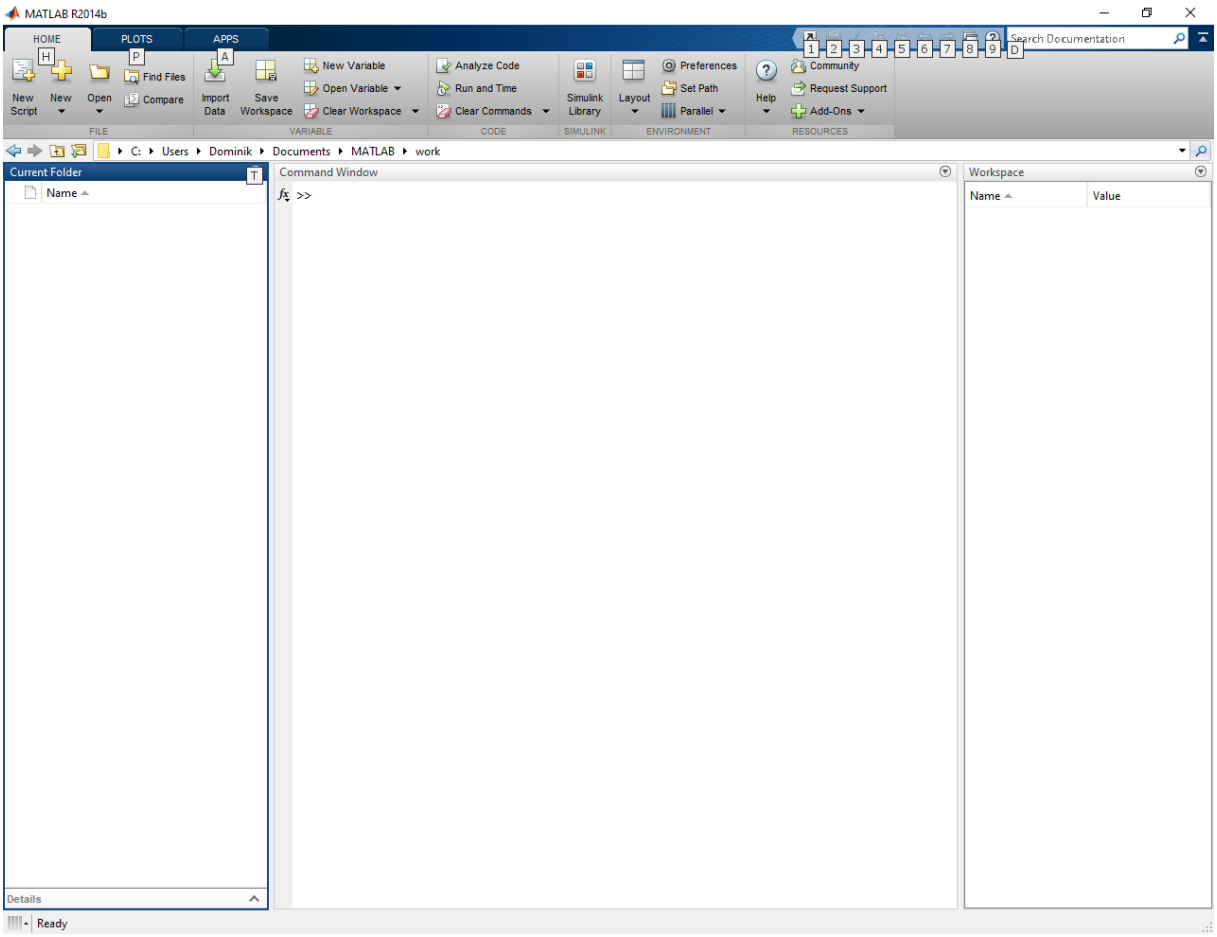
Instrukcja na zajęcia laboratoryjne z przedmiotu „Podstawy robotyki”

Zajęcia I oraz II – „Wstęp do Matlaba”

1. Podstawowe okna programu Matlab



Po uruchomieniu programu można zauważyć 3 główne bloki:
- Current Folder – W tym miejscu są wyświetlane wszystkie pliki w aktualny folderze do których można się odwołać w komendach programu. Folder ten jest ustawiany w pasku adresu znajdującym się pod paskiem narzędzi.
- Command Window – Miejsce, w którym użytkownik wpisuje komendy programu
- Workspace - Zawiera wszystkie zmienne w pamięci programu. Po dwukrotnym kliknięciu w wybraną zmienną otwiera się okno „Variables”, w którym można podglądnąć jej strukturę w formie macierzy (Każda zmienna w programie Matlab jest traktowana jako macierz. Wartość skalarna jest macierzą o wymiarach 1x1)

Uwaga: Gdy domyślana wielkość czcionki w bloku Command Window jest zbyt mała można ją zmienić klikając przycisk „Preferences” w zakładce „Home” wstążki programu Matlab. (W starszych wersjach wybierając kolejno File -> Preferences). Następnie wybierając z listy po lewej stronie „Matlab -> Fonts” aktualny styl oraz rozmiar czcionki używany to pisania kodu można zmodyfikować w bloku „Desktop code font”

Podczas pracy często jest również używane okno „Command History”, które prezentuje historię wykonywanych poleceń. Jeżeli nie zostało ono uruchomione podczas otwierania programu, można je otworzyć wydając w „Command Window” komendę „commandhistory”.

