Wydział Elektryczny
Zespół Automatyki (ZTMAiPC)
ZERiA

LABORATORIUM MODELOWANIA I SYMULACJI

Ćwiczenie 1

Wprowadzenie do środowiska Matlab-Simulink.

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się ze środowiskiem obliczeniowym Matlab-Simulink oraz zdobycie praktycznych umiejętności przeprowadzania symulacji modeli dynamicznych w środowisku Matlab-Simulink.

2. Wprowadzenie.

Pakiet Matlab (**Mat**rix **Lab**oratory) jest interakcyjnym i otwartym środowiskiem obliczeniowym integrującym analizę numeryczną, działania na macierzach, i przetwarzanie sygnałów z grafiką, co bardzo ułatwia jego wykorzystanie. Podstawową strukturę danych stanowi macierz (rzeczywista lub zespolona), której rozmiarów nie trzeba uprzednio deklarować. W istocie Matlab jest językiem programowania bardzo wysokiego poziomu.

Zalety i możliwości Matlaba poszerzają tzw. przyborniki (toolboxes) będące wyspecjalizowanymi funkcjami (m-plikami) przeznaczonymi do rozwiązywania określonych zagadnień z dziedziny teorii regulacji i sterowania, chemii, matematyki, przetwarzania obrazów i inne.

Niewątpliwym uzupełnieniem Matlaba, ważnym szczególnie z punktu widzenia automatyki, jest przybornik (biblioteka) Simulink, który służy do analizy i syntezy ciągłych, dyskretnych i dyskretno-ciągłych układów dynamicznych. Simulink jest środowiskiem, w którym symulację systemów dynamicznych wykonuje się w oparciu o schemat blokowy budowany z wykorzystaniem predefiniowanych bloków bibliotecznych.

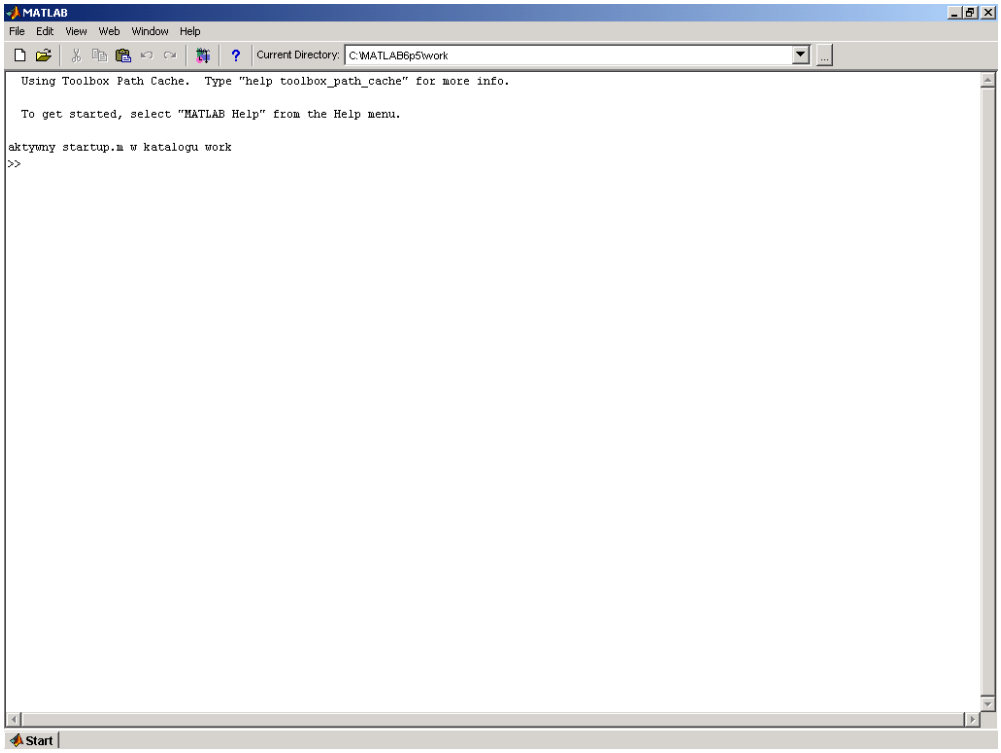
2.1 Matlab - Środowisko pracy.

