PODSTAWY AUTOMATYKI
Zajęcia nr 2 - 3x 45min

Wprowadzenie do MATLAB v.2012

Liczby:
Utwórz zmienną x=2 i sprawdź jej wartość a następnie oblicz jej pierwiastek
polecenie: **sqrt(x)**

Przykład sposobu zapisu precyzji dla liczby
a= - 0.00341
b=20.03 10^{-3} (**b=20.03E-3 lub b=20.03e-3**)

Liczby zespolone:
c=1 + 3i lub c=1 + 3j lub c=1 + 3*i
polecenia do wyświetlania części rzeczywistej i urojonej:
real(c) imag(c)

Przydatne polecenia podczas pracy w oknie 'command window'
1. przypisywanie zmiennej wartości bez potwierdzenia przez środowisko (;) np. **x=5;**
2. użycie przejścia do następnego wiersza „ ... " (3 kropki)
3. jednoczesna deklaracja wartości kilu zmiennych **x=1; y=2; z=3; (średnik)**
4. zapisz polecenie **x=1 y=2 z=3**
5. zapisz polecenie **4/0** (inf)

Korzystanie z help-a
Polecenie **help** nazwa funkcji
np. **help sqrt**

Macierze
- Definiowanie macierzy:
A= 0 2 -10
7 6 1
A=[0 2 -10; 7 6 1] lub
A=[0 2 –10 (enter – przejście do następnego wiersza)
7 6 1]

- wektor wierszowy **B=[1 2 3 4 5]**

- wektor kolumnowy **C= [3; 4; 5]** lub

C= [3
4
5]

Macierz o wartościach zespolonych
D= 2+3i -1+1.5i
3-7i 2i

D=[2 -1; 3 0]+ i* [3 1.5; -7 2]
lub
D=[2+3i -1+1.5i; 3-7i 2i]

Budowanie macierzy z podmacierzy
Z macierzy A = 3 -1
5 0
oraz wektorów B=[2 5 8] C= [7 ;6]

utwórz macierz D=[B
C A]
A=[3 -1; 5 0]; B=[2 5 8] C=[7;6]

polecenie: D=[B; C A]

Użycie dwukropka, generowanie wektorów

x= $wart_{min}$: krok : $wart_{max}$

wygeneruj wektory używając symbolu '**:**'

x=[1 2 3 4 5 6]

y =[-1 – 0.8 – 0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1]

wygeneruj macierz **A=[1 2 3 4 5 6 7 8 9 10**
1 3 5 7 9 11 13 15 17 19]

Sprawdź co dają polecenia :
X=1:5'
X=[1:5]'
Y=1+1:5
Y=1+[1:5]
pi

!!! uwaga czy można zmienić wartość stałej pi ?

Wygeneruj wektor 10 elementowy **Z** o równomiernym rozkładzie elementach z przedziału (0, pi/2)

linspace (0, pi/2, 10)

nie jesteś pewny jak działa polecenia → (help linspace)

Dostęp do elementów macierzy

X= [1 2 6 7 8 23 55 56 78]

A = 1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 -1 -2
0 4 5 9 2 1

Proszę sprawdź co dają następujące polecenia:

X(i:j) – elementy wektora x o numerach od **i** do **j** **np. polecenie X(3:6)**

A(i,:) – wszystkie elementy w wierszu i macierzy **A**

A(1,:) – zwraca wiersz **nr 1**

A(:,2) – zwraca kolumnę **nr 2**

A(i,j:l) – zwraca elementy w wierszu **i** macierzy **A** o numerach od **j** do **l**

A(i:k, j:l) – analogicznie tylko dla wierszy i kolumn

A(x, j:l) - wszystkie elementy w komunach od **j** do **l** macierzy A o numerach określonych przez elementy wektora x

A(:,:) = A

A(:) – wektor kolumnowy

Ćwiczenia:

1. Wyświetl 3 wiersz macierzy A
2. Wyświetl jej drugą i trzecią kolumną i przypisz wynik zmiennej D
3. wyświetl drugą i czwartą kolumnę
4. wyświetl fragment znajdujący się miedzy wierszami 2 i 3 oraz miedzy kolumnami 2 i4

odp:

1. **A(3,:)**
2. **D=A(:,2:3)**
3. **A(:,[2 4])**
4. **A(2:3, 2:4)**

Usuwanie fragmentu macierzy **polecenie []**

- usuń 3 wierz macierzy A

polecenie A(3,:) =[]

A =1 2 3 4 5 6
7 8 9 10 11 12
0 4 5 9 2 1

Proszę przećwiczyć następujące polecenia na swoich macierzach.

