AKADEMIA GÓRNICZO-HUTNICZA

im. Stanisława Staszica w Krakowie

Wydział Inżynierii Mechanicznej i Robotyki
Katedra Systemów Energetycznych i Urządzeń Ochrony Środowiska

Metody Numeryczne

Laboratorium 2

Aproksymacja

dr inż. Krystian Szopa
KRAKÓW, 2020

1 Wprowadzenie do zajęć

Podstawy teoretyczne do zagadnienia **aproksymacji** znajdują się w skrypcie wykładowcy:
Czajka Ireneusz, Gołaś Andrzej, *Inżynierskie metody analizy numerycznej i planowanie eksperymentu*, Wydawnictwo AGH, Kraków 2017.
Skrypt jest dostępny w bibliotece AGH lub w wersji online poprzez stronę bilbioteki.

Student przed przystąpieniem do realizacji instrukcji powinien zapoznać się z zagadnieniem **aproksymacji**, przedstawionym w **rozdziale 8** skryptu. Szczególnie z wprowadzeniem do tego rozdziału.

Aproksymacja w tej instrukcji zostanie opisana dla konkretnych przykładów, a pełne zrozumienie zagadnienia wymaga wiedzy nabytej podczas wykładów i ze skryptu.

2 Aproksymacja wielomianowa

Aproksymacja polega na na zastąpieniu jednej funkcji (aproksymowanej) inną funkcją (aproksymującą) poprzez znalezienie jej opisu matematycznego. Jeżeli funkcja aproksymowana jest określona w całym przedziale to mówimy o aproksymacji funkcji ciągłej, natomiast jeżeli funkcja aproksymowana jest określona tylko na skończonym zbiorze punktów to mamy do czynienia z **aproksymacją funkcji dyskretnej**. To właśnie aproksymacją funkcji dyskretnej będziemy zajmować się na zajęciach. A skończony zbiór punktów, którymi opisana jest funkcja aproksymowana, będziemy nazywać **węzłami aproksymacji**.

Przykład 1:
Należy wylosować zbiór 7 równomiernie rozmieszczonych punktów w przedziale od **-50** do **70**, wartości funkcji powinny zawierać się w przedziale od 0 do 10. Następnie znaleźć wielomiany pierwszego, drugiego i czwartego stopnia aproksymujące te punkty w sensie metody najmniejszych kwadratów.

Zacznijmy od wylosowania węzłów aproksymacji i nanieśmy je na wykres.

```
clear all          %od tych trzech linijek
clc                %powinien zaczynać się każdy program
close all          %ja nie będę ich zapisywał w kolejnych przykładach

x=-50:20:70
y=10*rand(1,length(x))
plot(x,y,'*')
```

Zwróć uwagę, że po każdym uruchomieniem programu losowane są inne punkty,

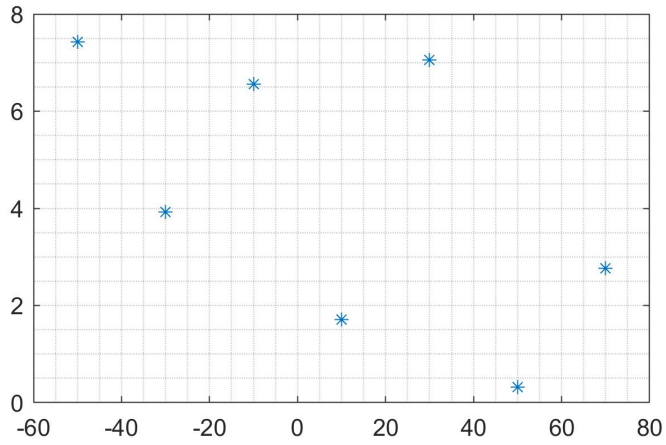
więc wykresy w instrukcji i u Ciebie będą się różniły. Ale nie ma to większego znaczenia. Ważny jest tok postępowania. U mnie wylosowane punkty mają następujące współrzędne:

```
x =

   -50   -30   -10    10    30    50    70

y =

    7.4313    3.9223    6.5548    1.7119    7.0605    0.3183    2.7692
```



Uwzględnienie w poleceniu **plot** `'*'`, pozwala narysować same węzły aproksymacji, bez łączenia ich linią co jest tutaj istotne.

Teraz musimy znaleźć taką funkcję **liniową** (ciągłą), która będzie aproksymowała te węzły w sensie metody najmniejszych kwadratów. Zanim dojdziemy do konkretnych rozwiązań, zastosujmy teorię w praktyce.
Funkcja liniowa ma ogólną postać:

$$y = a_1 + a_2 \cdot x \tag{1}$$

(lub $y = a_0 + a_1 \cdot y$, na poprzednich zajęciach jednak mówiłem, że ponieważ adresowanie w Matlabie jest od 1, to również będę indeksował od 1, co ułatwi implementację).

Teraz podstawiamy do powyższej zależności każdy z węzłów aproksymacji, otrzymując układ 7 równań. Zapis bardziej ogólny:

$$\begin{cases} y_1 = a_1 + a_2 \cdot x_1 \\ y_2 = a_1 + a_2 \cdot x_2 \\ y_3 = a_1 + a_2 \cdot x_3 \\ y_4 = a_1 + a_2 \cdot x_4 \\ y_5 = a_1 + a_2 \cdot x_5 \\ y_6 = a_1 + a_2 \cdot x_6 \\ y_7 = a_1 + a_2 \cdot x_7 \end{cases} \tag{2}$$

gdzie para (x_1, y_1) to współrzędne pierwszego węzła czyli w moim przypadku $(-50,\ 7.43)$. Podstawiając węzły aproksymacji, otrzymamy układ równań dla tego szczególnego przypadku:

$$\begin{cases} 7.43 = a_1 + a_2 \cdot (-50) \\ 3.92 = a_1 + a_2 \cdot (-30) \\ 6.56 = a_1 + a_2 \cdot (-10) \\ 1.71 = a_1 + a_2 \cdot 10 \\ 7.06 = a_1 + a_2 \cdot 30 \\ 0.32 = a_1 + a_2 \cdot 50 \\ 2.77 = a_1 + a_2 \cdot 70 \end{cases}$$

Teraz chcielibyśmy, znaleźć takie a_1 i a_2 żeby każde z tych równań zostało spełnione, czyli żeby każda z tych równości była prawdziwa. Wtedy wykres funkcji przechodziłby przez wszystkie punkty. Jednak mamy tylko 2 współczynniki funkcji a i aż 7 równań, więc znalezienie takiej funkcji liniowej, która spełniałyby te wszystkie równania jest niemożliwe. Wyjątkiem jest sytuacja, w której wszystkie punkty leżałyby w linii, ale jest to przypadek szczególny i w podejściu ogólnym (jak to) nie jest rozważany.

