ANALIZA DANYCH I PROCESÓW

Mgr inż. Paweł Wojciech Herbin

SZCZECIN 29 LUTEGO 2016

Spis treści

1. Wprowadzenie 5
2. MATLAB –wprowadzenie do interfejsu 6
3. Praca w trybie bezpośrednim 7
   3.1. Wprowadzanie zmiennych 7
   3.2. Przykłady poleceń 9
   3.3. Liczby rzeczywiste i ich formaty 9
   3.4. Zapis macierzy 10
4. Funkcje standardowe 14
5. Wykresy 15
6. Skrypt 17
7. Programowanie 18

1. Wprowadzenie

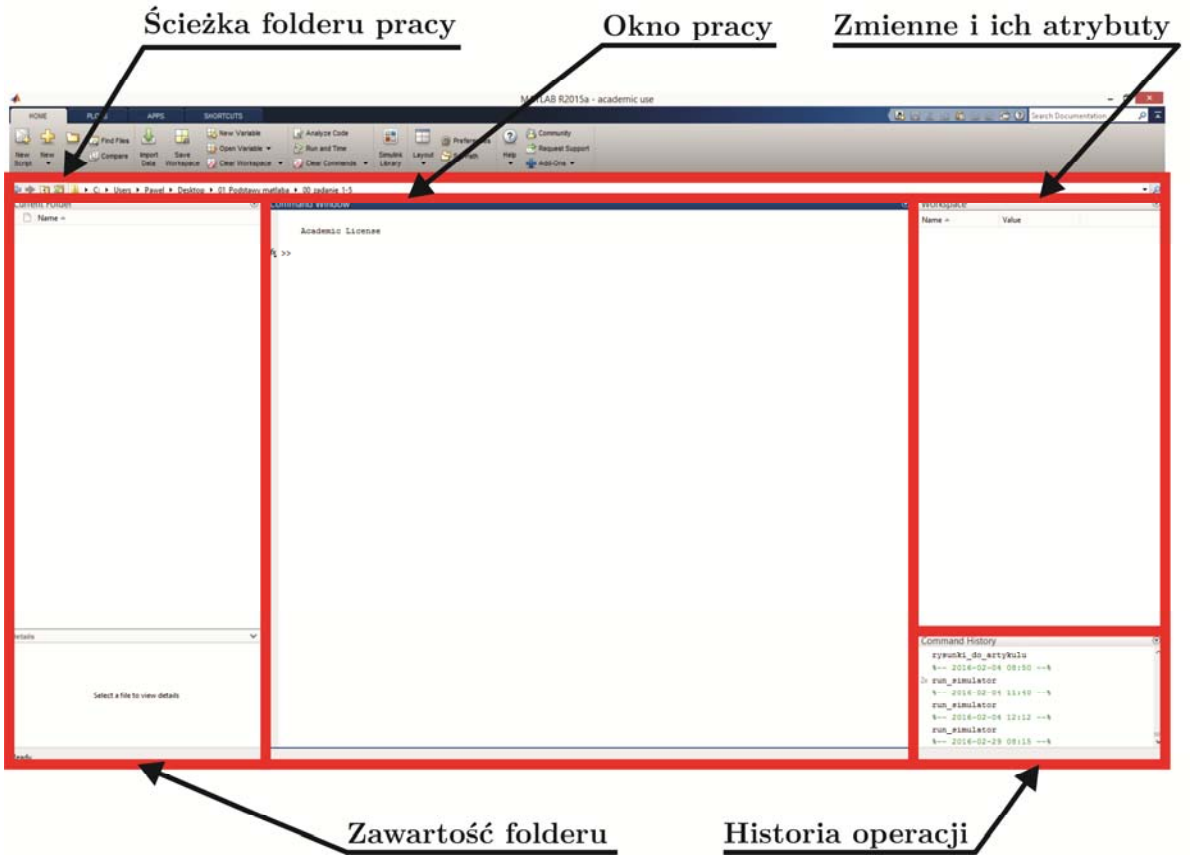
Przedmiot „Analiza danych i procesów” winien przybliżyć Państwu zagadnienia związane z opracowaniem dużej ilości danych. Zajęcia prowadzone będą w programie MATLAB. Matlab jest środowiskiem, w którym zaimplementowano zawansowane metody numeryczne, prezentację graficzną danych oraz język programowania. Celem niniejszego opracowania jest przedstawienie podstawowych metod numerycznych. Możliwości pakietu MATLAB są ogromne, w związku z tym, jeżeli ktoś z Państwa sięgnie w przyszłości do przedstawionych metod w niniejszym opracowaniu cel zajęć zostanie osiągnięty.

