Wprowadzenie do pakietu MathWorks MatLAB

Pakiet MathWorks MatLAB jest środowiskiem obliczeniowym z wbudowanym własnym językiem wysokiego poziomu. Nazwa pochodzi ze zbitki dwóch angielskich słów: Matrix (macierz) i Laboratory. Zbudowany jest on z jądra (MatLAB) oraz bibliotek dodatkowych (tzw. toolbox-ów), które poszerzają zakres jego stosowania.

1 Typy i formaty danych

1.1 Typy zmiennych

Jak każdy język programowania, MatLAB zawiera wbudowane podstawowe typy danych (sposoby w jakich zmienna jest przechowywana w pamięci). Informacje o nich uzyskuje się poleceniem: `help datatypes`, Do podstawowych typów danych zaliczamy:

- double - zmienna podwójnej precyzji (każda nowa zmienna jest domyślnie typu double, o ile nie ma innych przesłanek)
- single - pojedyńcza precyzja
- logical - zmienna logiczna - przyjmująca wartości `true` lub `false`
- uint8,uint16,uint32,uint64 - 8,16,32 i 64- bitowa liczba całkowita bez znaku.
- int8,int16,int32,int64 - 8,16,32 i 64- bitowa liczba całkowita ze znakiem.
- char - zmienna znakowa
- cell - zmienna komórkowa
- struct - struktura

1.2 Struktury

Struktury służą do grupowania zmiennych pod jednym typem. Definiuje się je w następujący sposób:
nazwa=struct('pole1', wartość1, 'pole2', wartość2, ...);
np. `Punkt=struct('x', 0.0, 'y', 0.0, 'z', 0.0);` Deklarowanie zmiennych będących strukturami i odwołanie się i do pól struktury następuje w sposób następujący:

```
P=Punkt;
P.x=1.0; P.y=2.0; P.z=P.x*P.y;
```

1.3 Formaty wyświetlania zmiennych

Zmienne mogą być wyświetlane na kilka sposobów. Informacje o nich uzyskuje się poleceniem `help format`. Do podstawowych formatów wyświetlania danych zaliczamy:

- short - 5 cyfr, stałoprzecinkowe
- short e - 5 cyfr, zmiennoprzecinkowa
- short g - 5 cyfr, stało- lub zmiennoprzecinkowa
- long, long e, long g - 15 cyfr, reprezentacja analogicznie do short
- rat - format ułamkowy
- hex - styl heksadecymalny
- bank - tryb walutowy

2 Znaki i nazwy specjalne

Informacje o znakach i operatorach specjalnych uzyskuje się poprzez wpisanie w konsoli polecenia `help ops`. Natomiast informacje o zmiennych i stałych specjalnych (np. π) uzyskuje się poleceniem `help elmat`.

Do najważniejszych stałych wbudowanych i zmiennych zaliczamy:

- `ans` - wartość ostatniego wyrażenia
- `eps` - dokładność obliczeń zmiennoprzecinkowych
- `realmax` - największa dostępna liczba zmiennoprzecinkowa
- `realmin` - najmniejsza dostępna liczba zmiennoprzecinkowa
- `pi` - 3.1415926535897....
- `i` oraz `j` - jednostka urojona
- `inf` - nieskończoność :)
- `NaN` - "Not-a-Number", np. wynik dzielenia zero przez zero

3 Podstawowe funkcje matematyczne

MatLAB posiada wbudowane funkcje matematyczne. Najważniejsze funkcje zostały zamieszczone w fig.1

Informacje na temat podstawowych funkcji matematycznych uzyskuje się poleceniem `help elfun`. Zestaw najważniejszych poleceń zawiera Fig.1. Informacje o bardziej zaawansowanych funkcjach uzyskuje się poleceniem `help specfun`.