To co możemy w tej sytuacji zrobić to znaleźć taką funkcję liniową, która będzie najlepiej dopasowana do tych punktów w sensie metody najmniejszych kwadratów (MNK).

Zapiszmy powyższy układ równań w postaci macierzowej

$$X \cdot A = Y \tag{3}$$

natomiast macierz X oraz wektory A i Y mają następującą postać:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \\ 1 & x_6 \\ 1 & x_7 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} \tag{4}$$

Mam nadzieję, że przejście z postaci (9) do (14) jest zrozumiałe. W razie problemów krótkie wyjaśnienie. Patrzymy na prawą stroną pierwszego równania $a_1+a_2 \cdot x_1$ i wpisujemy do macierzy $\mathbf{X}$ kolejne parametry stojące przy współczynnikach a. Przy a_1 stoi wartość 1 (nie jest rozpisana w układzie (9)), a przy a_2 stoi wartość x_1. Następnie zapisujemy kolumnowy wektor $\mathbf{A}$, w którym znajdują się 2 nieznane współczynniki a. W pierwszym wierszu wektora $\mathbf{Y}$ zapisujemy to co jest po lewej stronie równania $y_1 = a_0 + a_1 \cdot x_1$, czyli y_1. Następnie powtarzamy czynność dla każdego równania (wektor $\mathbf{A}$ oczywiście zapisujemy tylko raz).

Zwróć uwagę, że liczba wierszy w macierzy $\mathbf{X}$ odpowiada liczbie równań oraz liczbie węzłów aproksymacji. Natomiast liczba kolumn jest równa liczbie niewiadomych parametrów a. Liczba elementów w wektorze $\mathbf{Y}$ oczywiście że też jest równa liczbie równań i węzłów.
Układ równań w którym liczba równań jest większa od liczby niewiadomych nazywamy **nadokreślonym**.

Podstawiając za x_k i y_k współrzędne węzłów aproksymacji otrzymujemy:

$$\begin{bmatrix} 1 & -50 \\ 1 & -30 \\ 1 & -10 \\ 1 & 10 \\ 1 & 30 \\ 1 & 50 \\ 1 & 70 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 7.43 \\ 3.92 \\ 6.56 \\ 1.71 \\ 7.06 \\ 0.32 \\ 2.77 \end{bmatrix} \tag{5}$$

czyli jak widać macierze $\mathbf{X}$ i $\mathbf{Y}$ są znane, jedyną niewiadomą jest wektor $\mathbf{A}$, który zawiera współczynniki poszukiwanej funkcji liniowej.

Po przekształceniach (patrz skrypt zależności od (8.1) do (8.10)) można wyznaczyć wektor $\mathbf{A}$ z zależności:

$$A = (X^T \cdot X)^{-1} \cdot X^T \cdot Y \tag{6}$$

a otrzymane parametry a będą współczynnikami poszukiwanej funkcji.

Zadanie 1: Należy dokończyć zadanie opisane w **Przykładzie 1** dla funkcji liniowej. Czyli zaimplementować do Matlaba to co opisano powyżej. Należy wprowadzić macierz $\mathbf{X}$, $\mathbf{Y}$, wyznaczyć $\mathbf{A}$. Dla obliczonych współczynników dorysować na wykresie funkcję liniową.

Wskazówki:
1. Wprowadź macierz $\mathbf{X}$ w taki sposób, żeby nie wpisywać ręcznie każdej wartości, ponieważ dla 1000 punktów nikt *z palca* tego wprowadzać nie będzie. Możesz wykorzystać np. pętlę **for** lub inny sposób.
2. Wprowadź wektor $\mathbf{Y}$ (dla swoich wylosowanych punktów), zwróć uwagę, że wygląda tak samo jak wektor $\mathbf{y}$ (ten z wartościami funkcji z początku zadania), z tą

różnicą, że jest wektorem kolumnowym. Wektor wierszowy na kolumnowy (i na odwrót) można zamienić operacją transpozycji.
3. Zapisz powyższą zależność (6) na wyznaczenie wektora **A**. Zwróć uwagę, że wykonywane są tutaj operacje MACIERZOWE! Nie pomyl ich z tablicowymi.
Uruchom program.

Wyświetl zawartość obliczonej macierzy **A**. U mnie będzie to:

```
A =

    4.6220
   -0.0369
```

