Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Elektrotechniki Przemysłowej i Automatyki
Zakład Urządzeń i Systemów Automatyki

Komputerowa symulacja układów dynamicznych

Wprowadzenie do systemu MATLAB

Instrukcja laboratoryjna
(wersja robocza)

2017

1. System 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.

1.1. 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 - Applicaton 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 defniuje 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.

1.2. 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`.

Listing 1.1. Zmiana bieżącego katalogu

```
>> pwd
ans =
C:\Users\Paweł\Documents\Lab1
>> ls
.     ..    Zad1
>> cd Zad1
>> pwd
ans =
C:\Users\Paweł\Documents\Lab1\Zad1
```

1.3. Tworzenie zmiennych

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

Listing 1.2. Tworzenie zmiennych

```
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
```

1.4. Dostęp do elementów tablic / macierzy

Dostęp do elementów tablic lub macierzy realizowany jest przy pomocy nawiasów okrągłych Przykładowy dostęp do poszczególnych elementów macierzy zawarto na listingu 1.3

Listing 1.3. Dostęp do elementów wektora - przykład

```
>>v = [10, 20, 30, 40, 50]; % utworzenie wektora wierszowego
>>v(1) % pierwszy element wektora
ans =
10
>>v(end) % ostatni element wektora
ans =
50
>>v(2 : 3) % elementy wektora od pierwszego do trzeciego
ans =
20 30
>>v(1 : end-1) % elementy wektora od pierwszego do przedostatniego
10    20    30    40
```

Uwaga, sposób dostępu do elementów wektora kolumnowego pozostaje bez zmian. Sposób dostępu do elementów macierzy przedstawiono poniżej.

Listing 1.4. Dostęp do elementów macierzy - przykład

```
>>m = [10, 20, 30; 40, 50, 60; 70, 80, 90] % utworzenie macierzy
m =
    10    20    30
    40    50    60
    70    80    90
>>m(1, 1) % element z pierwszego wiersza i pierwszej kolumny
ans =
10
>>m(3, 2) % element z trzeciego wiersza i drugiej kolumny
ans =
80
>>m(end, end) % element z trzeciego wiersza i trzeciej kolumny
ans =
90
>> m(:, 2) % wszystkie elementy z drugiej kolumny

ans =
    20
    50
    80
```

```
>>m(:, :) %Wszystkie elementy z macierzy
ans =
    10    20    30
    40    50    60
    70    80    90
>>m(1:2, 2:3) %Macierz o wymiarze 2 x 2
ans =
    20    30
    50    60
```

1.5. 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 `.^`,

Poniżej przedstawiono przykład użycia operatorów arytmetycznych.

Listing 1.5. Przykład użycia operatorów arytmetycznych

```
>>x = [1; 2; 3]
x =
    1
    2
    3
>>y = [4; 5; 6]
y =
    4
    5
    6
>> x'
```

```
ans =
     1     2     3
>>x+y
ans =
     5
     7
     9
>>2 ./ x
ans =
    2.0000
    1.0000
    0.6667
>>x.^2
ans =
     1
     4
     9
>>x^2
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
```

Szczególną uwagę należy zwrócić na rozróżnienie operacji mnożenia i dzielenia macierzy z operacjami mnożenia i dzielenia ich poszczególnych elementów.

1.6. Operatory relacji

Prawda w języku Matlab określona jest wartością 1, fałsz wartością 0. W tabeli 1.1 przedstawiono występujące w języku Matlab operatory relacji.

Tabela 1.1. Operatory relacji w języku Matlab

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

1.7. Instrukcje złożone

Konstrukcje instrukcji złożonych oraz przykłady ich użycia przedstawiono poniżej.

Konstrukcja instrukcji warunkowej `if-else`:

```
if wyrazenie_logiczne
      instrukcje
elseif wyrazenie_logiczne
      instrukcje
else
      instrukcje
end
```

Przykład użycia instrukcji warunkowej `if-else` przedstawiono na listingu poniżej.

Listing 1.6. Przykład użycia instrukcji warunkowej if-else

```
if i > 0
      a = 1 ;
else
      a = 2 ;
end
```

Konstrukcja instrukcji iteracyjnej `for` wygląda następująco:

```
for zmienna = wyrazenie
      instrukcje
end
```

Poniżej przedstawiono przykład jej użycia.