Program Matlab jest zintegrowanym środowiskiem obliczeniowym ukierunkowanym szczególnie na obliczenia numeryczne. Nie wyklucza to jednak obliczeń symbolicznych – możliwych z zastosowaniem odpowiedniego przybornika (*Symbolic Toolbox*). Funkcjonalność Matlab'a jest związana przede wszystkim z ogromną biblioteką elementarnych i specjalnych funkcji matematycznych, operatorów macierzowych oraz dużą liczbą przyborników (bibliotek funkcji) wykorzystywanych do zastosowań specjalistycznych. Po uruchomieniu pakietu (poleceniem matlab.exe) ukazuje się główne okno programu, będące jednocześnie obliczeniowym centrum sterowania. Edycja poleceń przypomina system DOS. Polecenia wpisywane są w linii komend i zatwierdzane klawiszem <ENTER>. To jeden z dwóch dostępnych trybów pracy w środowisku Matlab – tryb pracy bezpośredni. Alternatywnie, możliwe jest zamknięcie grupy poleceń wewnątrz struktur zwanych dalej skryptami i funkcjami (tryb pracy pośredni).

Skrypty – stanowią sekwencję poszczególnych poleceń zapisaną w pliku tekstowym. Zmienne modyfikowane wewnątrz skryptu są dostępne po jego wykonaniu w przestrzeni roboczej (*workspace*). Skrypty jako takie nie zwracają wartości, a jedyną możliwością parametryzacji ich działania jest wcześniejsze zainicjowanie odpowiednich zmiennych (parametrów).

Funkcje – w odróżnieniu od skryptów, definiowane są jako zamknięte fragmenty algorytmu obliczeniowego. Posiadają listę parametrów wejściowych, zwracają wartość (lub wartości przez zastosowanie macierzy). Zmienne wewnętrzne są tworzone na czas wykonania się funkcji i nie są dostępne z poziomu przestrzenie roboczej.

Okno główne programu Matlab przedstawia rys.1.

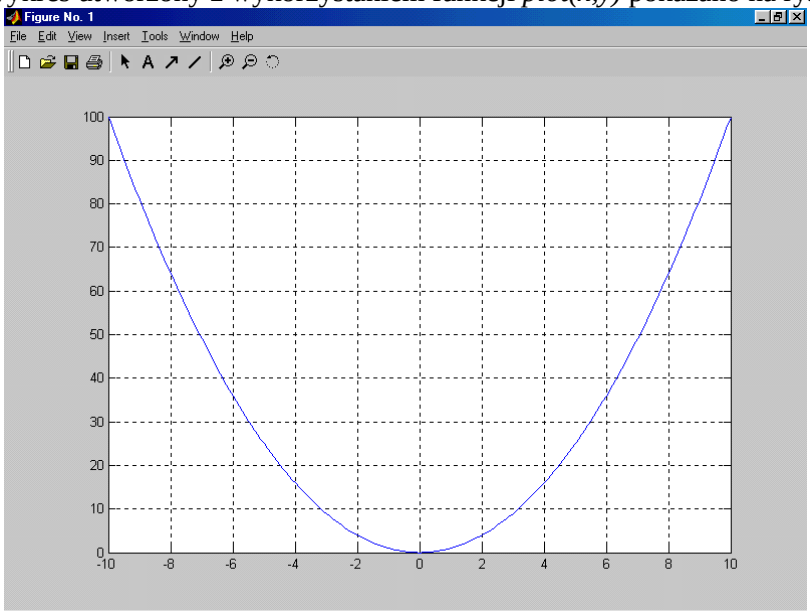


Rys. 1. Okno Główne programu Matlab.