Są to współczynniki funkcji liniowej, więc podstawiając je do zależności (1), otrzymamy:

$$y = 4.622 - 0.037x \tag{7}$$

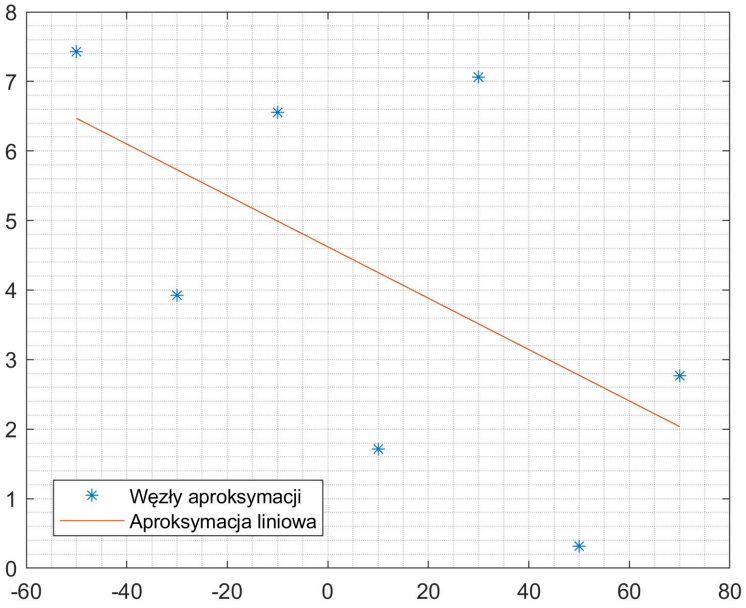
Ostatnią kwestią jest naniesienie tej prostej na wykres. Tutaj należy zrobić trzy rzeczy.
1. Wprowadzić zagęszczony wektor argumentów. Ponieważ chcemy uzyskać dokładne rozwiązanie i wykres (chcemy wiedzieć *co dzieje się* pomiędzy punktami), wprowadzamy wektor również w zakresie od **-50** do **70**, ale z mniejszym krokiem np. **1** (zamiast 20). **Uwaga!** Nazwij ten wektor **inaczej** niż **x**, ponieważ nadpiszesz sobie ten pierwszy wektor, a jeszcze będzie nam potrzebny. Może się nazywać np. **xx**.
2. Wyznaczyć wartości tej funkcji liniowej dla tego nowego wektora **xx**. Wektor z wartościami nazwij inaczej niż **y**, np. **w1**. Czyli musisz zapisać zależność tej funkcji liniowej. Nie wpisuj współczynników **a** *ręcznie*, tylko adresuj je z wektora **A**, np. **A(1)**.
3. Dorysuj ten wykres funkcji liniowej do węzłów.

```
plot(x,y,'*',xx,w1)
legend('Węzły aproksymacji','Aproksymacja liniowa')
```

W poleceniu **plot** możemy zapisywać parami wektory wykresów (plus ewentualnie w ' ' ich parametry). Należy pamiętać że te pary wektorów muszą mieć taką samą długość. Czyli tutaj **x** i **y** mają po 7 elementów, natomiast **xx** i **w1** po 121. Dodatkowo w przypadku kilku wykresów należy opisywać je legendą.

Uwaga. Ponieważ punkty za każdym razem są losowane, to z każdym uruchomieniem programu ten wykres będzie wyglądał inaczej!

Jeżeli masz problemy z samodzielnym rozwiązaniem możesz wspomóc się odpowiedziami, na końcu instrukcji. Gdy masz problem np. tylko z wprowadzeniem macierzy **X** (najczęstszy problem) to sprawdź tylko ten jeden fragment, a resztę próbuj robić samodzielnie. Jeżeli zrozumiesz to zadanie, to przyswojenie reszty będzie DUŻO łatwiejsze.



Zadanie 2: Dokonaj aproksymacji węzłów wielomianem drugiego stopnia i dorysuj krzywą do wykresu.

Zdobyte umiejętności powinny być wystarczające żeby zrobić to samodzielnie, wiec zastanów się jak to rozwiązać. Jeżeli jeszcze nie czujesz się pewnie skorzystaj z poniższych wskazówek.

Interpretacja matematyczna.
Postać wielomianu 2 stopnia:

$$y = a_1 + a_2 \cdot x + a_3 \cdot x^2 \tag{8}$$

Układ równań dla węzłów aproksymacji:

$$\begin{cases} y_1 = a_1 + a_2 \cdot x_1 + a_3 \cdot x_1^2 \\ y_2 = a_1 + a_2 \cdot x_2 + a_3 \cdot x_2^2 \\ \vdots \\ y_7 = a_1 + a_2 \cdot x_7 + a_3 \cdot x_7^2 \end{cases} \tag{9}$$

Układ równań w postaci macierzowej:

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ & \vdots & \\ 1 & x_7 & x_7^2 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_7 \end{bmatrix} \tag{10}$$

Wskazówki:
Zmiana w stosunku do poprzedniego zadania, polega przede wszystkim na tym, że

w macierzy **X** pojawiła się dodatkowa kolumna wyrazów x_k^2, a w wektorze **A** dodatkowy, nieznany współczynnik a_3. Reszta jest analogiczna do poprzedniego zdania. Jeżeli nie chcesz nadpisywać poprzednich macierzy **X** oraz **A**, nadaj im nowe zmienne.

Wykres rysujemy dla tego samego wektora **xx** (nie ma potrzeby wprowadzania nowego), natomiast wartości funkcji kwadratowej w nowym wektorze (np. **w2**), są obliczane na podstawie wyznaczonych parametrów. Nie zapomnij o operatorach tablicowych.

Zadanie 3: Przeprowadź aproksymację wielomianem czwartego stopnia i dorysuj krzywą do wykresu.

Wykonaj samodzielnie zadanie i przejdź dalej.