Układ przedstawionych w opracowaniu zadań składa się ze wstępu teoretycznego, opisu problemu oraz przedstawienia oczekiwanego rozwiązania.

Autor

2. MATLAB –wprowadzenie do interfejsu

1) Proszę uruchomić program MATLAB. Pierwsze uruchomienie MATLAB'a trawa nawet kilka minut w zależności od wydajności komputera na którym pracujemy.
2) Budowa okna MATLAB



Rysunek 1 Budowa okna MATLAB

Jeżeli okno wygląda inaczej niż na rysunku 1 należy wykonać następujące działania: Na wstążce należy kliknąć: *Layout -> Default* następnie *Layout -> Command History -> Docked*

3. Praca w trybie bezpośrednim

W obszarze Command Window wprowadza się komendy. Gotowość MATLAB'a na przyjęcie polecenia sygnalizowana jest znakiem zachęty >>. Po wpisaniu komendy zostanie ona wykonana po naciśnięciu klawisza ENTER.

3.1. Wprowadzanie zmiennych

ZASADY TWORZENIA NAZW ZMIENNYCH

Nazwa zmiennej musi zaczynać się od litery i może składać się z dowolnej liczby liter, cyfr i znaków podkreślenia, przy czym Matlab uwzględnia tylko 31 pierwszych znaków nazwy. Rozróżniane są duże i małe litery. Nie jest wymagane deklarowanie zmiennych, ani określanie ich typu (rozmiaru).

ZASADY ZAPISU KOMEND

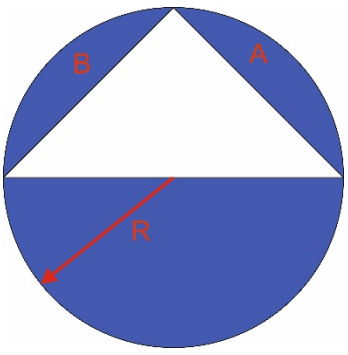
- jeżeli obliczenia mają być wykonane, a wynik polecenia nie ma być wyświetlony to należy na końcu polecenia umieścić znak średnika, np. A=1;
- jeśli w jednym wierszu chcemy napisać kilka poleceń , to możemy:
    a) oddzielić je średnikami – jeżeli nie chcemy oglądać wyników,
    b) oddzielić je przecinkami – jeżeli chcemy widzieć wyniki.
- znakiem dziesiętnym jest kropka.

Aby wprowadzić zmienną do obszaru roboczego należy podać odpowiednią komendę:

a) Skalar – macierz o wymiarach 1x1 ,np. B=1
b) Wektor kolumnowy 1 x n ,np. C=[1; 2; 3]
c) Wektor wierszowy n x 1 ,np. D=[4 5 6]
d) Macierz n x m ,np. E= [1 2 3; 4 5 6; 7 8 9; 10 11 12]

ZADANIE 1

Wprowadź zmienne niezbędne do obliczenia pola, które pozostanie po wycięciu z koła trójkąta równoramiennego, którego jeden z boków jest oparty na średnicy koła o promieniu R=100. Oblicz następujące wartości:



Pole koła, średnica, podstawa trójkąta, wysokość trójkąta, pole trójkąta, pole wycinka A oraz B, pole wynikowe

Następnie wykonaj wprowadzone polecenia tak aby wyświetlić tylko wynik pola wynikowego.