2. Podstawowe operacje

Najprostsze operacje przypisania wartości:

```
>> A=2;
>> B=3
>> C=A+B
```

Powyższy kod przypisuje wartość 2 do zmiennej A. Znak „;” na końcu oznacza, że wynik operacji nie zostanie wyświetlony w oknie komend. W przypadku przypisania wartości zmiennej B, takie wyświetlenie będzie miało miejsce. Następnie do zmiennej C zostanie przypisana wartość sumy dwóch wcześniej użytych zmiennych. Warto zwrócić uwagę, że wszystkie 3 zmienne pojawią się w oknie Workspace, gdzie można podglądnąć ich watość. Inną metodą na sprawdzenie wartości zmiennej jest wpisanie jej w oknie komend bez znaku „;” na końcu i kliknięcie przycisku enter.

3. Zmienne symboliczne i zapis macierzy

W programie matlab można operować nie tylko na zmiennych posiadających określoną wartość, lecz również na zmiennych symbolicznych (wymaga zainstalowanego Toolboxa Math). Aby wprowadzić zmienne symboliczne należy poprzedzić je słowem kluczowym syms, a następnie wypisać, oddzielając je spacjami.

```
>> syms a b c d e f g h;
```

Macierze zapisujemy w nawiasach kwadratowych podając kolejne elementy wiersza macierzy rozdzielone znakiem spacji lub tabulacji. Natomiast wiersze oddzielone są od siebie znakiem „;”. Istnieje również możliwość podania wielu poleceń w jednej linii tak jak pokazano poniżej:

```
>> A=[a b;c d];B=[e f;g h];
>> C=A*B
>> 2*A
```

Pierwsza linia powyższego kodu tworzy macierze A i B o wymiarach 2x2 składające się ze zmiennych symbolicznych. W drugiej linii macierz C jest wynikiem mnożenia powyższych macierzy. Ponieważ zostały użyte zmienne symboliczne, efektem jest otrzymanie wzoru na mnożenie macierzy.
Obliczenia na zmiennych symbolicznych można łączyć z wartościami liczbowymi tak jak wykonano to w trzeciej linii kodu.

Obliczenia algebraiczne w programie matlab mogą być również zapisane bez wcześniejszej deklaracji zmiennych symbolicznych. Wyrażenia takie zostanę automatycznie uproszczone:

```
>> f=sym(2*a+5*b*a-1.5*a+8*b+2.5*a-b)
f =
3.0*a + 7*b + 5*a*b
```

Zmienny symbolicznych można używać również do rozwijania oraz upraszczania wyrażeń algebraicznych. Pokazuje to przykład poniżej:

```
>> f=(a+b)^2
>> expand(f)
>> g=2*a+5*b*a-1.5*a+8*b+2.5*a-b
>> collect(g, a)
>> h=1/(1+1/(1+1/x))
>> simplify(h)
```

Na przykładzie powyżej utworzono 3 wyrażenia algebraiczne (przyjmując, że zmienne symboliczne w nich użyte zostały wcześniej zadeklarowane jako zmienne symboliczne) oraz poddano je działaniom różnych funkcji. Pierwsza z nich *expand* powoduje rozwinięcie wzoru, czyli dokonanie dostępnych w nim operacji mnożenia. Druga *collect* grupuje we wzorze podanym jako pierwszy parametr wyrażenia zawierające zmienną z drugiego parametru. Natomiast funkcja *simplify* powoduje uproszczenie wzoru i wykonanie możliwych w nim operacji na wartościach stałych.

Możliwe jest również podstawienie wartości do wyrażenia symbolicznego. W tym przypadku można wyróżnić dwie sytuacje – wyrażenie zawiera tylko jedną zmienną lub wyrażenie posiada więcej niż jedną zmienną:

```
>> f=5*a^2-6*(a-4)
>> subs(f, 2)
>> g=2*a+5*b*a-1.5*a+8*b+2.5*a-b
>> subs(g, a, 3)
```

Pierwszym parametrem funkcji *subs* jest wyrażenie algebraiczne zapisane z użyciem zmiennych symbolicznych. Jeżeli w wyrażeniu jest tylko jedna zmienną, jako drugi parametr można podać wartość jak ma być pod nią podstawiona. W sytuacji występowania większej ilości zmiennych symbolicznych najpierw podaje się zmienną, która będzie podstawiana, a dopiero jako trzeci parametr jej wartość.

W sytuacji otrzymania wielopiętrowych ułamków w wyniku przetwarzania wyrażeń symbolicznych przydatna może okazać się funkcja *pretty*, która wyświetla zapis równania w graficznej postaci:

```
>> licznik=5*a^2-6*(a-4)
>> mianownik=2*a+5*b*a-1.5*a+8*b+2.5*a-b
>> f=licznik/mianownik;
>> pretty(f)
```

Możliwe jest również wykreślenie przebiegu funkcji wykorzystującej zmienne symboliczne poprzez użycie funkcji ezplot:

```
>> f=sym(3*x^2 + 7*x + 5)
>> ezplot(f)
```

Pakiet Symbolic Toolbox realizujący wszystkie opisane powyżej operacje na wyrażeniach symbolicznych umożliwia również obliczanie pochodnych wyrażeń algebraicznych przy wykorzystaniu funkcji *diff* oraz ich całkowania – funkcja *int*.

```
>> f=sym(3*x^2 + 7*x + 5)
>> diff(f)
>> g=sin(alpha*t)+t^2-cos(alpha)
>> diff(g, alpha)
>> diff(g, t)
>> int(f)
>> int(f, -1, 1)
>> int(g, t)
```