W programie Matlab na uwagę zasługuje system pomocy. Pomoc ta dostępna jest jako zestaw funkcji wewnętrznych (help, lookfor, helpdesk i in.) jak i dokumentacja w formie PDF lub HTML umieszczona w katalogu C:\MATLAB6p5p1\help.

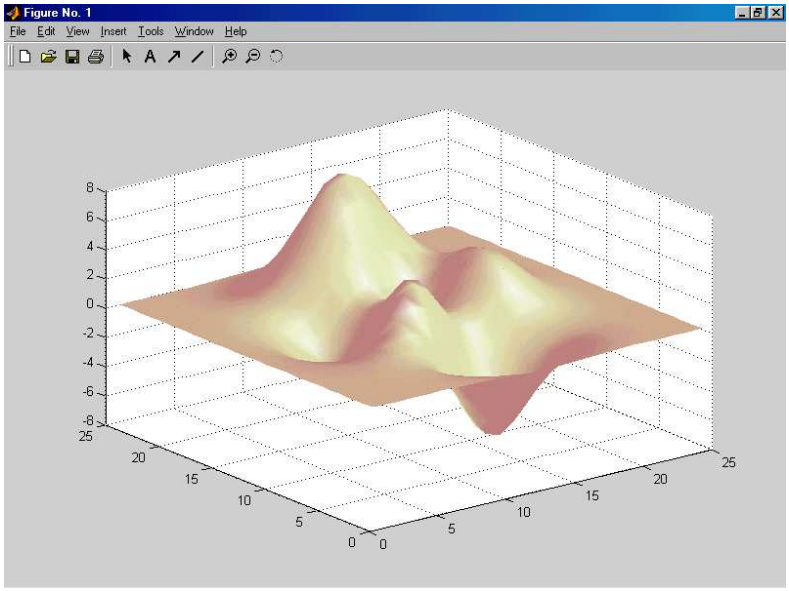
Oprócz obliczeń numerycznych, program Matlab umożliwia wizualizację wyników obliczeń. Do dyspozycji jest wiele funkcji graficznych wspomagających prezentacje danych i analiz. Podstawowe z nich, to: *plot(x,y)*, *subplot(m,n,p)*, *fplot(fun, lims)*, *grid* i inne.

Przykładowy wykres utworzony z wykorzystaniem funkcji *plot(x,y)* pokazano na rys. 2



Rys 2. Przykładowy wykres uzyskany z wykorzystaniem funkcji *plot(x,y)*.

Dodatkowym rozszerzeniem możliwości graficznych jest zbiór funkcji umożliwiających tworzenie wykresów dwu- i trójwymiarowych. Przykładowy wykres utworzony funkcją *surfl* pokazano na rys. 3.



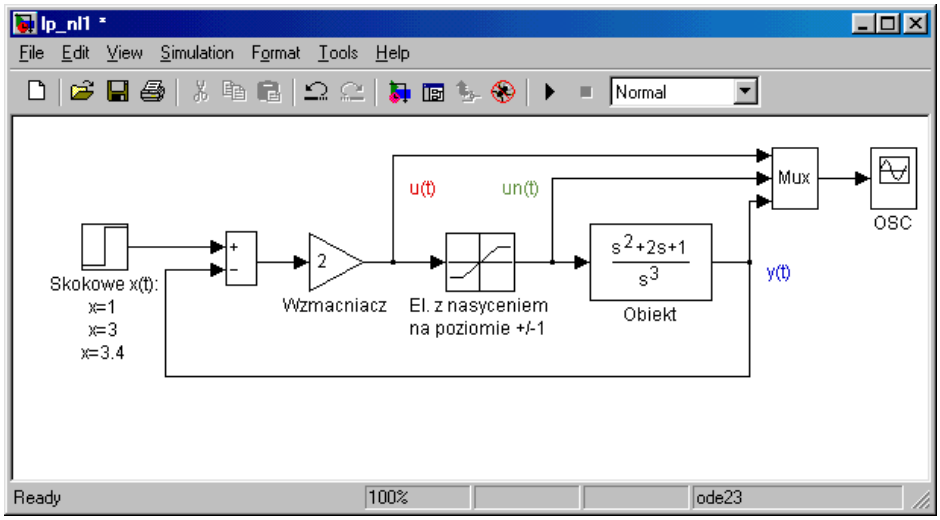
Rys. 3. Powierzchnia wykreślona za pomocą funkcji *surfl*.

