Podstawy automatyki i robotyki

Laboratorium 1

1.1. Wprowadzenie do systemu Matlab

Matlab jest wysokopoziomowym obiektowym językiem programowania, którego celem jest numeryczne wsparcie badań naukowych i inżynierskich. Język ten implementuje szereg zaawansowanych narzędzi z zakresu metod numerycznych. Z tego względu nazywany jest też numerycznym językiem programowania (numerical programming language). Jego składnia umożliwia łatwą implementację własnych algorytmów obliczeniowych, analizę danych i ich wizualizację. Język Matlab dystrybuowany jest w postaci zbioru narzędzi składających się na kompletne środowisko programistyczne. W jego skład wchodzą między innymi:
- konsola poleceń umożliwiająca interaktywną pracę,
- kompilator umożliwiający skompilowanie programu w celu jego szybszego wykonania,
- zaawansowany edytor skryptów,
- rozbudowany system pomocy
- środowisko Simulink będącym graficznym językiem programowania, które umożliwia modelowanie, symulację i analizę nieliniowych systemów dynamicznych.

1.2. Pojęcia podstawowe

Command Window – okno poleceń.

Path – ścieżka dostępu do katalogu.

Funkcjonalność języka Matlab realizowana jest przy pomocy zbioru plików, które implementują narzędzia numeryczne. Pliki te posegregowane są w katalogach które stanowią **listę ścieżek przeszukiwania** (Matlab search path list). Środowisko Matlab przeszukuje określone katalogi w poszukiwaniu plików z definicjami poszczególnych narzędzi i udostępnia je użytkownikowi w postaci interfejsu programistycznego (API - Application Programming Interface).

Możliwe jest dodanie własnego katalogu (ścieżki dostępu do katalogu) do zbioru katalogów które przeszukiwane są przez system Matlab. Zawarte w takim katalogu pliki, które definiują stworzone przez użytkownika algorytmy, dodawane są do narzędzi języka Matlab. W ten sposób można łatwo rozbudować funkcjonalność środowiska.

Current path – ścieżka dostępu do bieżącego katalogu.

Bieżący katalog jest katalogiem od którego system Matlab rozpoczyna przeszukiwanie zbioru katalogów.

Variable – zmienna, etykieta tekstowa odnosząca się do wydzielonego fragmentu pamięci.

Zmienna jest elementem języka umożliwiającym przechowywanie danych. W języku Matlab w odróżnieniu od języków ogólnego przeznaczenia jak np.: C, C++, Java, utworzenie nowej zmiennej realizowanej jest poprzez przypisanie do niej wartości. Nie istnieje konieczność definiowania typu zmiennej. Język Matlab w zależności od rodzaju danych, automatycznie dobierze typ nowej zmiennej.

Workspace – przestrzeń robocza.

Wydzielony obszar pamięci w którym środowisko Matlab przechowuje zmienne.

Script – (Skrypt) plik tekstowy z rozszerzeniem m.

Skrypt umożliwia grupowanie wyrażeń języka Matlab. Wyrażenia zebrane w skrypcie są wykonywane sekwencyjnie, od początku do końca skryptu.

Function – (Funkcja) plik tekstowy z rozszerzeniem m, który definiuje funkcję języka Matlab.

Funkcja jest konstrukcją programową, której algorytm realizowany jest w wydzielonej przestrzeni adresowej (wg Matlab – przestrzeni roboczej). Funkcja określona jest przez:
- nazwę,

- listę argumentów formalnych,
- ciało funkcji,
- listę parametrów wyjściowych.

Nazwa funkcji musi być taka sama jak nazwa pliku tekstowego który ją zawiera.

Przy pomocy listy argumentów formalnych do przestrzeni adresowej funkcji przekazywane są dane. W ramach ciała funkcji implementowany jest algorytm funkcji, którego wynik zapisywany jest do zmiennych zdefiniowanych w ramach listy parametrów wyjściowych. Dane zapisane w zmiennych, które stanowią listę parametrów wyjściowych przekazywane są do przestrzeni roboczej funkcji, która wywołała funkcję lub do bazowej przestrzeni roboczej w przypadku gdy funkcja wywołana jest ze skryptu lub jako polecenie z konsoli poleceń.

Array – (Tablica) uporządkowany zbiór danych jednego typu.