W funkcji *diff* jako pierwszy parametr podawane jest wyrażenie podlegające różniczkowaniu. W sytuacji, gdy w wyrażeniu występuje więcej niż jedna zmienna symboliczna, jako drugi parametr można podać tą zmienną, przez którą ma być wykonane różniczkowanie cząstkowe (obydwie zmienne muszą być wcześniej zadeklarowane jako symboliczne). Podobnie funkcja *int* jako pierwszy parametr przyjmuje wyrażenie symboliczne oraz również jako drugi parametr można podać zmienną, względem której ma być wykonane całkowanie. W przypadku podania trzech parametrów program obliczy wartość całki ograniczonej przez wartości podane jako drugi i trzeci parametr.

Ponadto matlab może być wykorzystywany do obliczania prostych równań algebraicznych oraz układów równań. Służy do tego funkcja solve:

```
>> solve(3*x^2 + 7*x + 5==0)
>> f=3*a+b-7;
>> g=a-2*b
>> [a, b]=solve(f,g)
```

W powyższym przykładzie rozwiązaniu podlega następujący układ równań:

$$\begin{cases} 3a + b = 7 \\ a - 2b = 0 \end{cases}$$

4. Operacje na macierzach
   - rand(3,3) – tworzy macierz o 3 wierszach i 3 kolumnach wypełnioną losowymi liczbami z przedziału (0,1)
   - zeros(2,2) – tworzy macierz o 2 wierszach i 2 kolumnach wypełnioną wartościami „0”
   - ones(1,2) – tworzy macierz o jednym wierszu i 2 kolumnach wypełnioną wartościami „1”
   - eye(3,3) – tworzy macierz jednostkową o 3 wierszach i 3 kolumnach
   - det(A) – oblicza wyznacznik macierzy A

   Macierz transponowana jest otrzymywana w programie Matlab przez dodanie apostrofu tak jak pokazano na poniższym kodzie.

```
>> A=rand(1,3)
>> A’
```

   W macierzach można odwoływać się do poszczególnych elementów i odczytywać lub modyfikować tylko wskazane fragmenty. Przykład został zaprezentowany poniżej. W macierzy jednostkowej A zostaje zmieniona wartość z „0” na „2” w pierwszym wierszu i drugiej kolumnie. W tym samym wierszu, ale w trzeciej kolumnie zostaje wpisana wartość „3”. Natomiast w trzeciej kolumnie drugiego wiersza zostanie przypisana wartość „4”.

```
>> A=eye(3,3);
>> A(1,2)=2;
>> A(1,3)=3;
>> A(2,3)=4;
```

5. Tworzenie skryptu

   W przypadku wykonywania operacji złożonych z wielu poleceń można zapisać je za pomocą skryptu, a następnie uruchamiając go, program wykona operacje tak, jakby wszystkie komendy ze skryptu były wpisywane kolejno do okna komend. Skrypt tworzymy klikając na przycisk „New Script” w lewym górnym rogu programu (w starszych wersjach wybierając kolejno File -> New File).

   Uwaga: Ważne jest, aby skrypt był zapisany w aktualnym folderze roboczym programu Matlab i pojawił się w oknie „Current folder”. W przypadku zapisu pliku ze skryptem w innym folderze należy zgodzić się aby Matlab zmienił folder roboczy na ten, w którym zapisano plik, lub wskazać go bezpośrednio w pasku adresu znajdującym się bezpośrednio pod wstążką narzędzi programu Matlab, a nad oknami „Current Folder” oraz „Command Window”.
   Aby pokazać przydatność tworzenia kodu w skrypcie rozważmy następujący przypadek.

```
P=[sqrt(2); 0; 1];
alpha=pi/4;
R=[cos(alpha) -sin(alpha) 0;sin(alpha) cos(alpha) 0;0 0 1];
P1=R*P;
T=eye(3,3);T(1,3)=-1;T(2,3)=2;
P2=T*P1
```

Tworzymy macierz P opisującą współrzędne punktu P $(\sqrt{2},0)$ w przestrzeni 2-wymiarowej. Następnie w zmiennej alpha określamy w radianach kąt obrotu tego punku wokół początku układu współrzędnych i przygotowujemy macierz rotacji R. Do zmiennej P1 przypisujemy współrzędne punktu po jego transformacji. Przedostatnia linia to utworzenia macierzy transformacji, która następnie jest wykorzystywana do zapisu w zmiennej P2 współrzędnych ostatecznego położenia punktu.

Teraz rozważmy sytuację, w której chcemy te same operacje wykonać dla innych wartości macierzy P. Przykładowo popełniliśmy pomyłkę wpisując jej współrzędne lub były one podane poprawnie, ale dotyczą punktu będącego początkiem odcinka, a teraz chcemy obliczyć, gdzie znajduje się jego koniec. W tym celu należałoby podać nowe wartości macierzy P i wywołać wszystkie pozostałe komendy (można to zrobić klikając na nie dwukrotnie w oknie „Command History" lub naciskając odpowiednią ilość razy strzałkę do góry w „Command Window"). Lepszym rozwiązaniem jest utworzenie skryptu i zapisanie w nim powyższych poleceń. Dzięki temu później można dokonać modyfikacji tylko jednej linii, a nowe uruchomienie skryptu wykona wszystkie komendy.