Rozmiary macierzy
size(A)
[n m]=size(A)
lub
n =size(A,1)
lub
m = size(A,2)

length(x) zwraca długość wektora

Arytmetyka macierzowa		
	Macierzowa	**„Tablicowa”**
Dodawanie	A+B	tak samo
Odejmowanie	A-B	tak samo
Mnożenie	A*B	A.*B=B.*A
Dzielenie	A/B lub B/A	A.\B=B.\A
Potęgowanie	A^2=A*A	A.^A
Transpozycja	A’	tak samo

Ćwiczenia
1. Zdefiniujcie macierze np.: A=[1 2; -5 -14] B= [1 1 ; 0 -7] lub swoje ☺

Oblicz i sprawdź arytmetykę macierzową i tablicową
- A*B i B*A
- A.*B i B.*A (co daje użycie operatora kropki ?)
- A^3
- $A^{.3}$
- oblicz iloczyn **$(A*B)^{-1}*(A*B)$**
Polecenie (A*B).^(-1)*(A*B)

- wyznacz macierz **$C=(A+B^T)/2$**
Polecenie (A+B’)./2 lub (A+B’)/2

Uwaga: jeśli coś nie wychodzi to sprawdź czy rząd macierzy pozwala na przeprowadzenie operacji arytmetycznej

Wykonaj mnożenie **A*x**
2. x=[1 4] A=[4 1; 7 2]
dlaczego błąd ??

A może tak (A*x’) dlaczego?

Przydatne funkcje generujące macierze

eye(m) lub eye(m,n) - macierz jednostkowa
eye(2) ans = 1 0
0 1
Sprawdź też:

ones(m) lub ones(m,n)

zeros() lub zeros(m,n)

rand() lub randn() – generowanie wektorów lub macierzy o elementach

Utworzenie macierzy diagonalnej z elementów wektora x=[1 4 5 78]
Polecenie:
A=diag(x)

```
A = 1    0    0    0
    0    4    0    0
    0    0    5    0
    0    0    0   78
```

Oraz odwrotnie - utworzenie wektora z elementów diagonalenej macierzy
Polecenie:
x=diag(A)

Macierz odwrotna = A^(-1)
inv(A)

Ilość elementow w macierzy
numel(A)

Ile wymiarowa jest macierz
ndims(A)

size(A) zwraca ilość wierszy i kolumn

Powielenie macierzy **A, m** razy w poziomie i **n** razy w pionie
repmat(A,n,m)
przykład dla zadeklarowanej macierzy A:
repmat(A,2,3)

reshape(A,n,m) – utworzenie macierzy o **n** wierszach i **m** kolumnach z elementów branych kolejno **kolumnami** z macierzy **A:**
Uwaga: jeśli A nie zawiera $m_x n$ elementów, to pojawi się komunikat o błędzie

reshape(A,2,12)

Macierz trójkątna dolna
tril(A)

Macierz trójkątna górna
triu(A)

Obrócenie A o 90^o w kierunku przeciwnym do ruchu wskazówek zegara
rot90(A)

Inne przydatne funkcje:

all(x) – zwraca 1 jeśli wszystkie elementy są niezerowe (wektor/ macierz)
any(x) – zwraca 1 jeśli wykryje jakiś element o wartości 0
cumsum(x) – zwraca sumę wszystkich elementów
diff(x) – różniczka elementów wektora **[X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)]**

funkcje do wyszukiwania i porównywania
find() – znajduje niezerowe elementy i podaje ich indeksy
isempty() – czy jest pusty
isequal() - czy jest równy
isfinite() – czy przyjmuje wartości skończone
isinf() – czy przyjmuje wartość nieskończoną
islogical() – czy zawratość wektoa jest tylu boolean
isnan() – czy nie jest liczbą (sprawdzenie czy jest łańcuchem znaków)
isnumeric() – czy jest liczbą