W języku Matlab każda zmienna jest tablicą. Tablice mogą grupować elementy wybranego typu jak: liczby, znaki, struktury lub obiekty. W przypadku gdy tablica grupuje liczby:
- pojedyncza liczba jest tablicą będącą skalarem o wymiarze: `1 x 1`,
- uszeregowany w ciąg zbiór liczb, jest wektorem wierszowym o wymiarze: `1 x n` lub wektorem kolumnowym o wymiarze: `m x 1`,
- uszeregowany w postaci wierszy i kolumn zbiór liczb, jest macierzą o wymiarze `n x m`,

Możliwe jest również składanie tablic wielowymiarowych. Tablice trójwymiarowe są zbiorem macierzy o ustalonym rozmiarze. Tablice takie mają wymiar: `n x m x k`.

Struct - (Struktura) zbiór danych różnego typu.

Struktury grupują dane różnego typu. W ramach jednej struktury mogą być zgrupowane elementy będące liczbami, znakami, tablicami liczbowymi, tablicami znakowymi, innymi strukturami lub obiektami. Element struktury nazywa się polem.

Operacje macierzowe – są to operacje wykonywane na tablicach (macierzach).

Operacje tablicowe (elementarne) – są to operacje wykonywane na elementach tablic (macierzy).

Wyrażenie – definicja zmiennej, operacje arytmetyczne, logiczne z użyciem zmiennych i funkcji.

1.3. Rozpoczęcie pracy w systemie Matlab.

W pierwszym kroku należy określić jaki jest bieżący katalog. W tym celu należy wydać polecenie: **`pwd`**. W wyniku do zmiennej **`ans`** przypisany zostanie ciąg znaków będący ścieżką dostępu do bieżącego katalogu. Następnie przy pomocy polecenia **`cd`**, przejść do żądanego katalogu. W trakcie pracy skrypty i funkcje zapisywać w żądanym katalogu. Zawartość katalogu można sprawdzić przy pomocy polecenia **`ls`**.

Przykład opisujący zmianę katalogu bieżącego:

```
>> pwd
ans =
       /Users/user/Documents/MATLAB
>> ls
       ROBOCZY
>> cd ROBOCZY/
>> pwd
ans =
       /Users/michal/Documents/MATLAB/ROBOCZY
```

1.4. Tworzenie zmiennych

Zmienne tworzy się poprzez przypisanie do etykiety zmiennej wartości. Operatorem przypisania jest znak =.

`c = 'z'`	Utworzenie znakowej zmiennej c poprzez przypisanie jej wartości (`'z'`)
`x = 5`	Utworzenie liczbowej zmiennej x poprzez przypisanie jej wartości (5)
`s = 'abc'`	Utworzenie wektora znakowego którego elementami są znaki: (`'a'`), (`'b'`) i (`'c'`)
`v = [1, 2, 3]`	Utworzenie wektora wierszowego poprzez przypisanie mu zbioru liczb (`1, 2, 3`)
`v = [1; 2; 3]`	Utworzenie wektora kolumnowego poprzez przypisanie mu zbioru liczb (`1, 2, 3`)
`v = 1:3`	Utworzenie wektora wierszowego poprzez przypisanie mu liczb od `1` do `3` z krokiem `1`
`v = 1:0.5:3`	Utworzenie wektora wierszowego poprzez przypisanie mu liczb od `1` do `3` z krokiem `0.5`
`m = [1, 2; 3, 4]`	Utworzenie macierzy o wymiarze 2 x 2

1.5. Dostęp do elementów tablic / macierzy

Dostęp do elementów tablic lub macierzy realizowany jest przy pomocy nawiasów okrągłych.

Przykład:
Dany jest wektor wierszowy `v = [10, 20, 30, 40, 50]`.

`v(1)`	Pierwszy element wektora v, element o wartości `10`
`v(end)`	Ostatni element wektora v, element o wartości `50`
`v(2 : 3)`	Elementy wektora v od drugiego do trzeciego, wektor elementów: `[20  30]`
`v(:)`	Wszystkie elementy wektora v, wektor elementów `[10 20 30 40 50]`
`v(1 : end-1)`	Elementy wektora v od pierwszego do przedostatniego, wektor elementów `[10 20 30 40]`

Uwaga, sposób dostępu do elementów wektora kolumnowego pozostaje bez zmian.