Rysunek 2 Koło z wyciętym trójkątem

ZADANIE 2

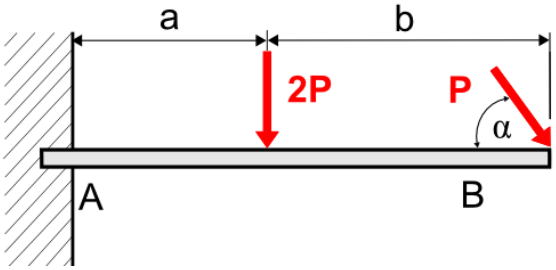
Obliczenie sił reakcji w podporach (mechanika, równania statyki)

Użyj funkcji:

sin(), cos()- kąt w radianach,

sqrt()-pierwiastek kwadratowy,

^ - operator potęgowania.



Rysunek 3 Schemat obciążenia belki

Dane:

a=1 m;
b=2 m ;
P=1000 N ;
α= 30 [deg];

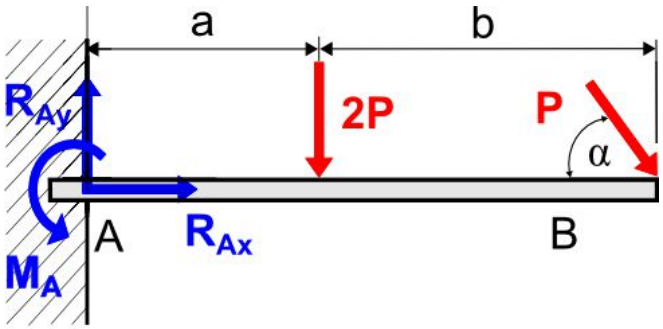
Szukane:

R_A=?
M_A =?

Wybieramy układ współrzędnych i uwalniamy od więzów.
Zaznaczamy wszystkie siły czynne.

Siłę P skierowaną pod kątem do poziomu rozbijamy na dwie składowe P_x i P_y następnie opisujemy ich wartość przy użyciu funkcji trygonometrycznych:

$$P_x = \cos(\alpha) * P$$
$$P_y = \sin(\alpha) * P$$



Rysunek 4 Reakcje podpór

W zadaniu są trzy niewiadome (R_{Ax}, R_{Ay}, M_A).

3.2. Przykłady poleceń

- Podstawienie: a=3; powoduje utworzenie zmiennej a o wartości 3. UWAGA: Średnik po poleceniu powoduje, że wartość będąca wynikiem nie będzie wyświetlana na ekranie.
- b=sin(a) b = 0.1411 oblicza wartość funkcji sinus dla zmiennej a, wynik zapisuje do zmiennej b i wyświetla na ekranie.
- Jeżeli nie podano nazwy zmiennej to wynik działania jest umieszczany w standardowej zmiennej ans, np.: cos(pi/3) ans = 0.5000
- Utworzona (zdefiniowana) zmienna jest pamiętana od momentu utworzenia, aż do chwili jej usunięcia. Możliwa jest przy tym nie tylko zmiana wartości, ale również rozmiaru zmiennej. Nazwy zmiennych i informacje o nich można uzyskać wywołując funkcje who i whos.
- Usunięcie zmiennej z pamięci:
  clear a - usuwa zmienną a;
  clear - usuwa wszystkie zmienne znajdujące się w pamięci.
- Zapisanie zmiennych na dysku: save nazwa_pliku (domyślnie przyjmowane jest rozszerzenie .mat). • Wczytanie danych z pliku dyskowego: load nazwa_pliku
- Korzystanie z podręcznej pomocy podającej opis funkcji: help nazwa_funkcji.
- Zawartość aktualnego katalogu można wyświetlić używając funkcji dir lub ls.
- Do zmiany katalogu służy polecenie: cd nazwa_katalogu

3.3. Liczby rzeczywiste i ich formaty

Podstawowym typem dla elementów macierzy wykorzystywanym przez MATLAB są liczby rzeczywiste. Maksymalną i minimalną wartość liczby rzeczywistej dodatniej można poznać za pomocą funkcji realmax i realmin. Do określenia sposobu, w jaki liczby rzeczywiste są przedstawione na ekranie służy polecenie format postać liczby, gdzie postać liczby określa postać, w jakiej liczby rzeczywiste będą wyświetlane na ekranie.