Więcej o operacjach macierzach znajdziecie w opracowaniu:
MATLAB array manipulation tips and tricks
http://home.online.no/~pjacklam/matlab/doc/mtt/index.html

Funkcje i stałe matematyczne

Stałe w matlabie to
- **pi**
- **i** lub **j** sort(-1)
- **eps** względna dokładność zmiennoprzecinkowa
- **inf** lub **Inf**
- **NaN** lub **nan** (not a number)

W matlabie można zmienić format (precyzję) wyświetlanych liczb:

help **format**

Wypróbuj polecenia:

short
short e
long
long e
short g
long g
hex
+
bank
rat
compact
loose

Funkcje matematyczne

Proszę o sprawdzenie poleceń na zadeklarowanych dowolnie liczbach / wektorach/ macierzach**:**

sin(z) cos(z) tan(z) cot(z) - trygonometryczne (**wynik radiany**)
asin(z) cos(cos) atan(z) acot(z) – cyklometryczne (**wynik radiany**)
sinh(z) cosh(z)… - hiperboliczne (wynik radiany)
asinh(z)…..acosh(z)… (wynik radiany)
sqrt(z) - pierwiastek
exp(z) - e^z
log(z) – ln z (jeśli z<0 wynik jest zespolony)
log2(z) – $\log_2 z$ (jeśli z<0 wynik jest zespolony)
log10(z)- $\log_{10} z$ (jeśli z<0 wynik jest zespolony)

opearacje - liczby zespolone
abs(z) – moduł liczby zespolonej **|z|**
angle(z) – zwraca arg liczby zespolonej
real(z) – cześć rzeczywista
imag(z) – cześć urojona
conj(z) liczba zesp sprzężona
complex(x,y) == x+y*i

ceil(x) - zaokraglenie w górę
floor(x) - zaokraglenie w dół
fix(x) - dodatnia w dół ujemna w górę
round(x) - do najbliższej całkowitej
rem(x,y) - reszta z dzielenia =**x-n*y** gdzie **n=fix(x/y)**
mod(x,y) - modulo reszta z dzielenia **=x-n*y** gdzie **n=floor(x/y)**
sign(x) - zwraca znak liczby
max(x) – największy element wektora
min(x) – najmniejszy element wektora
sum(x) - suma elementów wektora x lub wektora kolumnowego macierzy
prod(x) – iloczyn elementów wektora
mean(x) – średnia arytmetyczna

Łańcuchy znakowe

Przykład deklaracji:
s='Automatyka'
sprawdź polecenie: **a=double(s)**

a = 65 117 116 111 109 97 121 107 97

<u>konwersja odwrotna</u>
s=char(a)
można sprawdzić czy wektory są sobie równe – plecenie isequal
isequal(s,b)
łączenie w macierze łańcuchów znakowych
A=['Podstawy ' s] lub **A=['Podstawy'; s]**

Spróbujcie wyświetlić **A(5)** ?

Przydatne operacje na łańcuchach znaków:

length(s) – zwraca długość łańcucha
deblank(s) – usuwanie spacji z końca łańcucha **s=deblank(s)**
findstr(s1,s2) przydatne do szukania łańcucha znaków w łańcuchu **s1**
np. **s1= Podstawy Automatyki**
findstr(s1,'A') ans =9 – podaje pod którym indeksie w wektorze s znajduje się litera A
lower(s) – wszystko małymi literami
upper(s) – wszystko dużymi
strcat(s1,s2,s3,….) – łączenie łańcuchów konkatenacja
strcmp(s1,s2) s2 – może być macierzą łańcuchów – funkcja do porównywania
strcmpi(s1,s2) porównanie bez rozróżniania wielkości liter
strncmp(s1,s2,n) – porównuje n pierwszych znaków w dwu łańcuchach