Przykład:
Dana jest macierz `m = [10, 20, 30; 40, 50, 60; 70, 80, 90]`

```
m   =  10    20    30
       40    50    60
       70    80    90
```

`m(1, 1)`	Element z pierwszego wiersza i pierwszej kolumny, element o wartości 10
`m(2, 3)`	Element z drugiego wiersza i trzeciej kolumny, element o wartości 60
`m(3, 2)`	Element z trzeciego wiersza i drugiej kolumny, element o wartości 80
`m(end, end)`	Element z trzeciego wiersza i trzeciej kolumny, element o wartości 90
`m(1, :)`	Wszystkie elementy z pierwszego wiersza, wektor elementów [10 20 30]
`m(:, 2)`	Wszystkie elementy z drugiej kolumny, wektor elementów [10 20 30]
`m(:, :)`	Wszystkie elementy z macierzy
`m(:)`	Wszystkie elementy z macierzy w postaci wektora kolumnowego
`m(1:2, :)`	Wiersze macierzy m od pierwszego do drugiego, macierz o wymiarze 2 x 3
`m(:, 2:3)`	Kolumny macierzy m od drugiej do trzeciej , macierz o wymiarze 3 x 2
`m(1:2, 2:3)`	Macierz o wymiarze 2 x 2, o elementach: 20 30 50 60

1.6. Operacje arytmetyczne

Język Matlab udostępnia następujące operacje arytmetyczne:
- dodawanie macierzy (`+`),
- odejmowanie macierzy (`-`),
- mnożenie macierzy (`*`),
- dzielenie macierzy (`/`), operacja dzielenia jest odpowiednikiem mnożenia przez macierz odwrotną,
- dzielenie lewostronne macierzy (`\`), operacja ta zdefiniowana jest jako rozwiązanie `X = A\B` układu równań liniowych: `AX = B`,
- potęgowanie macierzy (`^`),
- transponowanie macierzy (`'`),
- mnożenie (elementów) tablic (`.*`),
- dzielenie (elementów) tablic (`./`),
- lewostronne dzielenie (elementów) tablic (`.\`),

- potęgowanie (elementów) tablic (.^),

Przykłady (zaczerpnięte z pomocy systemowej środowiska Matlab):
Dane są dwa wektory kolumnowe: x oraz y:

x	y
1	4
2	5
3	6

x'	= 1 2 3	y'	= 4 5 6	x + y	= 5 7 9	x - y	= -3 -3 -3
x + 2	= 3 4 5	x – 2	= -1 0 1	x * y	= błąd	x .* y	= 4 10 18
x' * y	= 32	x' .* y	= błąd	x * y'	= 4 5 6 8 10 12 12 15 18	x .* y'	= błąd
x * 2	= 2 4 6	x .* 2	= 2 4 6	x \ y	= 16/7	x .\ y	= 4 5/2 2
2 \ x	= 1/2 1 3/2	2 ./ x	= 2 1 2/3	x / y	= 0 0 1/6 0 0 1/3 0 0 1/2	x ./ y	= 1/4 2/5 1/2
x / 2	= 1/2 1 3/2	x ./ 2	= 1/2 1 3/2	x ^ y	= błąd	x .^ y	= 1 32 729
x ^ 2	= błąd	x .^ 2	= 1 4 9	2 ^ x	= błąd	2 .^ x	= 2 4 8

1.7. Operatory relacji

Prawda w języku Matlab określona jest wartością 1, fałsz wartością 0.

A < B	czy wartość operandu A jest mniejsza od wartości operandu B
A > B	czy wartość operandu A jest większa od wartości operandu B
A <= B	czy wartość operandu A jest mniejsza lub równa od wartości operandu B
A >= B	czy wartość operandu A jest większa lub równa od wartości operandu B
A == B	czy wartość operandów A i B są sobie równe
A ~= B	czy wartości operandów A i B są różne

1.8. Instrukcje złożone

Instrukcja warunkowa IF-ELSE

Konstrukcja	Przykład

if wyrażenie_logiczne
  instrukcje
elseif wyrażenie_logiczne
  instrukcje
else
  instrukcje
end
	if i > 0
	  a = 1;
	else
	  a = 2;
	end