Listing 1.7. Przykład użycia instrukcji iteracyjnej for

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

Instrukcja iteracyjna `while` przyjmuje następującą konstrukcje:

```
while wyrazenie
      instrukcje
end
```

Jej użycie zaprezentowano na poniższym listingu.

Listing 1.8. Przykład użycia instrukcji iteracyjnej while

```
while i < 5
      v(i) = i;
      i = i + 1;
end
```

1.8. Grupowanie wyrażeń (poleceń) - skrypty

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. Zawartość przykładowego skryptu zawarto na listingu 1.9.

Listing 1.9. Zawartość skryptu o nazwie: scr01

```
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`.

1.9. 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. Konstrukcja definicji funkcji wygląda następująco:

```
function [parametry_wyjsciowe] = nazwa_funkcji(parametry_formalne)
      %opis funkcji
      instrukcje;
end
```

Przykładową zawartość skryptu zawierającego definicję funckji przedstawiono na listingu 1.10. Na listingu 1.11 natomiast zaprezentowano wywołanie funkcji `fun01`.

Listing 1.10. Zawartość skryptu o nazwie: fun01

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

Listing 1.11. Wywołanie funkcji fun01

```
>>a=1;
>>b=2;
>>c=fun01(a,b);
>>c
c =
     3
```

1.10. Funkcje

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`.

Listing 1.12. Wyświetlenie opisu funkcji fun01

```
>>help fun01
  funkcja zwraca wynik będący sumą wartości argumentów formalnych
```

1.11. Elementy grafiki w systemie Matlab

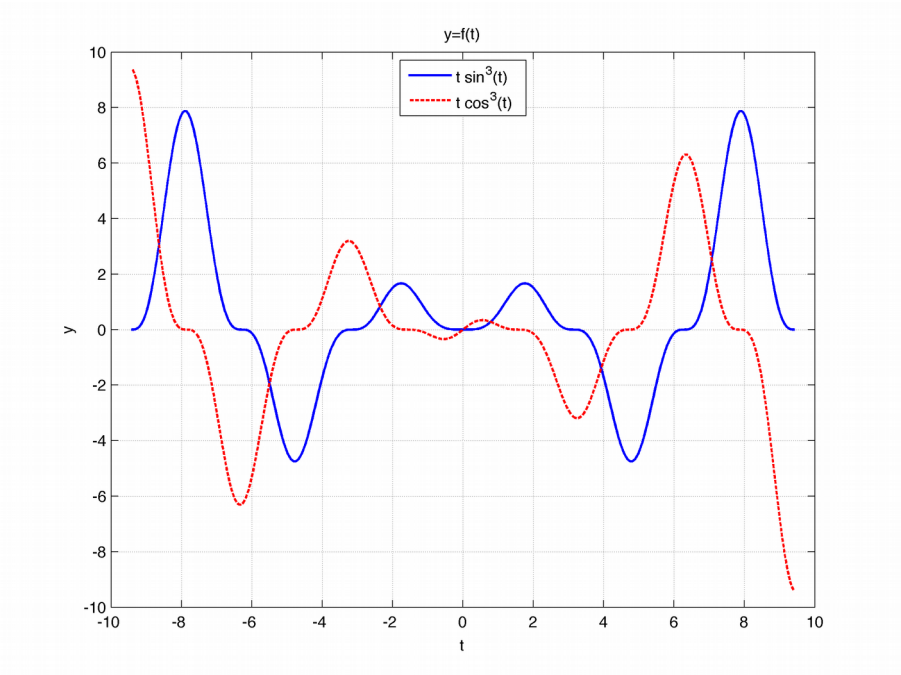
Język Matlab wyposażony jest w zbiór funkcji wysokopoziomowych umożliwiających otwarcie nowego okna, umieszczenie na nim pola graficznego, wykreślenie na tym polu grafiki 2D lub 3D oraz modyfikację wykresu. Wykorzystanie tych funkcji przedstawiono poniżej na przykładach.

Przykład 1

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 $\pi/180$

Listing 1.13. 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'); ylabel('y'); title('y=f(t)'); % opisanie osi i nadanie tytułu
legend('t sin^3(t)','t cos^3(t)','Location','North') % utworzenie legendy
```



Rysunek 1.1. Wynik działania skryptu graf01.m

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`. 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 przedstawione na listingu poniżej.