Konwersja

int2str(x) – całkowita na łańcuch (**co się stanie z liczbą niecałkowitą?**)
num2str(x) – konwersja liczby, macierzy polecenia na łańcuch
str2double(s) – łańcuch na liczbę rzeczywistą lub zespoloną

Okno poleceń funkcje edytorskie

clc – czyści cały ekran okna 'command window'
home – umieszcza kursor w lewym górnym rogu
diary *plik.txt* - polecenie do zapisu historii wszystkiego co piszemy w oknie poleceń do pliku
diary off/on – wyłącza/ włącza zapis
save – zapisuje binarnie wszystkie zmienne do pliku matlab.mat
save *plik.mat* – do zadeklarowanego pliku
save *plik.mat* *zmienna*– zapisuje tylko zmienne wymienione jako zmienna

uwaga: zapis jest binarny nie zapisuje się grafiki

load - wczytuje zmienne do pamięci
load *plik*
load *plik.rozsz.* – wczytuje zmienne zapisane w pliku tekstowym o podanej nazwie
dane musza być tablicą prostokątną – wczytane dane zostaną zapisane w macierzy o nazwie **plik**
who – wyświetla listę wszystkich zmiennych w środowisku
whos - wyświetla listę wszystkich zmiennych oraz rozmiar i rodzaj
clear - usuwa wszystkie zmienne
clear a z1 s2 – usuwa tylko wymienione zmienne (a z1 z2)
clear global z – tylko zmienną globalna z
clear all – usuwa zmienne i funkcje

Polecenia przydatne w pracy z plikami/katalogami w systemie operacyjnym

dir lub ls
cd
delete
pwd
!polecenie – wykona polecenie systemu operacyjnego np. **!md** (make dir) utworzy katalog
path – wyświetla listę do komponentów w matlabie
path(path,s1) – dodaje nowy katalog do listy określony łańcuchem znaków s1

czas w matlabie

clock
date
etime(t2,t1) – podaje różnicę pomiędzy t1 i t2 uwaga: t1 i t2 muszą być w formacie 'clock'
tic – start timera z zerowaniem
toc – wyświetla ile upłynęło od wyzwolenia polecenia tic

Ćwiczenia do przetestowania wiedzy i utrwalenia :

1. Utwórz macierz o rozmiarze 6x9

- o wszystkich elementach = 0
- o wszystkich elementach = 1
- wypełnioną liczbami pseudolosowymi

2. Oblicz moduł liczby zespolonej z=3 - 6i , argument, wyznacz liczbę sprzężoną
3. Oblicz $e^{2\sin(2x)}$ dla x=1:0.001:1000 oraz podaj czas wykonywania tych obliczeń
4. Utwórz wektor x=[3 4 5 0.1]

- utwórz macierz diagonalną z elementami wektora x
- za pomocą polecenia reshape utwórz z wektora macierz postaci

    1 7 13 19
    2 8 14 20
    …….
    6 12 18 24

5. Wyświetl wszystkie zmienne lokalne i zapisz je do pliku data

- usuń wszystkie zmienne
- wczytaj zmienne z pliku data do pamięci
- wyświetl zawartość katalogu roboczego utwórz swój katalog w którym będziesz pracować na zajęciach i przypisz jego ścieżkę dojścia do komponentów w matlabie.

Uwaga: Dla studentów katalogiem roboczym jest **Pulpit** systemowy. Inne katalogi mogą mieć zablokowaną możliwość zapisu.

Programowanie , funkcje oraz konstrukcja skryptu m-file w Matlabie

Na zadeklarowanych zmiennych sprawdź wyrażenie warunkowe:

```
A==B
A~=B
A<B
A>B
A<=B
A>=B
```

Operatory logiczne

```
A|B
A&B
~A
```

Sprawdź polecenia:

all(A)
any(A)
isequal(A,B,..)
isempty(A,B..)
Instrukcja IF

Instrukacja warunkowa IF

if *wyrażenie*
Instrukcja
end

If *wyrażenie*

elseif *wyrażenie*
instrukcja
.
..
else
instrukcja
end

Instrukcja SWITCH

switch *wyrażenie →(może być liczbą lub łańcuchem znaków)*
    case *wartość 1*
     instrukcje
    case *wartosc 2*
     instrukcje
    …
    otherwise
     instrukcje
end