Instrukcja iteracyjna FOR

Konstrukcja	Przykład

for zmienna = wyrażenie
  instrukcje
end
	for i = 1:3
	  for j = 1:5
	  m(i,j) = 1 / (i+j);
	end

Instrukcja iteracyjna WHILE

Konstrukcja	Przykład

for wyrażenie
  instrukcje
end
	for i < 5
	  v(i) = i;
	  i = i + 1;
	end

1.9. Grupowanie wyrażeń (poleceń)

Praca w konsoli z interpreterem poleceń jakim jest system Matlab może być nieefektywna w przypadku realizacji złożonego algorytmu. W celu wyeliminowania tego problemu, system Matlab wyposażono w możliwość uruchamiania skryptów. Skrypty te grupują wyrażenia języka Matlab.

W celu utworzenia nowego skryptu, należy w konsoli poleceń wydać polecenie: **edit** lub z menu **File** wybrać polecenie **New** a następnie **M-File**. Otwarte zostanie okno edytora skryptów. Skrypt jest plikiem tekstowym z rozszerzeniem m. W celu zapisania skryptu, należy z menu **File** wybrać polecenie **Save As...**. Nazwa skryptu nie powinna zawierać znaków: spacji, tabulacji oraz znaków diakrytycznych. Nazwa skryptu nie powinna też być liczbą. Skrypt można uruchomić, zatwierdzając w konsoli poleceń wyrażenie będące nazwą skryptu.

Przykład.

Zawartość skryptu o nazwie: scr01.m

```
clc;         % usunięcie zawartości okna poleceń
clear;       % usunięcie zmiennych z bazowej przestrzeni roboczej
a = 1;       % utworzenie zmiennej a i przypisanie jej wartości 1
b = 2;       % utworzenie zmiennej a i przypisanie jej wartości 1
c = a + b    % utw. zm. c i przypisanie jej wart. będącej wynikiem wyk. wyrażenia a + b
```

Skrypt uruchomiono wykonując polecenie o nazwie: scr01.

1.10. Funkcje

Utworzenie funkcji w języku Matlab tworzy się w taki sam sposób jak skrypty. Plik tekstowy zawierający definicję funkcji musi mieć taką samą nazwę jak funkcja. Pozostałe ograniczenia dotyczące nazewnictwa funkcji są takie same jak skryptów.

Definicja funkcji:

```
function [lista_parametrów_wyjściowych] = nazwa_funkcji(lista_parametrów_formalnych)
      % opis funkcji
      wyrażenia;
end
```

Użycie funkcji:

```
zmienna = nazwa_funkcji(lista_parametrów_aktualnych);
```

Przykład

Funkcja o nazwie fun01 zdefiniowana w pliku: fun01.m.

```
function [z] = fun01(x, y)
      % funkcja zwraca wynik będący sumą wartości argumentów formalnych
      z = x + y;
end
```

Wywołanie funkcji fun01.

```
a = 1;
b = 2;
c = fun01(a, b)
```

Uzyskanie informacji o funkcji fun01.

```
help fun01
```

1.11. System pomocy środowiska Matlab.

Środowisko Matlab zbudowane jest w oparciu o dużą liczbę funkcji, które realizują określone algorytmy. Funkcje te definiują zasoby środowiska. Informacje o dostępnych zasobach systemu Matlab można uzyskać wydając polecenie: **help** lub **doc**.

Przykład użycia polecenia **help**:

```
>> help
  HELP topics:

matlab/general                - General purpose commands.
matlab/ops                    - Operators and special characters.
matlab/lang                   - Programming language constructs.
matlab/elmat                  - Elementary matrices and matrix manipulation.
matlab/elfun                  - Elementary math functions.
matlab/matfun                 - Matrix functions - numerical linear algebra.
matlab/polyfun                - Interpolation and polynomials.
matlab/funfun                 - Function functions and ODE solvers.
...

>> help elfun
  Elementary math functions.

  Trigonometric.
    sin         - Sine.
    sind        - Sine of argument in degrees.
    sinh        - Hyperbolic sine.
    ...

>> help sin
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements of X.

    See also asin, sind.

    Overloaded methods:
       codistributed/sin

    Reference page in Help browser
       doc sin
```