Skrypt uruchamiamy naciskając klawisz F5 lub klikając na przycisk „Run" przedstawiający zielony trójkąt obrócony o 90° (w starszych wersjach przycisk „Save and run" – ikona niebieskiej strzałki skierowanej do dołu obok której jest kartka papieru).

W skrypcie można odwoływać się do wszystkich zmiennych dostępnych w workspace. Również nowo utworzona zmienna będzie dodana do przestrzeni roboczej. Program zachowuje się dokładnie tak samo jak w przypadku wpisywania tych komend w oknie „Command Window". Jedyną różnicą jest wyświetlenie w „Command History" tylko nazwy skryptu zamiast wszystkich komend w nim zapisanych.

6. Wykres dwuwymiarowy

Omawianie funkcji rysujących wykres rozpocznijmy od poniższego przykładu, który można uruchomić jako skrypt. Znaki „%" oznaczają w kodzie programu Matlab początek komentarza, który trwa do końca linii.

Pierwsza linia kodu odpowiedzialna jest za utworzenie macierzy *x* składającej się z jednego wiersza. W pierwszej kolumnie przyjmuje ona wartość 0, w drugiej – 0.1, w trzeciej – 0.2 i tak dalej, dopóki wartość ta nie będzie większa od 10. Macierz taką otrzymujemy poprzez zastosowanie dwukropków, a symbolicznie można ją przedstawić w następującej postaci:

[wartość początkowa]:*[skok]*:*[maksymalna wartość końcowa]*. Przy czym środkowy parametr *skok* jest opcjonalny. Można go pominąć, a wtedy przyjmuje on domyślną wartość równą 1.

```
x=0:0.1:10;
f=2*x.^2-3*x+5;
plot(x,f)
figure()
plot(x,f,'*r')
```

W drugiej linii następuje utworzenie macierzy, która dla każdego argumentu z macierzy *x* będzie posiadała wartość funkcji $f(x)=2x^2-3x+5$. Operator „.^" oznacza, że operacja potęgowania ma być wykonywana na każdym elemencie macierzy z osobna. W przypadku braku kropki program próbowałby wykonać mnożenie macierz razy macierz, co spowodowałoby powstanie błędu z powodu niezgodności rozmiarów. W analogiczny sposób kropka może być wykonana do operacji mnożenia oraz dzielenia. Jednak w ich przypadku operacja z wartością skalarną da taki sam wynik jak bez użycia kropki. Inaczej sytuacja wygląda gdy rozmiary macierzy na pozwalają na ich mnożenie, ale jednocześnie obydwie macierze mają dokładnie takie same rozmiary. Wtedy użycie operatora z kropką spowoduje, że dane działanie będzie wykonywane na odpowiadających sobie elementach macierzy. Na przykład dla macierzy *x* i y o wymiarach 2x5 operacja *x.*y* spowoduje, że mnożone będą: element z pierwszego wiersza i pierwszej kolumny macierzy *x* oraz element z pierwszego wiersza i pierwszej kolumny macierzy *y*, element z pierwszego wiersza i drugiej kolumny macierzy *x* oraz element z pierwszego wiersza i drugiej kolumny macierzy *y*, …, element z drugiego wiersza i piątej kolumny macierzy *x* oraz element z drugiego wiersza i piątej kolumny macierzy *y*.

W trzeciej linii funkcja plot rysuje wykres łącząc kolejne punkty liniami. Punkty są tworzone przez wybranie kolejnych elementów z macierzy tj. pierwszy element macierzy *x* jako współrzędna X pierwszego punktu oraz pierwszy element macierzy *f* jako współrzędna Y pierwszego punktu itd. Wykres otworzy się w nowym oknie, któremu zostanie nadany numer 1, widoczny na pasku nazwy. Funkcja *figure()* tworzy nowe okno i uaktywnia je to znaczy, że nowy wykres będzie narysowany w nim.
Druga funkcja plot posiada trzeci parametr będący ciągiem znaków. Odpowiada on za formatowanie stylu wykresu. Parametr ten może zawierać symbole odpowiedzialne za styl linii lub styl markera. Marker oznacza, że punkty nie będą łączone, lecz będą oznaczone na wykresie przez pojedyncze znaki – markery. Podanie znaków stylu linii powoduje, że punkty z wcześniej wpisanych macierzy będą łączone linią o danym stylu. Natomiast wpisanie symbolów oznaczających marker spowoduje wyświetlenie tylko tych punktów. Inną możliwością jest podanie symboli stylu linii oraz markera. Wtedy punkty oznaczone określonym markerem będą łączone linią o pożądanym stylu. Przykładowo ciąg znaków *'-.*b'* spowoduje wyświetlenie punktów w postaci gwiazdki oraz połączenie ich linią przerywaną składającą się na przemian z kreski oraz kropki. Ponadto można podać literę odpowiedzialną za kolor linii. W powyższym przypadku litera *b* oznacza kolor niebieski.