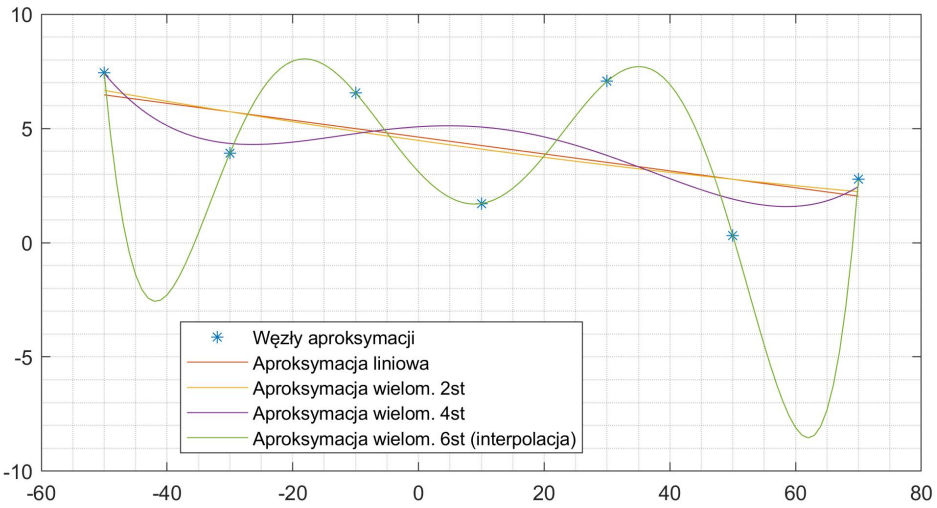
We wszystkich powyższych przykładach liczba równań była większa od liczby niewiadomych współczynników funkcji (układy równań były nadokreślone). Im większy stopień wielomianu, tym funkcja była lepiej dopasowana do węzłów w sensie MNK. Jednak w ogólnym przypadku wykres funkcji nie przechodził przez węzły. Jeżeli doprowadzimy do sytuacji, w której liczba niewiadomych parametrów a będzie równa liczbie równań, to wyznaczymy współczynniki funkcji, które będą spełniały wszystkie równania układu. Tym samym wykres funkcji będzie przechodził przez wszystkie węzły i będziemy mieli do czynienia z **interpolacją**.

Jeżeli liczba węzłów aproksymacji wynosi **N**, to w ogólnym przypadku wielomian stopnia **N-1** będzie interpolował te węzły.

Zadanie 4: Przeprowadź aproksymację wielomianem szóstego stopnia i dorysuj krzywą do wykresu.

Ostatecznie wykres powinien przedstawiać się mniej więcej jak na rys poniżej. Można zauważyć że w moim przypadku funkcja kwadratowa jest bardzo zbliżona do liniowej, o czym świadczy również mała wartość współczynnika $a_3 = 0.000098$ jednomianu kwadratowego.

Wielomian szóstego stopnia, jest interpolacją dyskretnej funkcji aproksymowanej i przechodzi przez wszystkie węzły.



Rysunek 1: Aproksymacja wielomianami 1, 2, 4 i 6 stopnia

2.1 Aproksymacja poleceniami Matlaba

Program Matlab jest rozbudowanym narzędziem i ma wiele zaimplementowanych funkcji. Jedną z tych funkcji jest aproksymacja wielomianowa, taka sama jaką przeprowadziliśmy wcześniej.

Wpisz w Command Window:

```
help polyfit
```

Jak sama nazwa **polyfit** wskazuje (*poly* od wielomian, *fit* – dopasuj), funkcja służy do dopasowania wielomianu do węzłów.

Zostanie wyświetlona pomoc. Najważniejszy jest pierwszy akapit. Można odczytać, że do polecenia **polyfit** należy podać 3 parametry. **X** i **Y** to wektory ze współrzędnymi węzłów, a **N** to stopień wielomianu jakim chcemy aproksymować te węzły.

Funkcja zwraca wektor współczynników **P**, który jest odpowiednikiem naszego wektora **A**. Różnica polega na tym, że w naszych zadaniach przy jednomianie najwyższego stopnia stał **ostatni** współczynnik w wektorze **A**. Natomiast tutaj **pierwszy** wyraz wektora **P** stoi przy jednomianie o najwyższym wykładniku. Czyli wektory **A** i **P** są względem siebie odwrócone.

Przykład 2: Dla punktów z **Przykładu 1**, przeprowadzić aproksymację wielomianem drugiego stopnia z wykorzystaniem funkcji **polyfit** i porównać współczynniki oraz wykresy z aproksymacją kwadratowW z **Przykładu 1**.

W tym momencie nasz kod programu (nowy skrypt) dla aproksymacji wielomianem drugiego stopnia wygląda mniej więcej tak:

```
x=-50:20:70
y=10*rand(1,length(x))

Y=y'
X2=[ones(length(x),1) x' x'.^2]
A2=inv(X2'*X2)*X2'*Y

xx=-50:70
w2=A2(1)+A2(2)*xx+A2(3)*xx.^2

plot(x,y,'*',xx,w2)
legend('Węzły aproksymacji','Aproksymacja kwadratowa Przykład 1')
grid minor
```

Teraz wykorzystajmy funkcję **polyfit**:

```
p2=polyfit(x,y,2)
```

Parametry **x** i **y** to współrzędne naszych punktów. Zwróć uwagę, że pomocy są wielkie litery X i Y, natomiast my wstawiamy małe ponieważ są to symbole dla wektorów z naszymi współrzędnymi. Więc w funkcji musi być zachowana tylko kolejność parametrów (względem *pomocy*), a nie konkretne symbole. Gdyby nasze węzły były zapisane w wektorach **a** i **b** to wówczas funkcja miałaby postać: *polyfit(a,b,2)*.

Pod zmienną **p2** zapisany jest wektor ze współczynnikami funkcji. Sprawdźmy jego wyrazy (można go transponować, żeby też był wektorem kolumnowym) i porównajmy z wektorem **A2**.

```
p2 =

   -0.0011
    0.0378
    5.2489

A2 =

    5.2489
    0.0378
   -0.0011
```

Wylosowano nowe punkty względem **Przykładu 1**, więc wynik też jest inny niż wcześniej. W każdym razie widać to o czym wspomniano wcześniej. Wartości współczynników są takie same, z tym że wektory są odwrócone względem siebie.