2.2 Nakładka *Simulink*.

Biblioteka *Simulink* dołączana do pakietu Matlab jest graficznie zorientowanym środowiskiem projektowym wyposażonym w funkcje:

- Konstrukcji modeli dynamicznych
- Analizy działania modeli dynamicznych przy różnych wymuszeniach
- Prezentacji wyników symulacji

W pełni interaktywne środowisko pracy *Simulink* umożliwia budowę modeli dynamicznych na bazie predefiniowanych bloków funkcjonalnych dołączanych wraz z pakietem. Funkcje edycyjne ułatwiają szybkie tworzenie modeli oraz ich modyfikację. W celu umożliwienia symulacji nakładkę *Simulink* wyposażono w zestaw bloków modelujących sygnały wejściowe. Podstawowe to: *step*, *const*, *ramp*. Możliwa jest też symulacja dla bardziej złożonych wymuszeń, w tym zdefiniowanych przez użytkownika. Symulacji układów sterowania można dokonywać dla różnych metod całkowania, zadanych parametrów (krok, rząd metody, czas symulacji, *solver* i in.). Przykładowy ekran nakładki *Simulink* pokazano na rys 5 (*Przykład 2)*



Rys 5. Przykładowy ekran roboczy pakietu *Simulink (Przykład 2)*

Prezentacja wyników symulacji w nakładce *Simulink* jest możliwa dzięki bogatej bibliotece bloków wyjściowych. Najprostsze z nich to: *display*, *scope*, *to workspace* i in. Dzięki temu wyniki symulacji

mogą być przesłane np. do przestrzeni roboczej Matlab'a i tam poddane dalszemu przetwarzaniu. Możliwości nakładki *Simulink* mogą zostać rozszerzone przez dodatkowe biblioteki bloków funkcjonalnych (*blocksets)*. Przykładowe biblioteki to: *Nonlinear Control Design Blockset* – wspomaganie projektowania nieliniowych układów sterowania, *Power System Blockset* – wspomaganie projektowania układów sterowania systemami maszyn i napędów dużych mocy. *DSP Blockset* – wspomaganie projektowania systemów wykorzystujących cyfrowe przetwarzanie sygnałów.

Jedną z bibliotek funkcji pakietu Matlab wykorzystywanych najczęściej przy projektowaniu układów sterowania jest biblioteka *control toolbox*. Umożliwia ona budowę modeli dynamicznych o różnych postaciach (transmitancja, równania stanu, postać kanoniczna) a także symulację działania tychże modeli dla różnych typów wymuszeń. Biblioteka *control toolbox* ściśle współpracuje z środowiskiem *Simulink* umożliwiając szybką i efektywną konstrukcję modeli dynamicznych. Modele dynamiczne mogą być przy tym ciągłe i dyskretne. Dzięki nakładce *Simulink* możliwe jest także modelowanie układów nieliniowych.

3. Podstawy programowania

W głównym oknie Matlaba polecenia podaje się po tzw. znaku zachęty >> i zatwierdza klawiszem Enter.
Nazwa zmiennej musi rozpoczynać się literą i może stanowić dowolnej długości ciąg liter, liczb i znaków podkreślenia. Jeżeli na końcu polecenia umieszczony zostanie średnik to potwierdzenie wykonania polecenia nie zostanie bezpośrednio wyświetlone. Sprawdzenie zawartości zmiennej można uzyskać poprzez wpisanie nazwy zmiennej w oknie poleceń (**ans**-standardowa zmienna robocza, w której przechowywany jest wynik operacji gdy nie zdefiniowano nazwy zmiennej wynikowej).
Liczby w MATLAB-ie można wpisywać w postaci:
- stałozycyjnej (kropka dziesiętna) np.