4 Polecenia i instrukcje

Do najważniejszych poleceń możliwych do wpisania w wierszu poleceń służą:

- `clear a; clear all;` - usunięcie zmniennej a, usunięcie wszystkich zmiennych.
- `clc;` - czyszczenie konsoli
- `close all;` - zamknięcie wszystkich okien
- `figure` - otwarcie nowej figury
- `hold on` - przytrzymanie "wykresu" i umożliwienie dodania kolejnego wykresu.
- `hold off` - uwolnienie ekranu. Każde następne wyświetlenie "wyczyszcza" ekran
- `ans` - wyświetlenie ostatniego wyniku
- `exit` - zamknięcie programu
- `a=load('nazwa pliku')` - wczytanie zmiennych
- `save nazwa_pliku zmienne -ascii` - zapis zmiennych w pliku tekstowym

4.1 Instrukcje warunkowe

Instrukcje warunkowe służą do sterowania obliczeniami w zależności od stanu (warunków). Instrukcje te zawsze kończą się poleceniem `end`. Do najważniejszych instrukcji zaliczamy:

4.2 if - else

Instrukcja `if` służy do wykonywania pewnych instrukcji, jeżeli spełnione są dane warunki. Składnia wygląda następująco:

Nazwa	Opis funkcji
abs	wartość bezwzględna; moduł liczby zespolonej; wektor wartości znaków łańcucha wg. kodu ASCII
acos, acosh	arcus cosinus, arcus cosinus hiperboliczny
acot, acoth	arcus cotangens, arcus cotangens hiperboliczny
acsc, acsch	arcus cosecans, arcus cosecans hiperboliczny
angle	argument liczby zespolonej, w radianach [-π, +π]
asec, asech	arcus secans, arcus secans hiperboliczny
asin, asinh	arcus sinus, arcus sinus hiperboliczny
atan, atanh	arcus tangens, arcus tangens hiperboliczny
atan2	arcus tangens dwuargumentowy, φ=atan2(imag(z), real(z)); $\varphi \in$ [-π, +π];
ceil	zaokrąglenie w kierunku +∞
conj	liczba sprzężona
cos, cosh	cosinus, cosinus hiperboliczny
cot, coth	cotangens, cotangens hiperboliczny
csc, csch	cosecans, cosecans hiperboliczny
exp	funkcja wykładnicza (ekspotencjalna)
fix	zaokrąglenie w kierunku zera
floor	zaokrąglenie w kierunku -∞
log	logarytm naturalny (przy podstawie e)
log2	logarytm przy podstawie 2;
log10	logarytm dziesiętny (przy podstawie 10)
pow2	potęgowanie przy podstawie 2; $pow2(f,x) = f.*2^x$
real, imag	część rzeczywista i urojona liczby zespolonej
rem	reszta z dzielenia
round	zaokrąglenie do najbliższej liczby całkowitej
sec, sech	secans, secans hiperboliczny
sign	znak
sin, sinh	sinus, sinus hiperboliczny
sqrt	pierwiastek kwadratowy
tan, tanh	tangens, tangens hiperboliczny

Rysunek 1: Podstawowe funkcje matematyczne w MATLABie (źródło:Mrozek & Mrozek, 2002)

```
if (warunek)
   instrukcje1;
else
   instrukcje2;
end
```

Program wykonuje `instrukcje1` wtedy i tylko wtedy, gdy `warunek` ma wartość logiczną "true". W przeciwnym wypadku wykonuje `instrukcje2`. Instrukcja `else` nie jest wymagana.

Przykładowa instrukcja wygląda następująco:

```
if (a<0)
   'liczba ujemna'
   a=abs(a);
else 'zero lub liczba dodatnia'
end
```

4.3 Pętla for

Pętla `for` wykonuje dany blok poleceń określoną ilość razy. Składnia polecenia wygląda następująco:

```
for k = początek : krok : koniec
   polecenie
end
```

Pętle `for` działają znacznie wolniej niż zastosowanie operacji kropkowych czy dwukropkowych. Dlatego należy stosować te pętle tylko w przypadkach niezbędnych lub "karkołomnych". Poniższe dwie instrukcje są równoważne:

```
for k=1:N A(k)=sin(k); end
A=sin(1:N);
```

przy czym na ogół druga instrukcja wykonuje się szybciej.

4.4 Pętla while

Pętla `while (warunek) end` jest wykonywana dopóki warunek ma wartość `true`. W pętli tej najpierw sprawdzana jest wartość logiczna warunku i w razie spełnienia wykonywane instrukcje.

Przykładowe polecenie wygląda następująco:

```
k=10;
while(k>0)
   sin(k)
   k=k-1;
end
```

4.5 switch

Instrukcja `switch()` jest instrukcją wyboru wielokrotnego. Jej składnia wygląda następująco:

```
switch wyrażenie
   case opcja polecenia
   case {opcja1, opcja2} polecenia
   otherwise polecenia
end
```

Instrukcja ta może służyć np. do zliczeń wystąpień znaków. Zastępuje ona wielokrotnie zagnieżdzone instrukcje `if - else`.