Obliczmy wartości funkcji dla parametrów otrzymanych poleceniem **polyfit**. Można to zrobić analogicznie jak wcześniej (należy pamiętać, żeby indeksować elementy od końca wektora). Można też wykorzystać polecenie **polyval**

```
help polyval
```

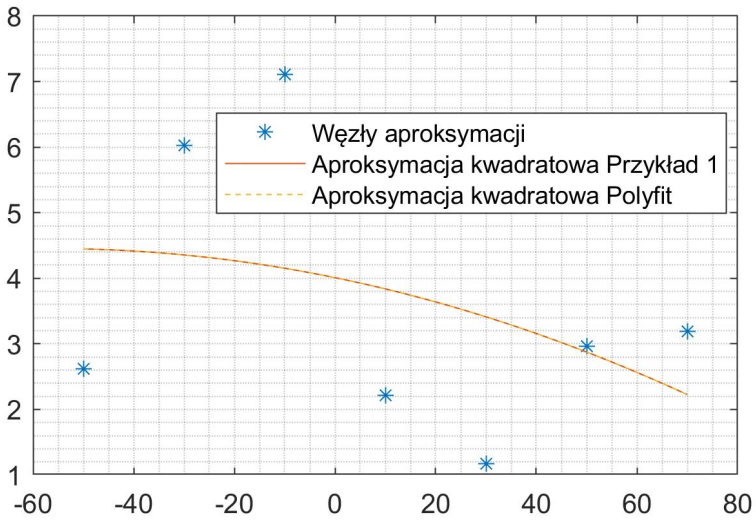
czyli wartości wielomianu. Jeżeli w miejsce pierwszego parametru podstawimy wektor współczynników **p2**, a na drugim miejscu zagęszczony wektor **xx**, to otrzymamy wartości funkcji. Czyli:

```
f=polyval(p1,xx)
```

Teraz wystarczy dorysować wykres

```
plot(x,y,'*',xx,w2,xx,f,'--')
legend('Węzły aproksymacji','Aproks. Przykład 1','Aproks. Polyfit')
grid minor
```

Wykres nowej funkcji rysuję linią przerywaną, ponieważ oba wykresy nakładają się na siebie i dla linii ciągłej byłby widoczny tylko jeden wykres.



Uwaga. Pamiętaj że wektory współczynników w **Przykładzie 1** są odwrócone względem tych z funkcji **polyfit**. Dlatego stosując polecenie **polyval** do obliczenia wartości funkcji w **Przykładzie 1** rozwiązanie będzie nieprawidłowe! Musisz najpierw odwrócić wektor współczynników poleceniem **flip()**.

3 Aproksymacja innymi funkcjami

Wielomian uogólniony możemy zapisać w postaci:

$$Q(x) = a_1 q_1(x) + a_2 q_2(x) + \ldots + a_j q_j(x) + \ldots + a_m q_m(x) \tag{11}$$

czyli:

$$Q(x) = \sum_{j=1}^{m} a_j q_x(x) \tag{12}$$

gdzie $q_j(x)$ jest funkcją bazową.
W poprzednich zadaniach naszymi funkcjami bazowymi były jednomiany:

$$q_1 = 1, \quad q_2 = x, \quad q_3 = x^2, \quad \ldots \quad q_m = x^{m-1}$$

jednak równie dobrze funkcjami bazowymi mogą być funkcje trygonometryczne, logarytmiczne, wymierne, itd.

Zadanie 5: Wprowadzić podane węzły aproksymacji i przeprowadzić aproksymację funkcją logarytmiczną w postaci:

$$y = a_1 + a_2 \cdot \ln(x) \tag{13}$$

Węzły aproksymacji:

```
x=125:125:625
y=[80 36.56 26.34 24.33 20.23]
```

Student powinien rozwiązać zadanie samodzielnie. W razie problemów poniżej zamieszczono kilka wskazówek.
Uwaga. Logarytm naturalny w Matlabie zapisujemy poleceniem **log( )**

Wskazówki:
1. Podstawowa różnica pomiędzy **Zadaniem 1**, a **Zadaniem 5** polega na zmianie funkcji bazowej.
W zadaniu 1 było:

$$q_1 = 1, \quad q_2 = x$$

W zadaniu 5 jest:

$$q_1 = 1, \quad q_2 = \ln(x)$$

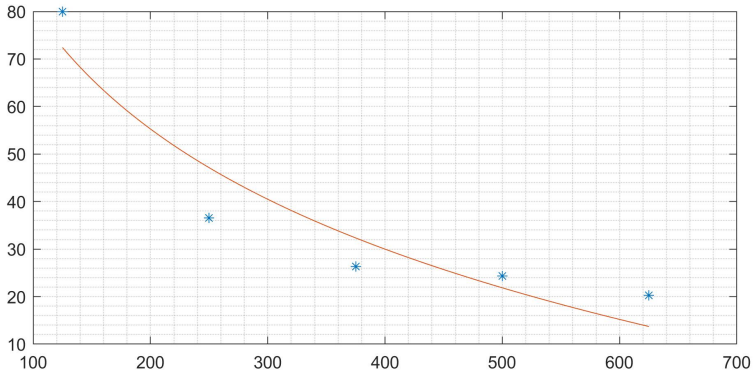
2. Względem **Zadania 1** zmieni się postać macierzy **X**:

$$X = \begin{bmatrix} 1 & \ln(x_1) \\ 1 & \ln(x_2) \\ 1 & \ln(x_3) \\ 1 & \ln(x_4) \\ 1 & \ln(x_5) \end{bmatrix} \tag{14}$$

Zwróć uwagę, że liczbą węzłów jest inna niż w Zadaniu 1, więc długość wektora **Y** również się zmieni. Przypominam, że logarytrm naturalny wpisujemy do programu poleceniem **log( )**.
3. Parametry macierzy **A** wyznaczamy tak samo jak wcześniej. Natomiast obliczając wektor z wartościami funkcji należy pamiętać, że mają być wyliczone dla nowej postaci funkcji (a nie dla wielomianu z Zadania 1):

$$y = a_1 + a_2 \cdot \ln(x) \tag{15}$$

4. Na wykres mają być naniesione węzły i krzywa funkcji logarytmicznej.



Zadanie 6: Dla tych samych punktów przeprowadź aproksymację funkcją logarytmiczną w postaci:

$$y = a_1 + a_2 \cdot \ln(x) + a_3 \cdot \ln(x).^2 \tag{16}$$

Dorysuj krzywą do wykresu.
Pamiętaj o operatorze tablicowym przy podnoszeniu do kwadratu.