Aby zmodyfikować zakresy wartości oraz argumentów funkcji wyświetlanych przez funkcję *plot*, należy posłużyć się funkcją *axis*:

```
>> k = axis
>> k(1) = -1;
>> k(2) = 11;
>> axis(k)
```

W pierwszej linii powyższego kodu następuje odczytanie skrajnych wartości na osi odciętych oraz rzędnych w aktywnym oknie funkcji *plot*. Wynikiem tej operacji jest macierz o wymiarach 1x4. Pierwszy jej element opisuje minimalna wartość na osi odciętych, drugi maksymalną wartość na tej samej osi. Analogicznie trzeci i czwarty element to kolejno wartość najmniejsza oraz największa na osi rzędnych. W drugiej i trzeciej linii kodu zmianie ulegają parametry odpowiedzialne za oś odciętych. Następnie czwarta linia kodu powoduje zastosowania formatowania wykresu w aktualnie otwartym oknie funkcji *plot*.

Style linii, markerów oraz symbole kolorów wymienione poniżej pochodzą z dokumentacji programu Matlab.

Style linii:

- - (linia ciągła)
- -- (linia przerywana składająca się z kresek)
- : (linia przerywana składająca się z kropek)
- -. (linia przerywana składająca się na przemian z kreski oraz kropki)

Style markerów:

- o
- +
- *
- .
- x
- s (kwadrat)
- d (diament, romb)
- ^
- v
- >
- <
- p (pentagram)
- h (hexagram)

Kolory:

- y (yellow)

- m (magenta)
- c (cyan)
- r (red)
- g (green)
- b (blue)
- w (white)
- k (black)

Zadanie 1

Zmodyfikuj skrypt z rozdziału 6 instrukcji, tak aby punkt był wyświetlony w postaci kropki, dziedzina funkcji znajdowała się w zakresie <-10,10>, a punkty były obliczane i wyświetlane co 0.01 na osi X.

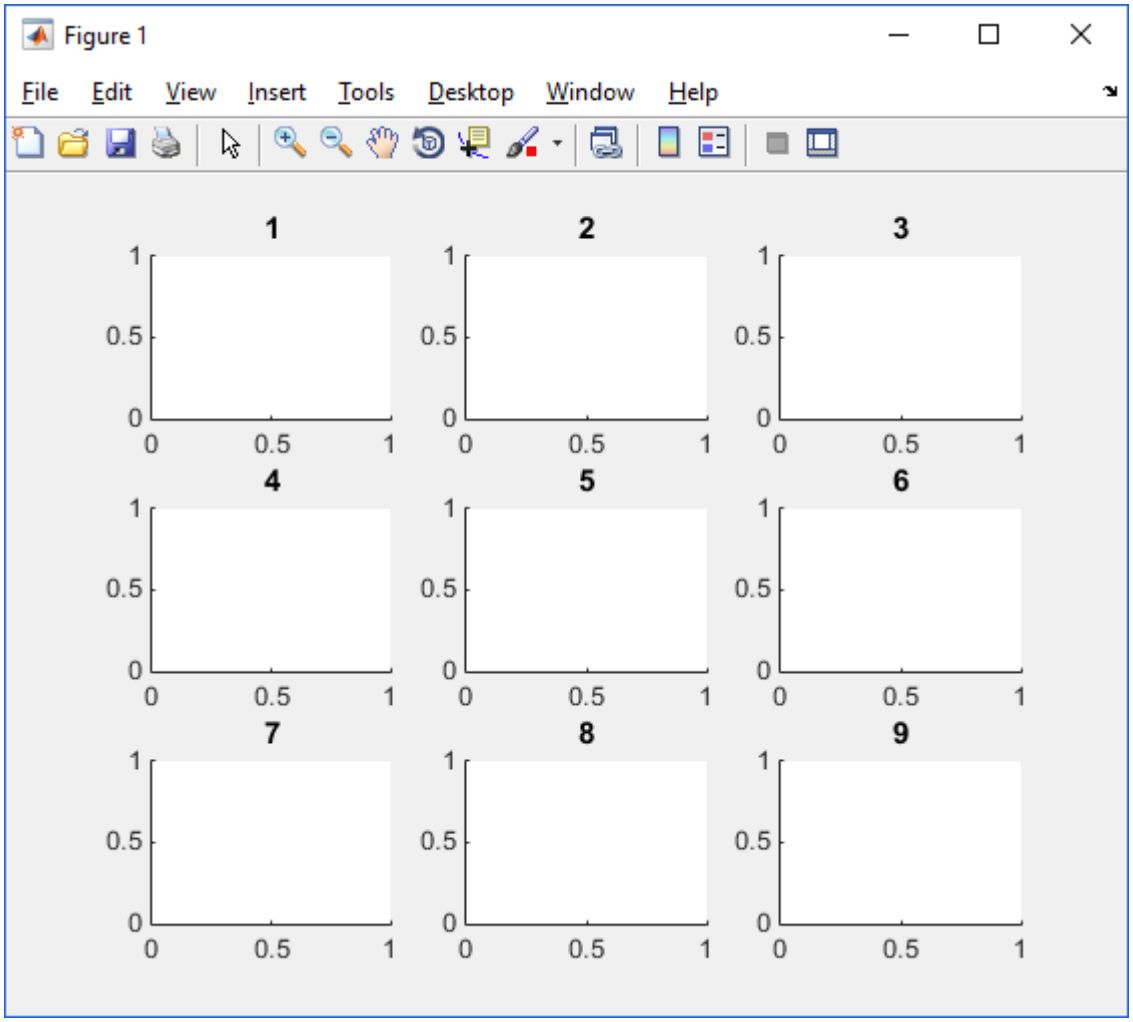
Aby przeanalizować wykonywanie kolejnego kodu najlepiej wklejać kolejno komendy w Command Window i po każdej obserwować zmiany zachodzące na wykresach.

```
>> x=0:0.1:10;
>> f2=2*x.^2+3*x-5;
>> plot(x,f,'r',x,f2,'b')
>> figure(2)
>> plot(x,f,'r-.')
>> figure(1)
>> plot(x,f2,'b-.')
>> figure(3)
>> subplot(1,2,1)
>> plot(x,f)
>> grid
>> subplot(1,2,2)
>> plot(x,f2)
>> close(1,2)
```