ćw. utwórz macierz A o rozmiarze 4x4 która w zależności od wartości c ma być
c=0 – wszystkie elementy 0
c=1 – wszystkie elementy 1
c=2 – macierz jednostkowa
c – macierz losowa

```
switch c
    case 0
     A=zeros(4)
    case 1
     A=ones(4)
    case 2
     A=eye(4)
otherwise
    A=rand(4)
end
```

Pętla FOR

FOR warunek, - wykonuj instrukcję tyle razy ile deklaruje warunek
instrukcje;
END

Warunek definiujemy następująco:
i=1:10 lub i=1:1:10

ćw. utwórz macierz 5x4 taką ze Aij=(i+j)/(j+j+1)

```
for i=1:5,
    for i=1:4,
        A(i,j)=(i+j)/(i+j+1);
    End
End
```

Pętla WHILE

while wyrażenie
    instrukcje
end

ćw. oblicz wartość 100 elementów ciągu
u1=1
u2=1
..
$u_n=u_{n-1}+u_{n-2}$ dla n>2

```
u1=1; u2=1; i=2; n=100 % warunki początkowe
while i<n
    u3=u1+2;
    u1=u2;
    u2+u3;
    i=i+1;
end
```

Instrukcja BREAK
Zatrzymuje działanie pętli for oraz while

```
while 1
n=n+1;
if n>5, break, end
end
```

n=6

Definicja funkcji oraz tworzenie skryptu m-file

function [wynik$_1$, wynik$_2$,…,] = nazwa_funkcji(arg$_{we1}$,arg$_{we2}$,..,)

ćw. obliczanie $e^{\sin(x)}$ dla dowolnego parametru x

```
function [y] = esin(x)
% Funkcja oblicza e do sin(x)
% komentarz – ta cześć będzie się wyświetlać przy wywołaniu polecenia help esim
y=exp(sin(x));
```

taką funkcję zapisujemy w tzw. M-file pod nazwa funkcji esin.m w swoim katalogu roboczym

wywołanie liczba=esin(3);

wywołaj help esim %tak się definiuje opis do funkcji!

ćw. obliczanie silni

```
function [s]=silnia(n)
%funkcja oblicza n! dla n<1 zwraca 0
s=0;
if(n<1) return
else
        s=1;
        for i=2:n,
                s=s*i;
        end
end
```

ćw. utwórz macierz o elementach a(ij)= 1/(i-j) dla i ~=j oraz 0 dla i=j

ćw. wygeneruj macierz 10x10 wypełnioną liczbami pseudolosowymi i wyświetl wszystkie elementy macierzy które mieszczą sie w przedziale (0.2, 0.5)

ćw. napisz skrypt przeliczający prędkość w [km/h] na [m/s]

Rysowanie wykresów funkcji:

Obsługa okna graficznego
Polecenia:
figure
figure(n)
close close(n)
clone all
clf

podział okna → polecenie **subplot(m,n,p) –** dzieli okno na n_xm pod okienek, oraz uaktywnia okienko o numerze p

plot(x,y)
plot(y)
plot(x,y,s)
plot(x1,y1,x2,y2,…)
plot(x1,y1,s1x2,y2,s2,…)

gdzie:
x - argumenty wektora/macierzy funkcji
y – wartości wektora /macierzy funkcji
s – symbole określające wygląd linii np. ‘o’, ‘x’

np.
w=rand(100);
plot(w) – wygeneruje wykres wektora pseudolosowego wg jego wzrastających indeksów

s=rand(100)
plot(s,w,’-o’) - wartości wektora w wg elementów wektora s punkty zaznaczone kółkami

jak generować argumenty

linspace(x1,x2,N) - generuje wektor N elementów rozłożonych równomiernie od x1 do x2

linspace(x1,x2) – generuje domyślnie 100 liczb z przedziału x1 do x2

znaki rodzaj linii:	**kolory linii**
‘-‘	' y’ - żółty
‘__‘	‘m’ - karmazyn
‘:’	‘c’ - turkus
‘-.’	‘r’ - czerwony
	‘g’ – zielony -
	‘b’ - niebieski
	‘w’ - biały
	‘k’ - czarny

znaki : ‘+’ ‘*’ ‘.’ ‘o’ ‘x’ ‘s’ ‘d’ ‘p’ ‘h’ ‘v’ ‘^’ ‘<’ ‘>’