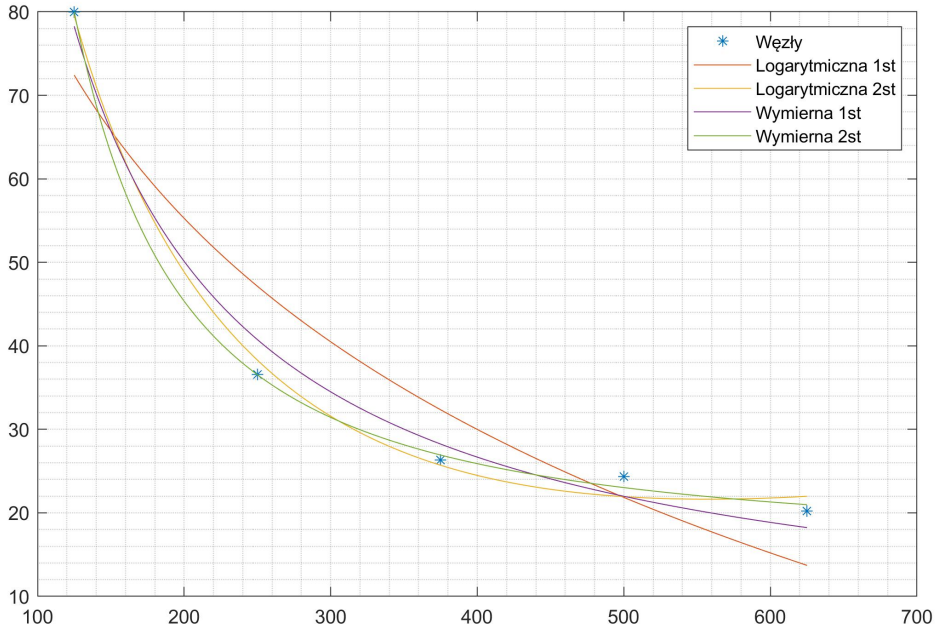
Do rozwiązania zadania należy wykorzystać wiedzę z zadań 5 i 2.

Zadanie 7: Przeprowadź aproksymację funkcjami wymiernymi w postaci:

$$y = a_1 + a_2 \frac{1}{x} \tag{17}$$

$$y = a_1 + a_2 \frac{1}{x} + a_3 \left(\frac{1}{x}\right)^2 \tag{18}$$

Uwaga. Podczas dzielenia przez wektor **x** pamiętaj aby wykonać to tablicowo `1./x`



4 Miary błędów i dopasowania funkcji

Po znalezieniu funkcji aproksymujących węzły, chcielibyśmy odpowiedzieć na pytanie: czy wyznaczone funkcje są dobrze dopasowane do tych węzłów? Oczywiście wstępnie *na oko*, można stwierdzić że w powyższych zadaniach funkcje drugiego stopnia są lepiej dopasowane niż pierwszego, ale inżynier powinien określić to ilościowo. Tym bardziej, że w większości przypadków ocena wizualna nie jest taka prosta i oczywista.

Podstawową miarą błędu jest błąd bezwzględny, czyli różnica pomiędzy wartością odczytaną z wyznaczonej funkcji aproksymującej $\hat{y}_k$, a wartością w węźle y_k.

$$\Delta y_k = \hat{y}_k - y_k \tag{19}$$

Graficzną interpretację tych błędów można znaleźć w skrypcie na Rys. 8.1.

Omówmy to na przykładzie **Zadania 5.**
Wyznaczmy błąd bezwzględny w pierwszym punkcie, dla funkcji logarytmicznej pierwszego stopnia. Wartość w pierwszym węźle (dla $x = 125$) to:

$$y_1 = 80$$

natomiast wartość odczytana z wykresu funkcji aproksymującej (też dla $x = 125$) to:

$$\hat{y}_1 = 72.42$$

Jest to pierwszy element w wektorze w którym są zapisane wartości funkcji logarytmicznej. W moim przypadku jest to zmienna **f**, więc możemy to ostatecznie w Matlabie zapisać:

```
D(1)=f(1)-y(1)
```

czyli: $\Delta y_1 = -7.58$

No dobrze to obliczmy teraz błąd dla drugiego węzła, czyli dla ($x = 250$). Wybór wartości dla węzła jest prosty, ponieważ jest to drugi element w wektorze **y**, czyli:

$$y_2 = 36.56$$

Jednak w przypadku funkcji aproksymującej wyznaczyliśmy wartości dla wektora zagęszczonego. Oznacza to, że drugi element w wektorze **f** jest wartością dla argumentu $x = 126$, a nie 250. Musimy określić indeks argumentu 250 w wektorze **xx** i dopiero zaadresować odpowiednią wartość. Argument $x = 250$ znajduje się w wektorze **xx** pod adresem **126** (Sprawdź!).
Teraz możemy obliczyć błąd bezwzględny dla drugiego punktu:

```
D(2)=f(126)-y(2)
```

czyli: $\Delta y_2 = 10.57$
Oczywiście znając krok w wektorze zagęszczonym, nie musimy za każdym razem szukać argumentu w wektorze **xx**, żeby zaadresować wartość funkcji **f**. Skoro różnica pomiędzy kolejnymi interesującymi nas punktami w wektorze **xx** wynosi 125, a krok jest stały i równy 1, to adresy będą różniły się o:

$$\frac{x_{k+1} - x_k}{h} = 125$$

Czyli wartości funkcji odpowiadające węzłom można wyznaczyć:

```
f(1), f(126), f(251), f(326), f(521)
```

Zadanie 8: Napisz program, który utworzy pięcioelementowy wektor **D**, w którym zapisane będą wartości bezwzględne błędów dla tych 5 węzłów aproksymacji. Program ma być zautomatyzowany, tak aby można go było szybko wykonać np. dla 1000 punktów. Możesz użyć do tego pętli **for** lub bezpośrednio adresować elementy wektorów innymi wektorami.
Uwaga! Częstym błędem popełnianym w tym zadaniu jest mylenie adresu/indeksu wektora z wartością elementu, która pod nim stoi.