W przeciwieństwie do instrukcji `switch` w języku C/C++, MatLAB przestaje sprawdzać kolejne warunki po napotkaniu pierwszej zgodności.

4.6 makra @

W MatLABie istnieje możliwość pisania krótkich, własnych funkcji (ang. *function handle*). Składnia wygląda następująco:

```
nazwa_funkcji=@(argumenty) polecenie;
```

Na przykład makro liczące pole koła byłoby zdefiniowane następująco:

```
pole=@(r) pi*r.*r;
```

W tym momencie w workspace pojawia się nowa zmienna (makro), którą wywołuje się poleceniem: `pole(5)`.

5 Macierze i wektory

UWAGA: Pierwszy element macierzy ma index 1, a nie, jak w C, zero.

Macierze i wektory można generować na kilka sposobów. Do najczęściej używanych należą:

- notacja dwukropkowa (:) np. `a=1:2:7` da wektor [1 3 5 7]
- użycie funkcji specjalnych (zeros(), ones(), eyes(), rand(), randn()) `a=zeros(5,6)` - generuje tablicę złożoną z samych zer o 5 wersach i 6 kolumnach.
- wpisanie przy użyciu [ ], np. `a=[1 2 3; 4 5 6; 7 8 9]`.
- wczytanie z pliku (funkcja `load`) `a= load('a.txt');`

Specjalnymi operatorami stosowanymi do obliczeń macierzowych są:

- (apostrof): służy do transpozycji macierzy
- (kropka): służy do traktowania macierzy jako tablica i wykonywania operacji komórka po komórce. Warunkiem działania jest identyczność rozmiarów zmiennych na których wykonywana jest dana operacja. Przykładem takiej operacji, gdzie operator kropkowy ma znaczenie, jest mnożenie. `a*b` oznacza mnożenie macierzy, natomiast `a.*b` wymnaża poszczególne elementy macierzy przez siebie, np.:

$$\mathbf{a} = \begin{pmatrix} 2 & 2 \\ 3 & 3 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

to:

$$\mathbf{a} * \mathbf{b} = \begin{pmatrix} 8 & 12 \\ 12 & 18 \end{pmatrix} \quad \mathbf{a}. * \mathbf{b} = \begin{pmatrix} 2 & 4 \\ 9 & 12 \end{pmatrix} \quad \mathbf{a}' = \begin{pmatrix} 2 & 3 \\ 2 & 3 \end{pmatrix}$$

- (\ i /): lewo i prawostronne dzielenie. `A\B` oznacza rozwiązanie równania Ax=B

5.1 Używanie notacji dwukropkowej

Notacja dwukropkowa służy do tworzenia wektorów, wyciągania informacji o poszczególnych fragmentach macierzy, np. jeżeli przez **A** oznaczymy macierz, to zapis:

- `A(:,2)` oznacza drugą kolumnę macierzy **A**
- `A(3,:)` oznacza trzeci wiersz macierzy **A**
- `A(:)` oznacza obliczenia dotyczące całej tablicy. Ma to zastosowanie np. w przypadku funkcji `min()`. Instrukcja `min(A)` zwraca wektor zawierający minima z każdej kolumny. Minimum z całej tablicy 2D można uzyskać na dwa sposoby: `min(min(A))` lub `min(A(:))`
- `A(:,[1,3])` oznacza wybranie 1 i 3 kolumny
- `A=1:5` powoduje utworzenie wektora poziomego zawierającego kolejne liczby całkowite [1 2 3 4 5]

6 Praca z m-plikami

Istnieją 2 rodzaje m-plików:

funkcyjne: posiadające w pierwszej linijce definicję funkcji:
function [parametry wyjścia] = nazwa_funkcji (parametry wejścia).

Działają na zmiennych wewnętrznych. Odwoływanie się do zmiennych zewnętrznych następuje poprzez polecenie `global`. Istnieją dwa polecenia, istotne z punktu widzenia plików funkcyjnych: `nargin` i `nargout`. Zwracają one odpowiednio ilość argumentów przesłanych do funkcji i ilość argumentów zwracających. Dzięki temu możliwe jest "przeciążenie" funkcji, definiowanie wartości domyślnych funkcji. Przykładem przeciążonej funkcji jest polecenie `svd`. Funkcja to może przyjmować albo tylko tablicę albo tablicę i próg tolerancji wartości własnych. Funkcja ta może zwracać albo wartości własne albo 3 macierze będące wynikiem dekompozycji.

skryptowe: zawierające polecenie linia po linii i działają na zmiennych dostępnych w workspace.