1.12. Elementy grafiki w systemie Matlab

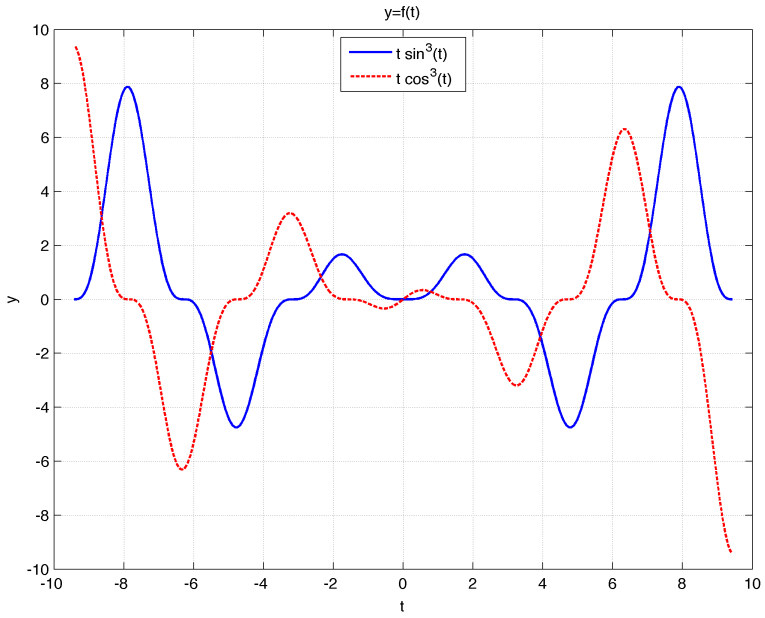
Język Matlab wyposażony jest w zbiór funkcji wysokopoziomowych umożliwiających otwarcie nowego okna, umieszczenie na nim pola graficznego, wykreślenie na tym polu grafiki 2D lub 3D oraz modyfikację wykresu. Wykorzystanie tych funkcji przedstawione zostanie na przykładach.

Przykład, grafika 2D:

Utworzyć wykres dwóch funkcji na jednym polu: $y_1(t) = t\sin^3(t), y_2(t) = t\cos^3(t)$w dziedzinie $t \in \langle -3\pi, 3\pi \rangle$z krokiem $\pi/180$.

Zawartość skryptu `graf01.m`

```
clc; clear;                         % wyczyszczenie konsoli i bazowej przestrzeni roboczej
t = -3*pi:pi/180:3*pi;              % utworzenie wektora dziedziny
y1 = t.*sin(t).^3;                  % definicja pierwszej funkcji, utworzenie wektora y1
y2 = t.*cos(t).^3;                  % definicja drugiej funkcji, utworzenie wektora y2
figure(1);                          % otwarcie nowego okna o numerze 1
plot(t,y1,'b', t,y2,'r--');         % umieszczenie na nowym oknie wykresu funkcji: y1, y2
grid on;                            % umieszczenie siatki na wykresie
xlabel('t');                % opisanie osi x wykresu
ylabel('y');                % opisanie osi y wykresu
title('y=f(t)');                    % nadanie tytułu
legend('t sin^3(t)','t cos^3(t)','Location','North')% utworzenie legendy
```



Dodatkowe informacje o użytych funkcjach można uzyskać wydając polecenie **`help`** `nazwa_funkcji`.

Język Matlab udostępnia funkcję umożliwiającą zapis zawartości okna graficznego do pliku. Możliwe jest między innymi wybranie typu pliku graficznego oraz rozdzielczości z jaką zostanie zapisany plik. Funkcja ta nosi nazwę: **`print`**.

Przykład:

Uruchomienie skryptu `graf01.m` spowoduje otwarcie okna graficznego o numerze 1 i umieszczenie na nim wykresów. W celu zapisania zawartości do pliku typu png z rozdzielczością 300dpi bez elementów sterujących okna, należy wydać polecenie:

```
print(1, '-dpng', '-noui', '-r300', 'nazwa_pliku.png')
```