Listing 1.14. Zawartość skryptu graf01.m

```
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.

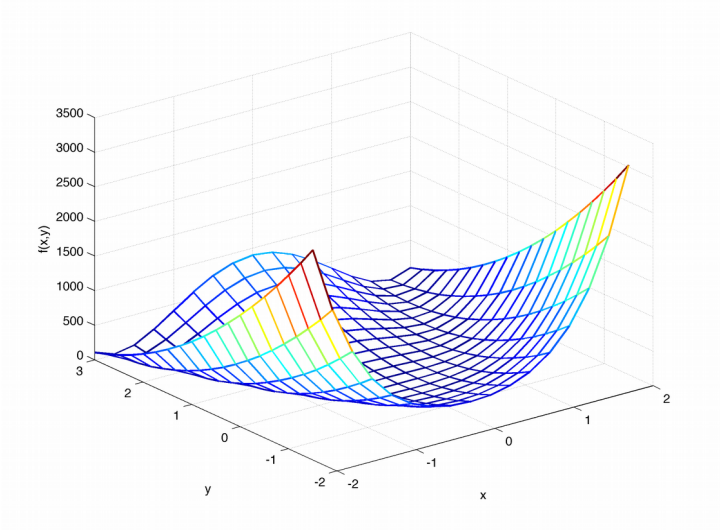
Przykład 2

Narysować skrypt generujący wykres funkcji Rosenbrock'a.

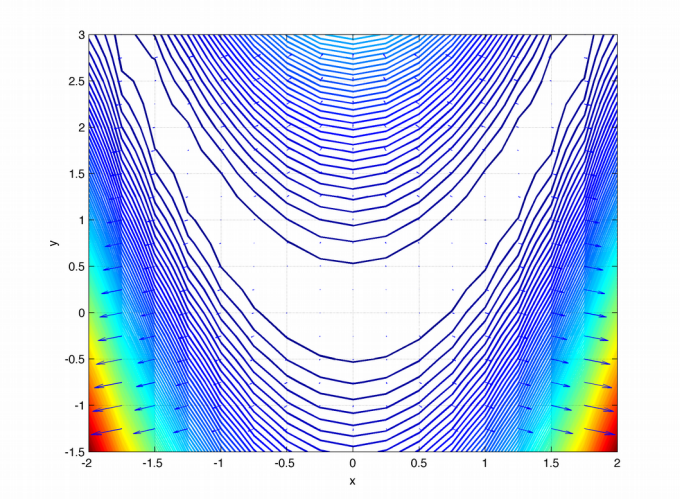
Listing 1.15. Zawartość skryptu graf02.m

```
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
```



Rysunek 1.2. Wynik działania skryptu graf02.m -okno numer 1



Rysunek 1.3. Wynik działania skryptu graf02.m -okno numer 2

Zadania do wykonania

Zadanie 1

Obliczyć wartość wyrażenia:

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

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

$$\sqrt{\frac{sin(0,6)}{e^{arccos(0,7)}}} \tag{3}$$

Zadanie 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
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: $\mathbf{A}\backslash\mathbf{Y}$
11. Porównać wartości wektorów **w** i **r**.

Zadanie 3

Wygenerować wykres funkcji:

$$y_1(t) = sin(t)e^{-0,08t} \tag{4}$$

$$y_2(t) = cos(t)e^{-0,15t} \tag{5}$$

gdzie: $t \in \langle 0, 12\rangle$ 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 utworzony wykres do pliku

Zadanie 4

Wygenerować wykres funkcji Ackley'a w 3D:

$$y = -20e^{-2\sqrt{0,5(x^2+y^2)}} - e^{-0,5cos(2\pi x)+cos(2\pi y)} + 20 + e^1 \tag{6}$$

gdzie: $x \in \langle -4, 4\rangle$, $y \in \langle -4, 4\rangle$ z krokiem równym 0,25.

Zadanie 5

Utworzyć skrypt (lub funkcję) wyznaczający wartość liczby π według zależności.

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = \frac{\pi}{4} \tag{7}$$