W MatLABie od wersji 7.5 pojawiła się możliwość dzielenie m-pliku na sekcje i uruchamianie tylko poszczególnych sekcji, a nie całego pliku. Plik dzieli się na sekcje poleceniem `%%`, a konkretną sekcję uruchamia się poprzez wciśnięcie `ctrl+enter`.

7 Wykresy i grafika

MATLAB posiada szereg funkcji służących do wyświetlania danych. Najważniejsze przedstawiono w Fig.2. Opcje wyświetlania linii (markerów) przedstawiono w Fig.3. Do wyświetlania kilku wykresów na jednym służy polecenie `subplot`. Np. wpisanie polecenia `loglog(x,y,'*k')`, gdzie x, y wektory wyświetli wykres bilogarytmiczny z czarnymi gwiazdkami.

Nazwa	Opis funkcji	Nazwa	Opis funkcji
`plot`	skala liniowa obu osi	`semilogx`	skala logarytmiczna x
`loglog`	skale logarytmiczne osi	`semilogy`	skala logarytmiczna y
`bar,barh`	słupkowy (pion, poziom)	`bar3,bar3h`	słupkowy 3-wymiarowy
`hist`	histogram	`stairs`	wykres schodkowy
`rose`	histogram kołowy	`compass`	wykres strzałkowy I
`polar`	wykres kołowy	`feather`	wykres strzałkowy II
`quiver`	wykres linii pola	`errorbar`	wykres błędów
`fill`	wypełnia obszary zamkn.	`fplot,ezplot`	wykres funkcji ciągłej
`area`	wypełnia pole wykresu	`plotmatrix`	wykresy punktowe
`stem`	wykres dyskretny	`pareto`	wykres Pareto
`comet`	wykres z warkoczem	`pie, pie3`	okrągły tort
`scatter`	wykres punktowy	`ribbon`	linie jak tasiemka (3-D)

Rysunek 2: Podstawowe wykresy w MATLABie (źródło:Mrozek & Mrozek, 2002)

Do opisu wykresu służą następujące polecenia:

- `title('nazwa wykresu');` - Nadanie nazwy wykresu
- `xlabel('nazwa osi OX');` - Opisanie osi X
- `ylabel('nazwa osi OY');` - Opisanie osi Y
- `axis(x_min, x_max, y_min, y_max);` - zakres osi

Kolor linii		Marker danych		Rodzaj linii	
znak	opis	znak	opis	znak	opis
b	niebieski	.	punkt	–	ciągła
g	zielony	o	okrąg	:	kropkowana
r	czerwony	*x*	znak ×	–.	linia typu kreska–kropka
c	turkusowy	+	plus	--	przerywana
m	fioletowy	*	gwiazdka	(spacja)	brak linii
y	żółty	*s*	kwadrat		
k	czarny	*d*	rąb		
		v	trójkąt z wierzchołkiem do dołu		
		^	trójkąt z wierzchołkiem do góry		
		<	trójkąt z wierzchołkiem w lewo		
		>	trójkąt z wierzchołkiem w prawo		
		p	pięciokąt		
		h	sześciokąt		

Rysunek 3: Podstawowe opcje wykresów w MATLABie (źródło: Osowski et al, 2006))

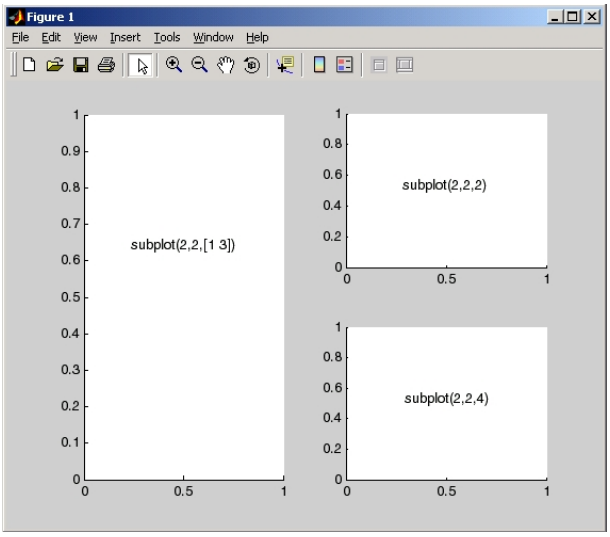
7.1 subplot

Każdą figurę można podzielić na kilka części. Podziału (i jednoczesnego wyświetlenia w konkretnej częsci). Składnia polecenia jest następująca:

`subplot(ile wierszy, ile kolumn, nr okna), plot(x,y);`

Numer okna liczony jest po kolei od lewego górnego rogu figury.