PRZYKŁAD

Przedstaw liczbę 2.478513 w różnej postaci używając funkcji format.

» format short

» 2.478513

ans = 2.4785

» format short e

» 2.478513

ans = 2.4785e+000

» format long

» 2.478513

ans = 2.47851300000000

» format long e

» 2.478513

ans = 2. 47851300000000e+000

3.4. Zapis macierzy

DEFINICJA MACIERZY PRZEZ WYPISANIE ELEMENTÓW:

Aby zapisać macierz $A = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 1 \end{bmatrix}$

Należy zapisać: A=[2,2,2,1; 1,2,3,1];

Aby zapisać macierz $A = [2 \quad 2 \quad 2 \quad 1 \quad 1 \quad 2 \quad 3 \quad 1]$

Należy zapisać: A=[2 2 2 1 1 2 3 1] lub A=[2,2,2,1,1,2,3,1]

Poszczególne elementy macierzy oddziela się spacjami lub przecinkami, a wiersze średnikami lub umieszcza się je w oddzielnych liniach.

DEFINICJA MACIERZY PRZEZ WYGENEROWANIE ELEMENTÓW: A=[MIN:KROK:MAX]

Polecenie generuje wektor poczynając od elementu o wartości min, kończąc na elemencie o wartości max z krokiem krok. Jeżeli parametr krok zostanie pominięty, przyjmuje się, iż krok=1. Aby zapisać macierz: $A = [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10]$

Należy zapisać: A=[1:10] lub A=[1:1:10]

ĆWICZENIE

Wygeneruj macierz dwuwierszową o wyrazach od 1 do 10 w pierwszym wierszu i o wyrazach od 2 do 20 (co 2) w wierszu drugim tak aby wynikowa macierz wyglądała w sposób następujący

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 & 20 \end{bmatrix}$$

DEFINICJA MACIERZY WYKORZYSTUJĄC ELEMENTY INNYCH MACIERZY:

Utwórz macierz D budując ją ze zdefiniowanych macierzy A, B i C.

A=[1 4 1; 2 0 1];

B=[3 1; 4 1];

C=[1 2 2 0 1; 2 4 7 1 0];

D=[A B; C]

UWAGA: Przy takim budowaniu macierzy należy pamiętać o zgodności wymiarów.

Wymiar i wyświetlanie macierzy:

[n,m]=size(A) - zwraca liczbę kolumn n i wierszy m macierzy A;

n=length(B) - zwraca wymiar wektora B (lub większy z wymiarów macierzy B);

A lub disp(A) - pokazuje macierz A na ekranie;

FUNKCJE WSPOMAGAJĄCE KONSTRUOWANIE MACIERZY

Definicja macierzy jednostkowej:

eye(n) – tworzy macierz kwadratową jednostkową o wymiarach [nxn]

eye(n,m) – tworzy macierz jednostkową o wymiarach [nxm]

Definicja macierzy wypełnionej jedynkami:

ones(n) – tworzy macierz kwadratową wypełnioną jedynkami o wymiarach [nxn]

ones(n,m) – tworzy macierz wypełnioną jedynkami o wymiarach [nxm]

Definicja macierzy wypełnionej zerami:

zeros(n) – tworzy macierz kwadratową wypełnioną zerami o wymiarach [nxn]

zeros(n,m) – tworzy macierz wypełnioną zerami o wymiarach [nxm]

ZADANIE 3

Zapisz macierz E za pomocą funkcji wspomagających tworzenie macierzy:

$$E = \begin{bmatrix} 1 & 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

DOSTĘP DO ELEMENTÓW MACIERZY

Odwołanie do elementów:

Przykład:

A=[1 2 3; 0 9 8; 1 1 0]

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 9 & \mathbf{8} \\ 1 & 1 & 0 \end{bmatrix}$$

Aby odwołać się do pogrubionego elementu należy zapisać:

A(2,3) - odwołanie do elementu w wierszu 2 i kolumnie 3;

ans = 8

PODSTAWOWE FUNKCJE OPERUJĄCE NA MACIERZACH

mean(A) – oblicza średnią w poszczególnych kolumnach macierzy A

max(A) – zwraca największą wartość w poszczególnych kolumnach macierzy A

min(A) – zwraca najmniejszą wartość w poszczególnych kolumnach macierzy A

DZIAŁANIA NA MACIERZACH

Suma i różnica macierzy :

PRZYKŁAD

Zdefiniuj macierz $A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix}$ oraz macierz $B = \begin{bmatrix} 5 & 4 & 3 \\ 0 & 1 & 2 \end{bmatrix}$

Suma macierzy:

C=A+B

Różnica macierzy:

D=A-B

Dodanie do elementów macierzy A liczby 5

Mnożenie macierzy:

PRZYKŁAD

Zdefiniuj macierz $A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix}$ oraz macierz $B = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Mnożenie macierzy:

E=A*B

Mnożenie macierzy przez skalar

F=A*5

ODWARANIE MACIERZY i TRANSPOZYCJA

Zdefiniuj macierz $A = \begin{bmatrix} 1 & 3 & 5 \\ 9 & 8 & 7 \\ 5 & 5 & 4 \end{bmatrix}$

Odwracanie macierzy:

G=inv(A)

Transpozycja macierzy:

H=A’ lub H=trans(A)

ZADANIE 4

Oblicz wartość macierzy X:

Y= A*X+B

UWAGA w rachunku macierzowym zamiast dzielenia należy mnożyć lewostronnie przez macierz odwrotną.

Przekształć wzór do postaci X=….

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 9 & 8 & 7 \\ 5 & 5 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 30 \\ 10 \\ 40 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1 \\ 5 \\ 7 \end{bmatrix}$$

ZADANIE 5

Wygeneruj macierz A przy wykorzystaniu funkcji ułatwiających tworzenie macierzy.

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

FUNKCJE PRZEKSZTAŁCAJCE MACIERZE

A=diag(x) – wstawia składniki wektora x na głównej przekątnej macierzy zerowej A,

x=diag(A) – tworzy wektor x z głównej przekątnej macierzy A

inv(A) – tworzy macierz odwrotn do macierzy A

tril(A) – tworzy z macierzy A macierz trójkątną dolną (zerowanie elementów nad główną przekątną)

triu(A) – tworzy z macierzy A macierz trójkątnągórną

rot90(A) – obraca macierzy A o 90 o w kierunku

repmat(A,m,n) – powiela macierz A m razy w pionie i n razy w poziomie

reshape(A,m,n) – tworzy macierz o m wierszach i n kolumnach

W MATLABIE ISTNIEJDWA RODZAJE OPERACJI NA MACIERZACH:

- zgodne z zasadami algebry macierzy (operatory macierzowe),
- operacje tablicowe traktujce macierze jak zwykłe tablice liczbowe.

Operatory tablicowe funkcjonuj w taki sposób,

że działaniom operatorów podlegajodpowiadające sobie pary elementów obu macierzy (macierze muszą miten sam wymiar). Operatory tablicowe poprzedzone są zawsze kropką.

Operator arytmetyczny	Operator macierzowy	Operator tablicowy
Suma	$A + B$	$A + B$
Różnica	$A - B$	$A - B$
Mnożenie	$A * B \neq B * A$ *nie jest przemienne*	$A.* B = B.* A$ *jest przemienne*
Dzielenie prawostronne	$A/B = B * A^{-1}$	$A./B$
Dzielenie lewostronne	$A\backslash B = A^{-1} * B$	$A.\backslash B$
Potęgowanie	$A^2 = A * A$	$A.\hat{}2$
Odwracanie	$A^{-1} = inv(A)$	$A.\hat{} - 1$

ZADANIE 6

Przetestuj WSZYSTKIE!! operatory macierzowe i tablicowe dla następujących macierzy i porównaj wyniki:

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix}; B = \begin{bmatrix} 1 & 3 & 5 \\ 9 & 8 & 7 \\ 5 & 5 & 4 \end{bmatrix}$$

4. Funkcje standardowe

Narzędzie MATLAB w swoich bibliotekach ma bardzo wiele funkcji. Funkcje zaimplementowane w środowisku MATLAB obsługują macierze, wiec z uwagi na tą właściwość argumentem każdej funkcji może być macierz. Wynikiem działań funkcji jest też macierz. Kilka z podstawowych funkcji przedstawiono w poniższej tabeli.