-2,46 >> **a= -2.46**
10,38 >> **b=10.38**

- zmiennopozycyjnej (znak e poprzedza wykładnik potęgi10) np.

$\mathbf{1{,}63 \cdot 10^{3}}$ >> **c= 1.63e3**
$\mathbf{3{,}51 \cdot 10^{-6}}$ >> **d= 3.51e-6**

Liczby zespolone można wprowadzać na dwa sposoby:
>> **z = 7 + 2j** lub >> **z = 7 + 2i**
Przy wyświetlaniu część urojona zawsze występuje z literą **i.**
Więcej informacji – patrz DODATEK

4. Program ćwiczenia

Wykonać poszczególne polecenia i zanotować wyniki.

1) Macierze
   a) Definiowanie i generowanie macierzy
   - wprowadzić wektor wierszowy *w* i kolumnowy *k* w następujący sposób:

```
>> w=[1 2 3 4];
>> w
>> s=[7,6,5,2]
```

   - wprowadzić wektor kolumnowy *k* w następujący sposób:

```
>> k=[4;5;6;7];
>>k
```

   - wprowadzić macierz *m* w następujący sposób:

```
>> m=[2 3 1 1; 4 3 2 1 ; 4 1 5 6; 1 3 1 2];
>> m
```

   - wygenerować wektor wierszowy *n* którego elementami są liczby całkowite z przedziału <1,5>:

```
>> n=1:5
```

   - wygenerować wektor wierszowy *n1* którego elementami są liczby całkowite z przedziału <1,3> zmieniające wartości o krok 0,5 :

```
>> n1=1:0.5:3
```

b) Odwołania do elementów macierzy i operacje na macierzach

- przetransponować wektor kolumnowy *k* w wierszowy w następujący sposób:
```
>> k=k';
>> k
```
- zmienić wartość parametru macierzy *m* w 1 wierszu i 3 kolumnie:
```
>> m(1,3)=16;
>> m
```
- odwrócić macierz *m* w następujący sposób:
```
>> m^(-1)
```
- wyznaczyć rząd macierzy *m*,
```
>> rank(m)
```
- wyznaczyć ślad macierzy *m*,
```
>> trace(m)
```
- wyznaczyć macierz odwrotną do *m*,
```
>> inv(m)
```
- wyznaczyć macierz trójkątną górną macierzy *m*
```
>> triu(m)
```
- wyznaczyć macierz trójkątną dolną macierzy *m*
```
>> tril(m)
```
- wyznacznik macierzy *m*,
```
>> det(m)
```

2) mnożenie

Dla macierzy *m* oraz A i B wykonaj następujące operacje i podaj różnice

```
>> A=[4 2;3 –2;-1 6]
>> B=[2 –1;2 5;2 –2]
>> A+B
>> A.*B
>> m^2
>> m.^2
>> A\B
>> A.\B
>> A+3
>> B/2
>> B\2
>> B.\2
```

NALEŻY PAMIĘTAĆ O SKOPIOWANIU WSZYSTKICH OPERACJI Z OKNA COMMAND WINDOW !