Oczywiście są też inne miary błędów obliczane dla punktów np. błąd względny. Jednak w przypadku np. 1000 punktów dostaniemy 1000 wartości, które ciężko poddać szybkiej analizie. Najlepiej by było gdybyśmy mogli określić jakość dopasowania wykresu za pomocą jednej liczby. Taką miarą jest pierwiastek błędu średniokwadratowego **RMSE** (*ang.* root mean square error) i jest wyrażony zależnością:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\hat{y}_k - y_k\right)^2} \tag{20}$$

Opiszmy operacje jakie musimy wykonać żeby obliczyć RMSE dla naszego przykładu:
1. Pierwszą czynnością jest wykonanie obliczenia w nawiasie ( ). Jest to błąd bezwzględny, obliczony dla wszystkich **N** punktów. Czyli dokładnie to co zrobiliśmy przed chwilą.
2. Kolejną operacją jest podniesienie do kwadratu błędu, dla każdego punktu z osobna. Czyli w naszym przypadku, podnosimy do kwadratu każdy element w wektorze **D**.
3. Sumowanie. Musimy po prostu dodać do siebie wszystkie podniesione do kwadratu błędy.
4. Uśredniamy wynik biorąc pod uwagę liczbę punktów. Czyli dzielimy liczbę którą otrzymaliśmy po sumowaniu przez N (czyli u nas $N = 5$). Robimy to po to, żeby wynik nie był mocno uzależniony od liczby punktów, które wzięliśmy pod uwagę.
5. Z całości obliczamy pierwiastek.

Zadanie 9: Zaimplementuj miarę **RMSE** do programu, wykorzystując powyższy opis i oblicz błąd dla funkcji logarytmicznej pierwszego stopnia.

Wynik: RMSE= 7.13

Popularnym wskaźnikiem opisującym dopasowanie funkcji do węzłów jest **współczynnik determinacji**, z którym na pewno spotkasz się również na zajęciach projektowych. Jest on opisany zależnością:

$$\mathrm{R}^2 = 1 - \frac{\sum\limits_{k=1}^{N} (\hat{y}_k - y_k)^2}{\sum\limits_{k=1}^{N} (y_k - \bar{y})^2}, \tag{21}$$

1. Licznik wyrażenia jest sumą kwadratów błędów bezwzględnych, czyli to samo co mieliśmy w przypadku RMSE.
2. W mianowniku $\bar{y}$ jest średnią arytmetyczną z wartości y_k.

W przypadku idealnym, gdy funkcja przechodzi przez wszystkie punkty, błąd bezwzględny dla każdego z nich jest równy **0**, czyli licznik wyrażenia również jest równy **0** i tym samym $\mathrm{R}^2 = 1$. Ze zrostem błędów wartość wskaźnika będzie spadać. Im wartość R^2 bliższa **1** tym funkcja jest lepiej dopasowana do punktów.

Zadanie 10: Zaimplementuj współczynnik R^2 i oblicz jego wartość dla funkcji logarytmicznej 1 stopnia.

Wynik: $\mathrm{R}^2 = 0.894$

Zadanie 11: Oblicz RMSE oraz R^2 dla pozostałych funkcji z rozdziału 3. Wybierz najlepiej dopasowaną funkcję

5 Odpowiedzi do zadań

Zadanie 1: Należy dokończyć zadanie opisane w **Przykładzie 1** dla funkcji liniowej. Czyli zaimplementować do Matlaba to co opisano powyżej. Należy wprowadzić macierz **X**, **Y**, wyznaczyć **A**. Dla obliczonych współczynników dorysować na wykresie funkcję liniową.

1. Wprowadzenie macierzy **X**:
Macierz **X** możemy automatycznie wypełniać elementami z wykorzystaniem pętli **for**, albo nawet za pomocą dwóch zagnieżdżonych pętli **for**.

W Matlabie możemy to zrobić sprawniej wstawiając od razu całe kolumny do macierzy X:

```
X=[ones(length(x),1) x']
```

Najpierw do Macierzy **X** wstawiamy całą kolumnę jedynek. Czyli korzystamy z polecenia **ones**, liczba wierszy jest równa liczbie węzłów (program sam to sprawdzi poleceniem **length**. Musimy też podać liczbę kolumn tej tablicy, czyli jedna kolumna.

Następnie wstawiamy cały wektor **x**. Jednak uwaga, ponieważ **x** jest zapisany jako wiersz musimy go obrócić do kolumny, robimy to poleceniem transpozycji.

2. Wprowadzenie wektora **Y**: Jest to dokładnie ten sam wektor co w treści zadania, więc jedynie trzeba go transponować do postaci kolumny: `Y=y'`

3. Wyznaczenie wektora **A:** Należy zaimplementować podaną zależność. Należy pamiętać, że są to operacje macierzowe:

```
A=inv(X'*X)*X'*Y
```

4. Utworzenie zagęszczonego wektora i obliczenie wartości funkcji na podstawie wyznaczonych współczynników:

```
xx=-50:70
w1=A(1)+A(2)*xx
```

Krok w wektorze **xx** wynosi 1, może być inny np. 2 albo 0.1. Musi być tylko wystarczająco mniejszy względem tego między węzłami. Wartości funkcji obliczane są dla nowego wektora **xx**, a współczynniki wybierane bezpośrednio z wektora **A**. Oczywiście wprowadzono nowe symbole **xx** oraz **w1** żeby nie nadpisać dotychczasowych wyników. **w1** jest pełną nazwą zmiennej i NIE oznacza pierwszego elementu w wektorze **w**. Żeby wybrać pierwszy element z tego wektora należałoby zapisać `w1(1)`

5. Rysowanie wykresu zostało w całości opisane pod zadaniem.

Zadanie 2: Dokonaj aproksymacji węzłów wielomianem drugiego stopnia i dorysuj krzywą do wykresu.