Nowością w powyższym kodzie może być funkcja *plot()*, która przyjmuje więcej niż 3 parametry. Zabieg taki stosuje się, aby na jednym wykresie zastosować więcej niż jedną serię danych. W parametrach należy podawać kolejno macierz współrzędnych X z pierwszej serii, macierz współrzędnych Y z pierwszej serii, styl wykresu dla pierwszej serii danych (opcjonalnie), macierz współrzędnych X z drugiej serii, macierz współrzędnych Y z drugiej serii, styl wykresu dla drugiej serii danych (opcjonalnie), itd. Należy przy tym pamiętać, aby rozmiar macierzy w każdej serii był jednakowy. Natomiast możliwa jest różna liczba punktów w każdej z serii danych. Przykładowo pierwsza seria składa się z tysiąca punktów, natomiast druga z dziesięciu lub tylko jednego (wtedy w innym kolorze zostanie zaznaczony jeden punkt).

Podanie w funkcji *figure()* jako parametr liczby, spowoduje otwarcie okna o tym numerze i uaktywnienie go. Jeżeli okno o podanym numerze już jest otwarte i znajduje się w nim wykres, to kolejne wywołanie funkcji *plot()* spowoduje zamianę występującego w nim wykresu. Taka sytuacja została zaprezentowana w kodzie powyżej. Wykres narysowany w trzeciej linii zostaje otwarty w oknie o numerze 1. Następnie w oknie 2 zostaje narysowany inny wykres. Jednak w linii 6 następuje powrót do pierwszego okna i wykres narysowany w następnej linii zastąpi ten znajdujący się w nim wcześniej. Z kolei funkcja *close()* umożliwia zamknięcie okien o podanych numerach.

Funkcja *subplot()* umożliwia narysowanie wielu wykresów w jednym oknie. Można powiedzieć, że takie okno zawiera macierz wykresów, ponieważ wykresy znajdują się w określonej liczbie wierszy oraz kolumn obok siebie, tak jak przedstawiono to na rysunku poniżej.



Pierwszy parametr funkcji *subplot()* określa w ilu wierszach będą wyświetlane wykresy, drugi dotyczy liczby kolumn. Natomiast trzeci parametr to numer aktywowanego wykresu. Numeracja ta zmienia się w pierwszej kolejności wierszami, a następnie kolumnami - od lewej do prawej oraz z góry do dołu - tak jak przedstawiono na rysunku powyżej. Podobnie jak w przypadku funkcji *figure()* również w przypadku *subplot()* można aktywować pole z już wykreślonym wykresem i dokonać jego modyfikacji lub zmiany.

W wyżej zaprezentowanym kodzie występuje również funkcja *grid*, która włącza wyświetlanie siatki na aktualnym wykresie.

Istnieje również możliwość dodanie tytułu wykresu. Dokonuje się to podając jako parametr funkcji *title()* odpowiedni ciąg znaków. Podobnie działają funkcje *xlabel()* oraz *ylabel()* dodające opisy odpowiednio osi X oraz Y. Istnieje również możliwość dodania do wykresu legendy. Służy do tego funkcja *legend()* przyjmująca jako parametry taką ilość ciągów znaków, która odpowiada ilości narysowanych wykresów. Aby nastąpiło poprawne dopasowanie legendy do danych na wykresie, wpisane ciągi znaków powinny być wymienione w takiej samej kolejności w jakiej pyły wypisane odpowiadające im serie danych w funkcji *plot()*. Zastosowanie tych funkcji prezentuje poniższy skrypt.

```
plot(x,f,'r',x,f2,'b')
title('Tytuł wykresu')
xlabel('Opis osi x')
ylabel('Opis osi y')
legend('Wykres funkcji f', 'Wykres funkcji f2')
```

Matlab posiada wiele funkcji odpowiedzialnych za rysowanie różnych wykresów, jednak w ramach niniejszego przedmiotu można ograniczyć ich znajomość do funkcji *plot()* oraz *plot3()*. Ta druga zachowuje się analogicznie jak omawiana wcześniej *plot()* z tą różnicą, że w zakresie danych przyjmuje 3 macierze, które są odpowiedzialne za współrzędne w przestrzeni 3D, a następnie rysuje wykres przestrzenny.