3) Informacje o zmiennych, usuwanie zmiennych, zapis i odczyt zmiennych.

a) wyczyścić okno poleceń MATLABA:
```
>> clc;
```
b) wyświetlić informacje o zmiennych w następujący sposób:
```
>> who
>> whos
```
c) usunąć z przestrzeni roboczej zmienne *k* i *w*, w następujący sposób:
```
>> clear k w;
>> who
```
d) zapisać macierz *m* w pliku dyskowym i odczytać w następujący sposób:

- zapis z dowolnym rozszerzeniem
```
>> save m.ala m -mat
```
następnie usunąć zmienną *m* z przestrzeni roboczej poleceniem:
```
>> clear m;
>> who
```
i wprowadzić ponownie do przestrzeni roboczej (odczyt z pliku)
```
>> load m.ala m –mat
```

```
>> disp(m)
```

- zapis w formacie Matlaba

```
>> save m m
```

następnie usunąć zmienną *m* z przestrzeni roboczej poleceniem:

```
>> clear m;
>> who
```

i wprowadzić ponownie zmienną *m* do przestrzeni roboczej (odczyt z pliku)

```
>> load m m
```

- zapis w formacie ASCII

```
>> save m.txt m -ascii
```

następnie usunąć zmienną *m* z przestrzeni roboczej poleceniem:

```
>> clear m;
>> who
```

i wprowadzić ponownie zmienną *m* do przestrzeni roboczej (odczyt z pliku)

```
>> load m.txt m -ascii
```

4) Operacje na łańcuchach
   a) Wprowadzanie i wyświetlanie tekstu

```
>> s='To jest tekst';
>> disp(s)
```

   b) Wprowadzanie danych

```
>> x=input('Podaj liczbę: ')
```

5) Podstawowe operacje graficzne
   a) Wprowadzić wektor wartości zmiennej niezależnej

```
>> t=-pi:.01:pi;
```

   oraz wyznaczyć wartości funkcji *sinus* dla zadanego wektora wartości zmiennej niezależnej

```
>> y=sin(t);
```

   b) Wykreślić wykres funkcji *sinus* dla zadanych wektorów argumentów i wartości

```
>> plot(t,y);
```

   c) Dostosować skalę wykresu do zadanego przedziału zmienności funkcji

```
>> axis([-pi-0.1 pi+0.1 –1.1 1.1]);
```

   d) Wprowadzanie opisów osi, opisów wykresu:

- włączenie wyświetlania siatki współrzędnych

```
>> grid on
```

- tytuł wykresu

```
>> title('tutaj należy wprowadzić tytuł wykresu');
```

- opis osi x, który pojawi się pod osią poziomą

```
>> xlabel('tutaj należy wprowadzić opis osi x');
```

- opis osi y, który pojawi się obok osi pionowej

```
>> ylabel('tutaj należy wprowadzić opis osi y');
```

- umieszczenie legendy na bieżącym wykresie

```
>> legend('wprowadzić nazwę parametru przedstawionego na wykre-
się,2);
```

- umieszczenie na rysunku podanego tekstu w miejscu o współrzędnych *x, y*:

```
>> text(1,0.2,'komentarz do wykresu');
```

6) Tworzenie skryptów.
   a) Otworzyć *m-file editor* poleceniem *File>New>M-File*
   b) W kolejnych wierszach skryptu wpisać następujące polecenia:

```
d=[4 2 –1;3 –2 5;-1 6 –2]
g=[4 31 –19]'
x=g^(-1)*d
```

   c) Zapisać skrypt na dysku pod nazwą *lin.m* używając polecenia *File>Save As*
   d) Wyczyścić okno poleceń MATLABa poleceniem *clc* i następnie wprowadzić polecenie:

```
>> lin
```

Skrypt można również uruchomić z poziomu *m-file editor* poleceniem *Debug>Run*.

e) Napisz następujący skrypt. Przeanalizuj jego działanie.:

```
% początek skryptu
     ilosc=1
      a=rand(1,1)
     b=round(a*10)
     disp('Wylosowano liczbę z przedziału [1,10].')
     x=input('Podaj jaka to może być liczba:  ' );
        while (x~=b),
            x=input('Podaj jeszcze raz:  ' );
            ilosc=ilosc+1;
        end
     disp(['Trafiłeś za ', num2str(ilosc),' razem'])
% koniec skryptu
```