Podstawowe funkcje MATLAB	
round()	Zaokrąglenie do najbliższej liczby całkowitej
real() ,imag()	Część rzeczywista i urojona liczby zespolonej
angle()	Argument liczby zespolonej

abs()	Moduł liczby zespolonej, wartość bezwzględna
log(), log2(), log10()	Logarytm naturalny, logarytm o podstawie 2, logarytm o podstawie 10
sqrt()	Pierwiastek kwadratowy
asin(),acos()	Funkcje cyklometryczne(wynik w radianach)
sin(), cos(), tan(), cot()	Funkcje trygonometryczne(argumenty w radianach)
Funkcje operujące na wektorach	
max()	Zwraca największy element wektora (każdej kolumny macierzy)
min()	Zwraca najmniejszy element wektora (każdej kolumny macierzy)
sum()	Zwraca sumę elementów wektora (każdej kolumny macierzy)
prod()	Zwraca iloczyn elementów wektora (każdej kolumny macierzy)
mean()	Zwraca średnią arytmetyczną elementów wektora (każdej kolumny macierzy)

5. Wykresy

Okno graficzne można utworzyć na trzy podstawowe sposoby:

- Poprzez menu **File/New/Figure,**
- Poprzez wywołanie funkcji generującej okno **figure(Numer_okna)**
- Poprzez wywołanie jednej z funkcji generującej wykres.

Podczas pracy może być otwarte wiele okien graficznych jednakże tylko jedno z nich jest aktywne. Każde okno ma swój unikalny numer.

Oknami zarządza się z wykorzystaniem następujących komend:

- **figure(Numer_okna)** – uaktywnia okno o numerze **Numer_okna**
- **close(Numer_okna)** – zamknięcie okna o numerze **Numer_okna**
- **clf** – usunięcie zawartości aktywnego okna
- **subplot(m,n,nr)** – podział okna na części. **m** wierszy **n** kolumn i aktywacja elementu o numerze **nr**
- **hold on/off** -włącza/wyłącza tryb zachowania zawartości okna graficznego

W trybie bezpośrednim możliwy jest wybór aktywnego okna za pomocą myszki.

Do wyświetlania grafiki dwuwymiarowej (krzywych) służy funkcja plot. W zależności od liczby argumentów wywołanie funkcji może mieć różną postać.

plot(X,Y) - rysuje wykres funkcji y=f(x) wektory X i Y muszą mieć taką samą długość

plot(Y) – rysuje wykres elementów wektora y, przyjmując za X kolejne liczby całkowite począwszy od 1

PRZYKŁAD

```
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);
figure
plot(x,y1,x,y2,'--',x,y3,':')
```

ZADANIE 7

Projektowane urządzenie może być sterowane za pomącą dwóch funkcji sterujących:

a) sind(t) – argument t podany w stopniach
b) $\frac{1}{\sqrt{t}+1}$

Sterowanie może odbywać się na jeden z czterech sposobów:

1. Włączona jest tylko funkcja a)
2. Właczona jest tylko funkcja b)
3. Sygnałem sterującym jest suma funkcji a) i b)
4. Sygnałem sterującym jest iloczyn funkcji a) i b)

