.0309	13.8985
2	5.8859	2.8066
3	1.2090	0.8845
4	1.1135	0.7174
5	1.09	0.6992
6	1.0895	0.69
7	0.6709	0.4406
8	0.6332	0.3560
9	0.2069	0.1213
10	1.4778e-11	9.7931e-12
11	Niedokładny pomiar	Niedokładny pomiar

Wykresy

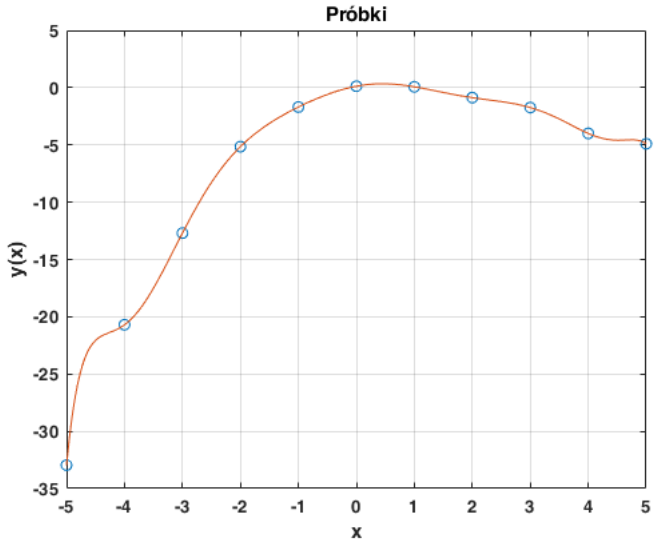
Stopnia drugiego



Stopnia piątego



Stopnia dziesiątego



Wnioski:

Zaimplementowane algorytmy aproksymacji wielomianów działają poprawnie w obu wariantach, dając dużą dokładność już dla wielomianu stopnia 5.

Szczególnym przypadkiem jest wielomian stopnia dziesiątego, w którym drastycznie spada błąd przybliżenia – jest to spowodowane tym, że dla 10 punktów i punktu x = 0, zawsze znajdzie się wielomian przechodzący przez nie. Co więcej, sam kształt krzywej odbiega od tych o niższym stopniu. Ponownie, wynika to z faktu, że funkcja jest przybliżana w danych punktach, a w reszcie przestrzeni argumentów, może zachowywać się dowolnie.

Minimalne odchylenia mogą być spowodowane szumem pomiarowym danych. Na podstawie normy wielomianu 10-ego stopnia można dojść do wniosku, że metoda wykorzystująca rozkład QR jest delikatnie lepsza w aproksymacji.

Efektywność algorytmów dla tak niewielkich próbek będzie zawsze wysoka – nie następują tu skomplikowane obliczenia złożone czasowo.