Przykładowy podział przedstawia fig.4



Rysunek 4: Przykładowy podział figury z użyciem subplot (źródło: MathWorks MatLAB help

8 Wybrane metody numeryczne

1. Rozwiązywanie układów równań liniowych $Ax = b$ z:
   wykorzystaniem SVD: `x=pinv(A)* b`
   wykorzystaniem dekompozycji qr, lu
2. miejsca zerowe funkcji: `fzero('fx',punkt_początkowy);`
3. całkowanie symboliczne: `syms x; int(x.^2)`
4. pochodne symboliczne: `syms x; diff(cos(x));`
5. interpolacje i aproksymacje: `polyfit(x,y,stopien)` - obliczanie współczynników wielomianu interpolującego wielomianu,
   liczenie wartości w punktach: `polyval(współczynniki,punkty)`
   przy wykorzystaniu funkcji sklejanych `spline(x,y,punkty)`

9 Zadania

1. Wygeneruj macierz prostokątną **A** o wymiarach 5x6 o rozkładzie normalnym o średniej $\hat{x} = 1$ i odchyleniu $\sigma = 0.5$. Następnie przemnóż 3 wers wg. zasady pierwsza kolumna $\times 1$, druga $\times 2$, itd. Następnie stwórz zbiorczy wykres, gdzie pierwsza kolumna stanowi dziedzinę (zbiór argumentów), a pozostałe kolumny zbiór wartości.
2. Wygeneruj wektor $\vec{x}$ wypełniony liczbami o rozkładzie równomiernym z zakresu <-50, 100>. Policz średnią, odchylenie oraz wyświetl histogram dla 15 klas (przedziałów).
3. Rozwiąż przy użyciu: dzielenia lewostronnego i dekompozycji SVD, równanie $\mathbf{A}\vec{x} = \vec{b}$, gdzie **A**=rand(4), $\vec{b} = [1:4]'$.
4. Wygeneruj tablicę A o rozmiarze 100x100 wypełnioną wg wzoru:

$$A_{n,m} = \sqrt{(n-z)^2 + (m-x)^2}; \tag{1}$$

   , gdzie z,x to odpowiednio ostatnia i przedostatnia cyfra numeru indeksu. Wykorzystaj makro do tego. Wynik zilustruj z wykorzystaniem funkcji `contourf`.
5. Wygeneruj tablicę **X** o rozmiarze 20x10 o rozkładzie normalnym ($\hat{x} = 10$, $\sigma = 1.75$) oraz wektor d składający się z 20 kolejnych liczb naturalnych. Następnie dokonaj dekompozycji LU macierzy **X**. W oparciu o powstałe macierze **L** i **U** dokonaj odwrócenia macierzy **X**. Rozwiąż równanie liniowe $\mathbf{X}\vec{m} = \vec{d}$. W pliku A.txt zapisz macierz odwrotną do **X**, a w pliku AT.txt macierz transponowaną do **X**.
6. Rozwiń w szereg funkcję exp(x). Oblicz wartość dla x=3 przy założonej tolerancji tol=2.5E-12; Wygeneruj wykres wartości rozwinięcia w zależności od ilości sumowanych członów. Na wykres nanieś wynik funkcji uzyskanej przez wbudowaną funkcję MATLAB: `exp(3)`. Powstają funkcję zapisz w pliku funkcyjnym przyjmującym i zwracającym różną ilość argumentów.
7. Wykonaj interpolację funkcji bramkowej. Porównaj przy użyciu wykresów wynik uzyskany z wykorzystaniem wielomianów (przetestuj 3 różne stopnie) z wynikiem uzyskanym przez funkcje sklejane.
8. Napisz plik funkcyjny, który będzie liczył pole i obwód koła.