7. Funkcja warunkowa *if*

W programie Matlab funkcja warunkowa nieznacznie różni się w budowie w porównaniu do języka C. Warunek w funkcji *if* nie musi znajdować się w nawiasie. Również inne komendy wykonywane gdy warunek będzie spełniony lub przeciwnie nie zapisujemy w żadnym bloku. Będą one traktowane jako warunkowe aż do pojawienia się w skrypcie słowa kluczowego *elseif*, *else* lub *end*. Z tego powodu każda instrukcja warunkowa musi zakończyć się komendą *end*. Należy również zwrócić uwagę na słowo kluczowe *elseif*, w którym nie ma spacji. Dodanie spacji przed dwoma ostatnimi literami spowoduje, że Matlab będzie ten kod interpretował jako pojawienie się kolejnego zagnieżdżonego if'a, dla którego również będzie oczekiwał zakończenia słowem *end*. Przykład poprawnego wykorzystania funkcji warunkowej został zaprezentowany poniżej.

```
a=input('Podaj wartość a: ')
b=input('Podaj wartość b: ');
if a>0
    disp('Podano a większe od 0')
elseif b~=0
    disp('Podano b różne od 0')
else
    disp('Podano a mniejsze lub równe 0 oraz b równe 0')
end
```

W powyższym przykładzie funkcja *input()* wypisuje w oknie Command Window tekst podany jako parametr, a następnie oczekuje na podanie przez użytkownika wartości liczbowej, która zostanie przez nią zwrócona. Natomiast funkcja *disp()* ogranicza swoje działanie wyłącznie do wyświetlenia komunikatu o podanej w parametrze treści.

8. Pętla for

W przeciwieństwie do funkcji warunkowych, które w Matlabie wykazują duże podobieństwo do najpopularniejszych języków programowania, zapis pętli for zdecydowanie się różni. Zamiast podawania trzech wyrażeń w nawiasie po słowie kluczowym *for* w programie matlab musimy wykonać przypisanie do indeksowanej zmiennej zakres wartości. Zakres ten określany jest przy pomocy operatora dwukropka „:", który został szczegółowo omówiony w niniejszym wykresie przy okazji pierwszego skryptu w punkcie 6.
Przykład użycia pętli for zagnieżdżonych w sobie został przedstawiony na poniższym przykładzie.

```
x=[];y=[];a=1;
for i=1:5
    for j=-1:8
        x(a)=i;
        y(a)=j;
        a=a+1;
    end
end
plot(x,y,'.')
```

Skrypt ten ma za zadanie narysować kropki w miejscach przecięcia się wartości całkowitych osi X oraz Y na obszarze ograniczonym nierównościami:

- $1 \leq x \leq 5$
- $-1 \leq y \leq 8$

W przykładzie tym macierze *x* oraz *y* zostały zadeklarowane jako macierze puste przed pętlami for. Również zmiennej *a* została przypisana wartość początkowa 1. Zmienna ta będzie wskazywała kolejny element w macierzach *x* i *y*, w którym będzie zapisywana wartość określona przez zmienne zmieniane w pętlach for.

Uwaga: W powyższym przykładzie macierze są używane jako tablice znane z języków wysokiego poziomu. Należy przy tym pamiętać, że numeracja elementów w macierzy programu Matlab rozpoczyna się od wartości 1, a nie jak w wielu innych językach od 0.

Zadanie 2

Zmodyfikuj powyższy kod tak, aby na wykresie pojawił się prostokąt ograniczony równaniami: $1 \leq x \leq 10$, $-1 \leq y \leq 1$. Podpowiedź: aby uzyskać kształt prostokąta, należy zachowując tą samą wielkość wykresu zwiększyć ilość wyświetlanych w nim punktów – zagęścić je.

Zadanie 3

Zmodyfikuj powyższy kod tak, aby narysować koło o środku w punkcie (2,2) oraz promieniu r=2.

9. Funkcje

W programie Matlab istnieje również możliwość tworzenia własnych funkcji. Tworzone są podobnie jak skrypty i podobnie są zapisywane z rozszerzeniem *.m. Kolejne podobieństwo dotyczy wymogu umieszczenia pliku z funkcją w folderze roboczym, aby można było z niej korzystać.

Funkcje są tworzone przy pomocy słowa kluczowego *function*. Po nim znajduje się nazwa macierzy zwracanej, później znak równości, nazwa funkcji oraz w nawiasach okrągłych wymieniane są zmienne wejściowe – parametry. Schematycznie można zapisać to w następujący sposób:

```
function macierz_wyjściowa = nazwa_funkcji (parametry)
```

Ponieważ w Matlabie nie deklaruje się typów zmiennych więc również wypisując parametry nie wypisuje się typów, a jedynie nazwę zmiennej. Na uwagę zasługuje fakt, że w Matlabie nie używa się słowa kluczowego *return* w celu określenia wartości zwracanej przez funkcję. Zwrócona zawsze będzie wartość zmiennej zadeklarowanej jako wejściowa, którą ta będzie posiadała na końcu wykonywania funkcji.

Uwaga: Plik musi posiadać taką samą nazwę jak nazwa funkcji. W przeciwnym przypadku Matlab nie rozpozna tej funkcji i nie będzie możliwe jej wykorzystywanie w innych funkcjach lub skryptach. Powoduje to, że musimy utworzyć tyle plików ile stworzymy funkcji do uruchamiania w programie. Natomiast w jednym pliku może znajdować się większa ilość funkcji. Będą one dostępne w obrębie tego pliku. Jednak poza nim (w skryptach lub w oknie Command Window) dostępna będzie tylko ta o nazwie identycznej jak nazwa pliku.
W przypadku tworzenia większej ilości funkcji w jednym pliku, każda z nich musi być zakończona słowem kluczowym *end*. W przypadku zamieszczenia tylko jednej funkcji można to słowo pominąć.