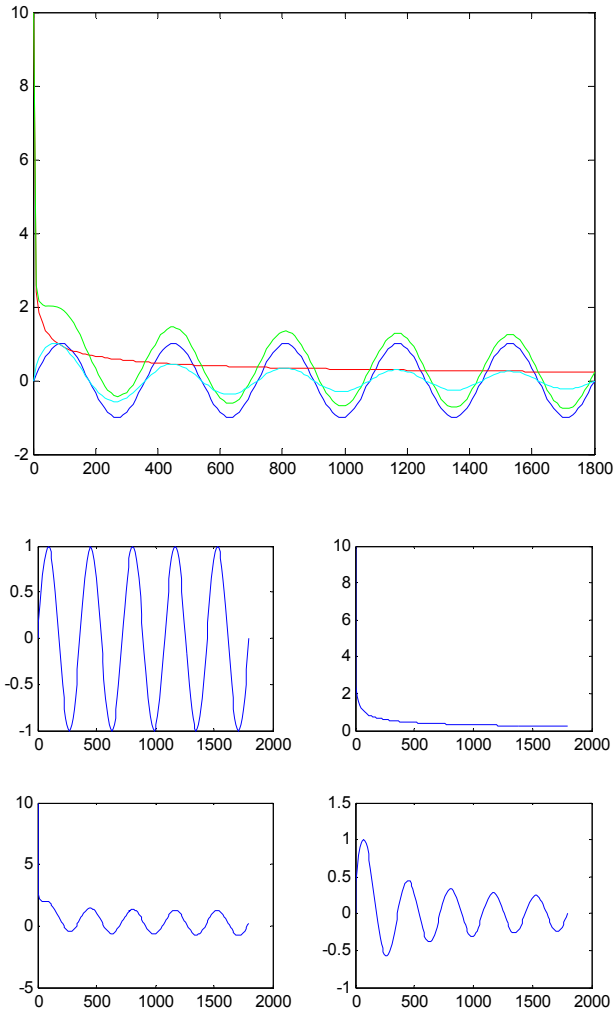
Sporządź wykres przebiegu funkcji dla każdego z czterech przypadków w przedziale od zera do 5 cykli funkcji a). Wykresy umieść w czterech oknach

Okno 1- przebieg funkcji a)

Okno 2- przebieg funkcji b)

Okno 3- przebieg wszystkich typów sterowań na jednym wykresie

Okno 4- podziel okno na cztery części i w części 1 wyświetl sposób 1, w części 2 sposób 2 itd.



6. Skrypt

W Matlabie można pracować w ten sposób, że wszystkie polecenia potrzebne do rozwiązania problemu, reguły, funkcje, itp., pisze si w oknie tekstowego edytora, a potem cały ten tekst zapisuje w pliku z rozszerzeniem „.m". Tak powstaj tzw. m-pliki, tak tworzy się programy/procedury (nazywane skryptami) i funkcje. Do przechowywania takiej twórczości przeznaczony jest specjalny folder. Matlab wie, że musi tam szuka programu (lub funkcji), gdy jest on wywołany (przez wpisanie nazwy pliku bez rozszerzenia „.m"). My powinniśmy mieć świadomość, że tylko tam możemy zapisywać tworzone m-pliki.

Skrypt najlepiej utworzy wybierając z menu opcję File/New/M-File. Otwiera się nowe puste okienko. Wpisywane tam polecenia nie są od razu wykonywane. Stanowi zawarto m-pliku rozumianą jako treść programu. Zapisanie tej zawarto ci na dysku odbywa si w klasyczny sposób: File/Save As... . Pamiętać trzeba, aby skontrolować miejsce zapisu i upewni się, że będzie to folder przeznaczony na m-pliki.

Struktura skryptu jest bardzo prosta. Tak naprawdę można powiedzieć, że w skrajnym przypadku nie ma żadnej struktury. Jest po prostu ciąg poleceń i instrukcji. Poleca się jednak **przestrzegać pewnych zasad**. Zasady te, to:

- rozpoczynanie skryptu od komentarzy opisujących jego przeznaczenia i metodę stosowaną do obliczeń,
- umieszczenie w tym opisie **nazw macierzy i zmiennych**, które muszą znaleźć się w przestrzeni roboczej Matlaba, by program, czyli skrypt, mógł na nich operować,
- umieszczenie w tym miejscu także **opisu macierzy i zmiennych** tworzonych przez program,
- opatrywanie sekwencji działa lub ciekawszych instrukcji odpowiednim komentarzem wyjaśniającym, co ułatwi później ewentualne poprawiane skryptu lub wykorzystanie jego zawartości w innych tworzonych m-plikach (kopiuj/wklej).