ćwiczenie :
na wykresie w oknie fugure(1) dla argumentów z zakresu x=<0;2pi>
narysuj przebiegi funkcji:
- sin(x) - czerwoną linią
- cos (x) zieloną linią z kreseczkami '|"
- $\sin^3(x)$ niebieską linią kropkowaną

Opisywanie wykresów

xlabel('string')
ylabel('string')
title('text')
test(x,y,'text')
legend(s1,s2,s3..,)
grid on/off
axis(xmin,xmax,ymin,ymax) - zakresy dla argumentów x i wartości y
axis ij - zmienia ukł. współ. na układ macierzowy z początkiem w lewym górnym rogu
axis xy – ukł. kartezjański (domyślny)
axis equal – jednostka na podziałce osi x i y taka sama
axis on/off – wł/wył. osie współrzędnych na wykresie
v=axis – zwraca wektor wierszowy v=[x_{min},max,y_{min},max]

więcej ustawień funkcji axis → **help axis**

<u>funkcja FPLOT</u>

fplot(f,[x0,xk])
f- łańcuch znaków wywołujący funkcję z m-file
x0, xk początek i koniec przedziału rysowania funkcji

np.
y=cos(10e^x) w przedziela <-2,2>

funcos.m
function [y]=funcos(x)

x=-2:0.1:2;
y=funcos(x);
plot(x,y);

lub za pomocą funkcji fplot
fplot('funcos',[-2,2]);

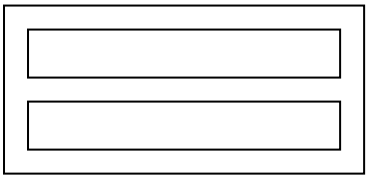
wykresy logarytmiczne:

loglog(y,y,s) – rysuje wykres używając skali logarytmicznej na obu osiach
semilogx(x,y,s) – wykres używając skali log na osi x
semilogy(x,y,s) - ------//----------y

logspace(x1,x2,N) – generuje wektor wierszowy o N elementach o wartościach rozmieszczonych logarytmicznie od $\mathbf{10^{x1}}$ do $\mathbf{10^{x2}}$

<u>ćwiczenie dla utrwalenia:</u>

1. W jednym okienku figure przedstaw oddzielnie wykresy funkcji sin(x) oraz cosinus(x) na przedziale <-2pi,2pi>



PRZYKŁADY OBLICZEŃ NUMERYCZNYCH

Miejsce zerowe funkcji

x1=zero(f,x0) – miejsce zerowe funkcji f, x0- początkowe przybliżenie wartości szukanego miejsca zerowego

f=inline('cos(2*x-pi)');
x=fzero(f,3)
x = 2.3562

fzero(inline('abs(x)+1'),1)
Exiting fzero: aborting search for an interval containing a sign change
because NaN or Inf function value encountered during search.
(Function value at -Inf is Inf.)
Check function or try again with a different starting value.
ans = NaN

fzero('cos',3)
ans = 1.5708

Minimum funkcji:

f1=inline('sin(x)+cos(x)');
fplot(f1,[-5,5])
x1=**fminbnd(f1,-5,5) – szukanie minimum na przedziale**
x1 = -2.3562
f1(x1) = -1.4142

Pierwiastki wielomianu:

$W(x,a)=a_1x^n+a_2x^{n-1}+..+a_nx+a_{n+1}$

r=roots(a) – pierwiastki wielomianu

roots([1 3 -4])
ans = -4 1
a=poly(r) – zwraca wektor współczynników W(x,a) o pierwiastkach r
p=polyval(a,x0) - zwraca wartości wielomianu W(x,a) w punkcie x=x0
gdzie x0 – może być wektorem

Elementy algebry liniowej

rank(A) - rząd macierzy
det(A)
norm(A)
L=eig(A) – wektor L zawierający wartości własne