Powyższy opis powinien być bardziej zrozumiały po napisaniu poniższej funkcji oraz zapisaniu jej w pliku *suma.m* znajdującym się w folderze roboczym.

```
function y = suma(a,b)
    y=a+b;
```

Powyższa funkcja zwraca sumę dwóch zmiennych podanych jako jej parametry. Można ją wywołać wpisując w oknie Command Window poniższe polecenie.

```
c=suma(2,3)
```

Zadanie 4

Napisz funkcję, która przyjmuje parametry funkcji kwadratowej (a, b, c) oraz zwraca macierz zawierającą jej miejsca zerowe, informacje o ilości miejsc zerowych oraz informację, czy miejsca zerowe są liczbami rzeczywistymi, czy zespolonymi. Ponadto powinien zostać narysowany wykres fragmentu funkcji kwadratowej obejmujący jej miejsca zerowe (jeżeli występują) lub wierzchołek.

Funkcja powinna rozpatrywać przypadek podania przez użytkownika funkcji liniowej (a=0) i wyznaczenia dla niej miejsca zerowego (jeżeli istnieje).

Zadanie 5

Napisz funkcję szyfrującą oraz deszyfrującą dla szyfru cezara. Szyfr ten działa na zasadzie zastępowania litery inną literą znajdującą się w alfabecie dalej o określoną ilość X miejsc.

Przykładowo dla X=2, litera „a” zostanie zastąpiona przez „c”, „b” przez „d” itd. W przypadku zakończenia alfabetu przesunięcie liczone jest od początku. Dla wspomnianego przypadku litera „y” zostanie zastąpiona przez „a”, natomiast litera „z” przez „b”.

Alfabet w funkcjach powinien zawierać spację, cyfry od 0-9 oraz duże i małe litery alfabetu łacińskiego. Funkcje jako parametr powinny przyjmować tekst do szyfrowania/zaszyfrowany oraz wartość przesunięcia, natomiast zwracać powinny tekst zaszyfrowany/odszyfrowany.

Funkcje powinny zawierać macierz ze zdefiniowanym alfabetem znaków podlegających szyfrowaniu i na jej podstawie powinno następować szyfrowanie. Zabrania się oparcia sposobu działania funkcji na instrukcji switch oraz wielokrotnym wykorzystaniu instrukcji if.

W sytuacji podania znaku nienależącego do alfabetu funkcja powinna zwrócić błąd.

W wykonaniu zadania pomocna będzie funkcja find(). Więcej informacji o sposobie jej działania można uzyskać wpisując w Command Window Matlaba funkcje:
>> man find
lub
>> help find

Przykład użycia funkcji find:

```
alfabet = ['a' 'b' 'c';'d' 'e' 'f'];
[i, j] = find(alfabet == 'c');
```

Zadanie 6

Zmodyfikuj kod z zadania 5 tak, aby funkcja dokonywała szyfrowania kluczem. Funkcje powinny przyjmować jako parametry tekst do zaszyfrowania/odszyfrowania oraz ciąg znaków będących kluczem szyfru.

Szyfrowanie kluczem różni się od szyfru cezara zmienną wartością przesunięcia uzależnioną od klucza. Kolejne znaki szyfru są przesuwane o tyle ile wynosi pozycja kolejnych znaków klucza. Przykładowo dla alfabetu zawierającego wyłącznie znaki a-z oraz dla klucza o wartości „klucz" pierwszy znak szyfrowanego ciągu zostanie przesunięty o 11 (pozycja litery k), drugi o 12 (pozycja litery l) itd. W przypadku zakończenia ilości znaków w kluczu przesunięcie ponownie zaczyna się od jego pierwszego znaku.

Przykładowo tekst „abcdefghij" dla kucza „klucz" zostanie zaszyfrowany jako „lnxgeqsdcj" (dla alfabetu złożonego wyłącznie z małych liter alfabetu łacińskiego).

Należy rozpatrzeć przypadek w którym szyfrowany tekst będzie krótszy od klucza.

Sprawozdanie

- Na ocenę:
  - 3.0 poprawnie wykonane zadania 1-4 wraz z opisem,
  - 4.0 poprawnie wykonane zadania 1-5 wraz z opisem,
  - 5.0 poprawnie wykonane zadania 1-6 wraz z opisem.
- Sprawozdanie należy przesłać w formie elektronicznej na adres dozog@kia.prz.edu.pl jako załącznik do wiadomości e-mail. Temat wiadomości „PR Sprawozdanie I grupa LX", gdzie X jest numerem grupy laboratoryjnej osób wykonujących sprawozdanie.
- Sprawozdanie powinno być wykonane na bazie formatki zamieszczonej na stronie z instrukcją.
- Jako poprawny sposób opisu zadania w sprawozdaniu uważa się zamieszczenie:
  - Kodu źródłowego,
  - Wyników przedstawiających efekt działania skryptu (np. zrzuty ekranu),
  - Szczegółowego opisu kodu źródłowego z wyjaśnieniem w jaki sposób zamieszczone w nim instrukcje rozwiązują problem. W tym celu można podeprzeć się komentarzami w kodzie lub podzieleniem kodu za pomocą komentarzy na kolejno numerowane bloki instrukcji, a następnie wykorzystaniu numerów tych bloków w opisie.
- Czas nadsyłania prac: do godziny rozpoczęcia czwartych zajęć laboratoryjnych z przedmiotu „Podstawy robotyki" w semestrze (około 2 tygodnie)
- Opóźnienie w wysyłaniu sprawozdania o każdy rozpoczęty tydzień powoduje obniżenie oceny o 0,5 stopnia.