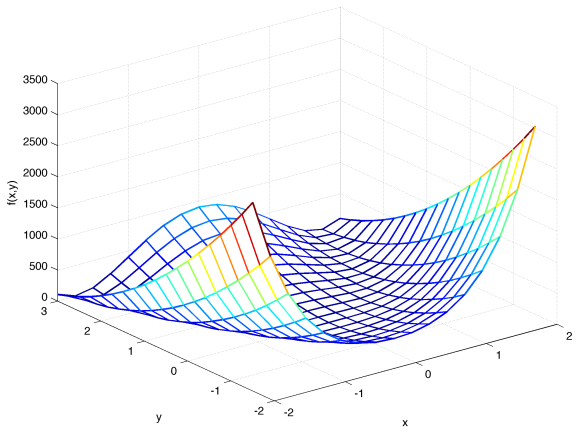
Pierwszy argument określa numer okna. Drugi argument definiuje rodzaj pliku. Trzeci argument nakazuje pominąć elementy sterujące okna graficznego. Czwarty argument określa rozdzielczość z jaką funkcja zachowa wykres. Piąty argument oznacza nazwę pliku do którego zapisana zostanie zawartość okna.

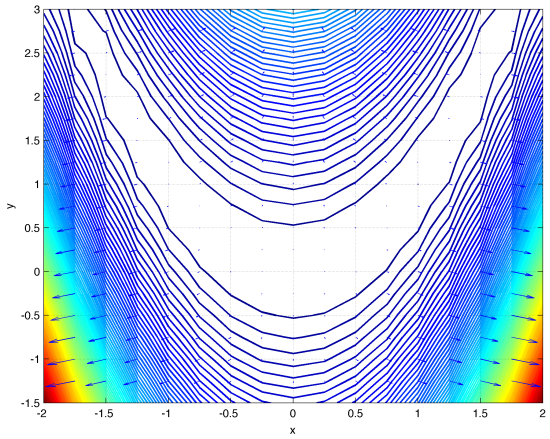
Przykład, grafika 3D:

Zawartość skryptu `graf02.m`

```
clc; clear all;

f = @(x,y) (1-x).^2 +100.*(y-x.^2).^2; % funkcja Rosenbrock'a
x = -2:0.25:2;                         % zakres zmian zmiennej x (dziedzina)
y = -1.5:0.25:3;                       % zakres zmian zmiennej y (dziedzina)
v = 0:30:3500;                         % poziomy linii na konturze

% wykres 3D
[X, Y] = meshgrid(x,y);    % utworzenie siatki do wykresu 3D
F = f(X,Y);                % wyznaczenie wartości funkcji Rosenbrock'a w dziedzinie
figure(1);                 % utworzenie okna o numerze 1
mesh(X,Y,F);               % naniesienie wykresy trójwymiarowego na okno numer 1
xlabel('x');               % opisanie osi x
ylabel('y');               % opisanie osi y
zlabel('f(x,y)');          % opisanie osi z

% wykres konturowy
figure(2);                      % utworzenie okna numer 2
[C, h] = contour(X,Y,F,v);      % naniesienie wykresu konturowego na okno numer 2
xlabel('x');                    % opisanie osi x
ylabel('y');                    % opisanie osi y
grid('on');                     % opisanie osi z

% gradient
hx = 0.25;                              % krok ilorazów różnicowych względem osi x
hy = 0.25;                              % krok ilorazów różnicowych względem osi y
[dx,dy] = gradient(F,hx,hy);                  % wyznaczenie gradientów
hold('on'); quiver(X,Y,dx,dy); hold('off');   % naniesienie gradientów na wykres
konturowy
```

W wyniku uruchomienia skryptu `graf02.m`, utworzone zostaną poniższe wykresy:





Uwaga:
Funkcja Rosenbrock'a zdefiniowana została w postaci funkcji anonimowej (bez nazwy). Ogólna definicja takiej funkcji jest następująca:

```
nazwa_funkcji = @(argumenty) wyrażenie_w_funkcji_argumentów
```

np.:
```
funKwadrat = @(t, a, b, c) a.*t.^2 + b.*t + c;
```

wywołanie:
```
x = -1 : 0.1: 1;
y = funKwadrat(x, 1, 2, 3);
```

2.1. Elementy metod numerycznych – Aproksymacja

Aproksymacja jest zadaniem polegającym na przybliżeniu funkcji aproksymowanej przy pomocy innej funkcji zwanej funkcją aproksymującą. Funkcja aproksymowana najczęściej określona jest w postaci zbioru punktów, na przykład danych pomiarowych, jak wartości napięcia w czasie. Funkcja aproksymująca określana jest przy pomocy przyjętego wyrażenia algebraicznego. Nie jest konieczne by w całym zakresie analizowanych danych, funkcja aproksymująca przechodziła przez wszystkie punkty określone funkcją aproksymowaną. Krzywa otrzymana w wyniku aproksymacji powinna przebiegać tak aby jej odchylenia od punktów funkcji aproksymowanej były minimalne.