Zadanie 1

Określ działanie następujących poleceń:

```
>> tt=[1:1:5;1:2:10]
>> m(:,2)
>> m(:,1:3)
>> m(3,:)
>> m(3,4)
>> m(:)
>> m(9)
>> m(5:9)
>> m=[m;w]
>> z(:,[3,4])=m(:,1:2)
>> m(5,:)=[]
```

Zadanie 2

Napisać skrypt, którego wynikiem będzie wygenerowanie na jednym rysunku dwóch wykresów funkcji y_1=sin(t) i y_2=2*cos(3*t+1) w przedziale t=-pi:0.01:pi. Każdy przebieg powinien być wykreślony innym kolorem linii, osie powinny posiadać odpowiednie opisy, na odpowiednio zeskalowanym rysunku powinna znajdować się legenda.

Zadanie 3

Napisać skrypt sprawdzający ile spośród 6 dowolnych liczb wprowadzonych przez użytkownika mieści się w przedziale <1,20>.

Literatura

1. B. Mrozek, Z. Mrozek: *MATLAB i Simulink: poradnik użytkownika*. Helion, Gliwice, 2004.
2. A. Zalewski, R. Cegieła: *Matlab - obliczenia numeryczne i ich zastosowania*.Wydawnictwo Nakom, Poznań, 2000.
3. J. Brzózka, L. Dorobczyński: *Programowanie w Matlab*. Wydawnictwo Mikom,Warszawa, 1998.

DODATEK

`help` - system pomocy
`help` *`temat`* - podaje listę wszystkich poleceń odnoszących się do danej grupy (np.
`help ops` - Operatory i specjalne znaki.
`help general` – polecenia ogólne
`clear all` - usunięcie wszystkich zmiennych z przestrzeni roboczej

Definiowanie formatu wyświetlania liczb

format - powrót do standardowych ustawień
format short - 5 cyfr, reprezentacja stałoprzecinkowa
format long - 15 cyfr, reprezentacja stałoprzecinkowa
format short e - 5 cyfr, reprezentacja zmiennoprzecinkowa
format long e - 15 cyfr, reprezentacja zmiennoprzecinkowa
format rat - wypisywanie liczb w postaci ułamka

Operatory arytmetyczne

dodawanie	+
odejmowanie	–
mnożenie	*
dzielenie	/
potęgowanie	^

Funkcje matematyczne – argumentami funkcji mogą być liczby lub macierze

FUNKCJA	OPIS
`sin(x), cos(x), tan(x), cot(x)`	Funkcje trygonometryczne: sinus, cosinus, tangens, cotangens; argument podawany jest w radianach (stała **pi** określa liczbę π)
`sqrt(x)`	pierwiastek kwadratowy
`log(x)`	logarytm naturalny ln x
`log2(x)`	logarytm o podstawie 2 $\log_2(x)$
`log10(x)`	logarytm dziesiętny $\log_{10}(x)$
`exp(x)`	e^x
`abs(x)`	wartość bezwzględna lub moduł liczby zespolonej
`angle(x)`	argument liczby zespolonej
`real(x)`	część rzeczywista liczby zespolonej
`imag(x)`	część urojona liczby zespolonej
`conj(x)`	liczba zespolona sprzężona

Operacje na macierzach:

- **operacje macierzowe** – wykonywane na całych macierzach zgodnie z regułami algebry,
- **operacje tablicowe** – wykonywane na poszczególnych elementach macierzy.