Do macierzy **X** wprowadź 3cią kolumnę, nie zapomnij o transpozycji i operatorze tablicowym. Oblicz wartości funkcji według zależności na funkcję kwadratową. Pamiętaj aby nadać nowe symbole i nie nadpisać poprzednich wyników. Dorysuj wykres.

```
X2=[ones(length(x),1) x' x'.^2]
A2=inv(X2'*X2)*X2'*Y
w2=A2(1)+A2(2)*xx+A2(3)*xx.^2

plot(x,y,'*',xx,w1,xx,w2)
legend('Węzły aproksymacji','Aproks. liniowa','Aproks. wielom. 2st')
grid minor
```

Wektory **Y** oraz **xx** są takie same jak wcześniej więc nie musisz ich wprowadzać po raz kolejny.

Zadania 3 i 4.
Zadania należy wykonywać analogicznie do dwóch poprzednich. Ich samodzielne wykonanie nie powinno sprawić problemu.

Zadanie 5: Wprowadzić podane węzły aproksymacji i przeprowadzić aproksymację funkcją logarytmiczną w postaci:

$$y = a_1 + a_2 \cdot \ln(x) \tag{22}$$

Zadanie wykonujemy analogicznie do zadania z funkcją liniową. Należy wprowadzić macierz **X** opisaną we Wskazówkach. Następnie wyznaczyć wektor **A** i obliczyć wartości funkcji logarytmicznej dla zagęszczonego wektora.

```
X=[ones(length(x),1) log(x)']
Y=y'
A=inv(X'*X)*X'*Y

xx=125:625;
f=A(1)+A(2)*log(xx);
```

Zwróć uwagę na to o czym pisałem wcześniej, że logarytm naturalny zapisujemy w Matlabie jako **log( )**.

Zadania 6 i 7. Samodzielne wykonanie zadań nie powinno sprawić problemu.

Zadanie 8: Napisz program, który utworzy pięcioelementowy wektor **D**, w którym zapisane będą wartości bezwzględne błędów dla tych 5 węzłów aproksymacji. Program ma być zautomatyzowany, tak aby można go było szybko wykonać np. dla 1000 punktów. Możesz użyć do tego pętli **for** lub bezpośrednio adresować elementy wektorów innymi wektorami.

Oczywiście fragment programu jest dalszą częścią **Zadania 5**.

Zacznijmy od zastosowania pętli **for**, w celu wyznaczenia wektora **yy**, który będzie zawierał tylko te wartości funkcji **f**, które odpowiadają węzłom. Ponieważ wiemy, że jest to co 125 element z wektora **f** możemy zapisać:

```
N=length(x)
for k=1:N
  yy(k)=f(1+125*(k-1))
end
```

czyli w pierwszym kroku zapiszemy do wektora **yy** na pierwszą pozycję **f(1)**, w drugim **f(126)**, itd.

Teraz wektor **D** możemy utworzyć odejmując po prostu jeden wektor od drugiego (elementy na odpowiadających sobie pozycjach zostaną odjęte):

```
D=yy-y
```

Pamiętaj, że wektory możemy od siebie bezpośrednio odjąć (dodać) tylko wtedy kiedy mają tyle samo elementów.

Teraz drugie podejście. Zamiast pętli **for** możemy bezpośrednio wybrać elementy z wektora **f**:

```
D=f(1:125:end)-y
```

Zapisując `f(1:125:end)` program wstawi w to miejsce nowy wektor, który będzie zawierał tylko te 5 elementów wektora **f** (adresujemy wektor za pomocą innego wektora, patrz pierwsze zajęcia).

Zadanie 9: Zaimplementuj miarę **RMSE** do programu i oblicz błąd dla funkcji logarytmicznej pierwszego stopnia.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\hat{y}_k - y_k\right)^2} \tag{23}$$

Podnosimy do kwadratu wszystkie elementy wektora z błędem bezwzględnym (patrz Zadanie 8):

```
E=D.^2
```

Sumujemy do siebie elementy wektora **E**, można to zrobić w pętli:

```
Suma=0;
for k=1:length(E)
  Suma=Suma+E(k)
end
```

Można też wykorzystać poleceniem **sum()**:

```
Suma=sum(E)
```

Sumę dzielimy przez liczbę elementów:

```
W=Suma/length(E)  %albo N, albo length(D), albo length(y), itd.
```

Na koniec obliczamy pierwiastek z tej wartości:

```
RMSE=sqrt(W)
```

Oczywiście to wszystko można zapisać w bardziej skondensowanej postaci (nawet w jednej linijce). Jeżeli środowisko programistyczne nie daje możliwości bezpośredniego operowania na wektorach (np. nie ma polecenia **sum**), to może zaistnieć konieczność rozpisanie tego bardziej szczegółowo i wtedy niezbędne będzie wykorzystanie np. pętli **for**.

Zadanie 10: Zaimplementuj współczynnik R^2 i oblicz jego wartość dla funkcji logarytmicznej 1 stopnia.

Wynik: $R^2 = 0.894$

Samodzielne wykonanie zadania nie powinno sprawić problemu.

Zadanie 11: Oblicz RMSE oraz R^2 dla pozostałych funkcji z rozdziału 3. Wybierz najlepiej dopasowaną funkcję.

Samodzielne wykonanie zadania nie powinno sprawić problemu.

Wyniki:
Funkcja **logarytmiczna pierwszego** stopnia: RMSE= 7.13, $R^2 = 0.894$
Funkcja **logarytmiczna drugiego** stopnia: RMSE= 1.57, $R^2 = 0.995$
Funkcja **wymierna pierwszego** stopnia: RMSE= 2.59, $R^2 = 0.986$
Funkcja **wymierna drugiego** stopnia: RMSE= 0.72, $R^2 = 0.999$

Wszelkie zastrzeżenia, błędy merytoryczne oraz propozycje przedstawienia wybranych informacji w sposób bardziej klarowny, uprzejmie proszę o kierowanie na adres e-mail kszopa [at] agh.edu.pl.