Komentarze w skrypcie poprzedza się znakiem procentu %. Znak ten ma znaczenie tylko do końca wiersza. Treść komentarza, który umieszczony jest na początku skryptu traktowana jest jak tzw. „help" skryptu (objaśnienie). Można ten „help" wywołać bez uruchamiania skryptu poleceniem

help nazwa_skryptu

Polecenia skryptu muszą odnosić się do zmiennych, które znajduj się w przestrzeni roboczej Matlaba (w pamięci Matlaba) lub do tych zmiennych, które utworzone są przez sam skrypt. Te zmienne w wyniku działania skryptu będą także znajdować się w przestrzeni roboczej.

7. Programowanie

Podczas tworzenia skryptów wykorzystuje się także język programowania Matlaba. Na zajęciach poznamy i zastosujemy podstawowe jego konstrukcje (typowe dla wielu innych języków). Zestawienie operatorów relacji i operatorów logicznych stosowanych w wyrażeniach języka podaje tabela.

OPERATOR RELACJI	
A==B	A=B
A~=B	A≠B
A<B	A<B
A>B	A>B
A<=B	A≤B
A>=B	A≥B

OPRERATOR LOGICZNY	
A|B	A lub B
A & B	A i B
~A	NIE A

Poniżej opisane są krótko podstawowe instrukcje języka.

Instrukcja for

Instrukcja for pozwala na powtarzanie wybranego fragmentu kodu określoną ilość razy.

Szablon instrukcji for (uwaga na przecinek):

```
for zmienna_iterowana = macierz_wartosci ,
Kod do wielokrotnego powtarzania
end
```

Pętle w wybranych przypadkach można przerywać przy pomocy instrukcji break.

Przykład:

```
a=zeros(10,5); % alokacja pamieci
for i=1:10,
for j=1:5,
a(i,j)=i*j;
end
end
```

Instrukcja while

While stanowi pętle warunkową, fragment kodu w pętli będzie wykonywany dopóki jest spełnione wyrażenie warunkowe. Szablon instrukcji while (uwaga na przecinek):

while wyrazenie_warunkowe,

Kod do wielokrotnego powtarzania

*en*d

Przykład:

```
licznik1=0;
 licznik2=0;
 suma=0;
while (licznik1<10 & licznik2<10),
licznik1=licznik1+0.1;
licznik2=licznik2+0.2;
suma=licznik1+licznik2;
end
```

Instrukcja warunkowa if

Instrukcja pozwala na wykonanie jednego z kilku fragmentów kodów zawartego pomiędzy instrukcjami if, elseif, else. Wybór realizowanego kodu zależy od spełnienia odpowiednich wyrażeń warunkowych, gdy żadne z nich nie jest spełnione jest wykonywany kod występujący za operatorem else. Szablon instrukcji if:

```
If wyrazenie_warunkowe_1
Kod wersja 1
elseif wyrazenie_warunkowe_2
Kod wersja 2
elseif wyrazenie_warunkowe_3
Kod wersja 3
.....
else
Kod wersja N
end
```

Przykład:

```
%% y=a*x^2+b*x+c
a=1;
b=2;
c=3;
wyznacznik=b^2-4*a*c;
if wyznacznik>0
x1=(-b+sqrt(wyznacznik))/(2*a);
x2=(-b-sqrt(wyznacznik))/(2*a);
elseif wyznacznik==0
x1=-b/(2*a);
x2=x1;
else
x1=NaN;
x2=NaN;
end
```

Instrukcja switch-case

W przypadku listy znanych argumentów wywołania wygodnie jest skorzystać z funkcji switch-case. Szablon instrukcji switch-case:

```
switch p
case 1
instrukcja 1
case 2
instrukcja 2
otherwise
inna instrukcja
end
```