Jeśli jeden z argumentów jest skalarem to mówimy o operacjach tablicowych.

operacja	macierzowa	tablicowa	uwagi
dodawanie	+	+	
odejmowanie	–	–	
mnożenie	*	.*	
potęgowanie	^	.^	
dzielenie prawostronne	/	./	A./B => A(i,j)/B(i,j)
dzielenie lewostronne	\	.\	A.\B => B(i,j)/A(i,j)

Funkcje do konstruowania macierzy

FUNKCJA	OPIS
`eye(x)`	Macierz jednostkowa (jedynki na przekątnej)
`ones(x)`	Macierz o elementach równych 1
`rand(x)`	Macierz losowa o rozkładzie równomiernym
`rands(x)`	Macierz losowa o rozkładzie normalnym
`zeros(x)`	Macierz z elementami zerowymi

Wyrażenia logiczne

Wyrażenia logiczne służą do porównania wartości zmiennych o tych samych rozmiarach.
Gdy porównywane są skalary to jeśli wyrażenie logiczne jest prawdziwe zwracana jest wartość **1**, jeśli fałszywe - wartość **0**.

Operatory relacyjne

operator	znaczenie
`x == y`	x = y
`x ~= y`	x ≠y
`x < y`	x < y
`x > y`	x > y
`x <= y`	x ≤y
`x >= y`	x ≥y

Operatory logiczne

operator	znaczenie
`x | y`	x lub y (OR)
`x & y`	x i y (AND)
`~x`	nie x (NOT)

Grafika

Okna graficzne

`figure` - tworzy nowe okno graficzne,
`figure(n)` - tworzy nowe okno graficzne o numerze **n**,
`close` - zamyka aktywne okno graficzne,
`close all` - zamyka wszystkie okna graficzne,
`clf` - czyści zawartość aktywnego okna graficznego,
`subplot(m,n,p)` - dzieli okno graficzne na mniejsze prostokątne okienka umieszczając je w **m**-wierszach i **n**-kolumnach, **p** – numer aktywnego okienka, w każdym okienku można umieścić odrębny wykres.

Wykresy dwuwymiarowe (przykłady)

nazwa	opis
`plot`	Skala liniowa obu osi
`loglog`	Skala logarytmiczna obu osi
`bar`	Wykres słupkowy
`hist`	histogram
`rose`	Histogram kołowy
`polar`	Wykres kołowy
`stairs`	Wykres schodkowy
`semilogx`	Skala logarytmiczna osi x
`semilogx`	Skala logarytmiczna osi y
`fplot`	Wykres funkcji ciągłej

`plot(x,y)` - rysuje wykres y = f(x),
`plot(y)` - rysuje wykres elementów wektora y, przyjmując za x kolejne numery indeksów elementów wektora zaczynając od 1,
`plot(x,y,'kolor_rodzaj_linii')` - rysuje wykres y = f(x) z określeniem sposobu rysowania linii,
`plot(x1,y1,x2,y2,...)` - rysuje w jednym oknie wiele wykresów,
`plot(x1,y1,'kolor_rodzaj_linii1',x2,y2,'kolor_rodzaj_linii2', ...)` - rysuje w jednym oknie wiele wykresów z określeniem sposobu rysowania linii każdego z nich.
`hold on` - powoduje dodanie nowego wykresu do wykresu już istniejącego
`title('tekst','FontSize',10)` – wprowadzenie tytułu wykresu czcionką o wielkości 10 pkt

Zestawienie kolorów i rodzajów linii

symbol	kolor		symbol	rodzaj linii
y	żółty		-	ciągła (domyślna)
m	karmazynowy		--	kreskowana
c	turkusowy		:	kropkowana
r	czerwony		-.	kreska-kropka
g	zielony		+	krzyżyk
b	niebieski		*	gwiazdka
w	biały		.	kropka
k	czarny		o	kółko
			x	iks

Instrukcja warunkowa

if wyrażenie
    polecenia
elseif wyrażenie
    polecenia
else
    polecenia
end

Wyrażenia są relacją lub wyrażeniem logicznym. Wynik wyrażeń użytych po if oraz elseif musi być liczbą.

Instrukcje iteracyjne

Nieokreślona liczba obiegu pętli

while wyrażenie
    polecenia
end

Ścisle określona liczba obiegu pętli

for zmienna= wyrażenie
    polecenia
end