Przykład
Aproksymacja funkcji sinus która w zadanym przedziale czasu przyjmuje wartości Y w chwilach `t`. Funkcją aproksymującą jest wielomian: $y_a = a_1 t^3 + a_2 t$. Zadanie aproksymacji polega na wyznaczeniu wartości współczynników a_1 oraz a_2. Należy zauważyć, że problem sprowadza się do rozwiązania liniowego układu równań `Y = A*X`. Gdzie Y jest znanym wektorem wartości funkcji aproksymowanej, X jest znaną macierzą, której kolumny definiowane są jako funkcje dziedziny funkcji aproksymowanej, natomiast A jest poszukiwanym wektorem współczynników funkcji aproksymującej.

Zawartość skryptu `aproks01.m`.

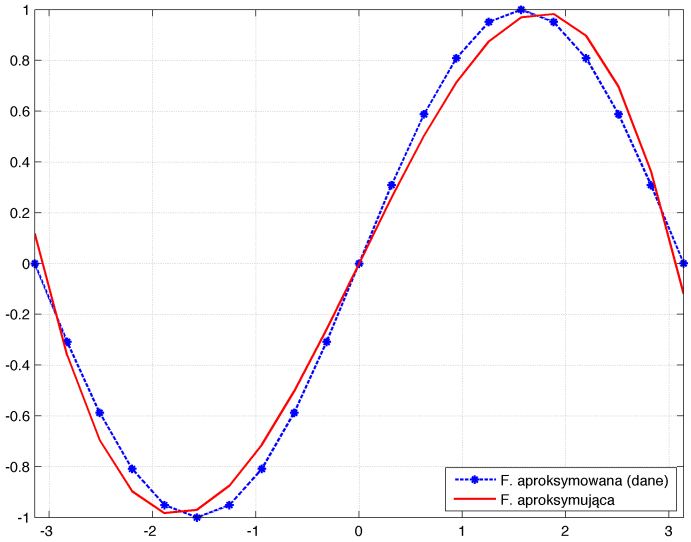
```
clc; clear;
t = -pi : pi/10 : pi;
Y = sin(t);             % funkcja aproksymowana Y = f(t)

X = [t.^3; t];          % y = a1*t^3 + a2*t
A = Y / X;              % współczynniki funkcji aproksymującej

Ya = A*X;               % wartości funkcji aproksymującej

figure(1); plot(t,Y,'b*--', t,Ya,'r', 'Linewidth', 1.5);
grid('on'); xlim([-pi pi])
```

Uwaga:
W skrypcie `aproks01.m` dane aproksymowane są funkcją `y = a1*t^3 + a2*t`. W przypadku zastosowania innej funkcji aproksymującej należy zmodyfikować macierz X.

W przypadku aproksymowania innymi funkcjami należy zmodyfikować macierz X. Przykłady przedstawiono w poniższej tabeli:

Funkcja aproksymująca	Macierz X
`y = a1*t + a0`	`X = [t; t.^0];`
`y = a1*t.^2 + a2*t + a0`	`X = [t.^2; t; t.^0];`
`y = a1*t.^3 + a1*t.^2 + a2*t + a0`	`X = [t.^3; t.^2; t; t.^0];`
`y = a1*t.^5 + a1*t.^3 + a2*t`	`X = [t.^5; t.*3; t];`

W wyniku uruchomienia skryptu `aproks01.m`. uzyskano a_1=-0,09, a_2=0,83. Wykres przedstawiający wynik aproksymacji jest następujący:



3. Zadania do wykonania

1. Oblicz wartości wyrażeń:

$$\ln\left(\sqrt[3]{a}+\sqrt[4]{b}-1\right)\text{gdzie } a=1{,}3 b=0{,}9$$

$$\cos\left(\frac{\pi}{3}\right)+\sin(21^o)+\arctan^2(0{,}3)$$

$$\sqrt{\frac{\sin(0{,}6)}{\mathrm{e}^{\arccos(0{,}7)}}}$$

2. Utworzyć skrypt realizujący następujące polecenia:
   1. Utworzyć poziomy wektor v0 o sześciu elementach. Wartość początkowa powinna być równa 0 natomiast wartość końcowa 2.5.
   2. Utworzyć nowe wektory poziome v1, v2, v3, v4 poprzez wykonanie operacji: dodanie do wektora v wartości 1, odjęcie od wektora v wartości 2, pomnożenie wektora v przez wartość 3, podzielenie wektora v przez wartość 4.
   3. Utworzyć macierz A (5x6) poprzez konkatenację utworzonych wektorów.
   4. Utworzyć wektor kolumnowy Y z czwartej kolumny macierzy A i usunąć ten wektor z macierzy.
   5. Utworzyć macierz R (5x5) której elementy są wybrane losowo o rozkładzie jednostajnym.
   6. Przemnożyć poszczególne elementy macierzy A i R przez siebie, tworząc nową macierz A.
   7. Utworzyć macierze od D1 do D5 poprzez wstawienie w k-tą (1, 2, 3, 4 , 5) kolumnę macierzy A wektor Y.
   8. Utworzyć wektor d składający się z wyznaczników utworzonych macierzy Dk.
   9. Utworzyć wektor w poprzez podzielenie wartości wektora d przez wyznacznik macierzy A.
   10. Utworzyć wektor r realizujący polecenie: A\Y
   11. Porównać wartości wektorów w i r.

3. Wygenerować wykresy funkcji:

$$y_1(t)=\sin(t)\mathrm{e}^{-0{,}08t}$$
$$y_2(t)=\cos(t)\mathrm{e}^{-0{,}15t} \text{ dla } t\in\langle 0{,}12\rangle \text{z krokiem równym } 0{,}1$$

Wykresy powinny być zamieszczone na jednym oknie. Wykres pierwszej funkcji powinien być rysowany linią ciągłą czerwoną a wykres drugiej funkcji, linią przerywaną niebieską. Opisać osie wykresów, nadać tytuł wykresowi i dodać legendę do wykresu umieszczoną wewnątrz pola wykresu centralnie u góry.

Zapisać wykres funkcji do pliku typu png. Przyjąć rozdzielczość 150dpi oraz ustaloną grubość linii równą 1,5.

4. Wygenerować wykres funkcji Ackley'a w 3D:

$$y = -20\mathrm{e}^{-2\sqrt{0{,}5(x^2+y^2)}} - \mathrm{e}^{0{,}5\cos(2\pi x)+\cos(2\pi y)} + 20 + \mathrm{e}^1 \text{gdzie } \begin{matrix} x \in \langle -4{,}4 \rangle \\ y \in \langle -4{,}4 \rangle \end{matrix} \text{z krokiem równym } 0{,}25$$

5. Utworzyć skrypt (lub funkcję) wyznaczający wartość liczby pi według zależności:

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{2\mathrm{i}+1} = \frac{\pi}{4}$$

6. Utworzyć skrypt, który realizuje algorytm aproksymacji funkcji cos(t) w przedziale od -pi do pi z krokiem pi/10. Narysować wykresy funkcji aproksymowanej i aproksymującej.

7. Utworzyć skrypt, który realizuje algorytm aproksymacji danych tabelarycznych. Narysować wykresy danych tabelarycznych i funkcji aproksymującej.

```
X = [  0       30      60      90      120     150     180    ]
Y = [  140.00  131.67  123.24  134.70  146.06  157.32  168.47 ]
```

4. Pytania

1. Czym różnią się operacje tablicowe od macierzowych, podać przykłady.
2. Czym różni się skrypt o funkcji w języku Matlab, podać przykłady.
3. Co to jest przestrzeń robocza środowiska Matlab.
4. Co to jest zmienna i jak jest definiowana w systemie Matlab.
5. Co to jest funkcja bez nazwy i jak jest definiowana.
6. Wymienić i podać składnię instrukcji iteracyjnych, podać przykłady.
7. Co to jest aproksymacja, podaj przykład implementacji aproksymacji metodą najmniejszych kwadratów w języku Matlab