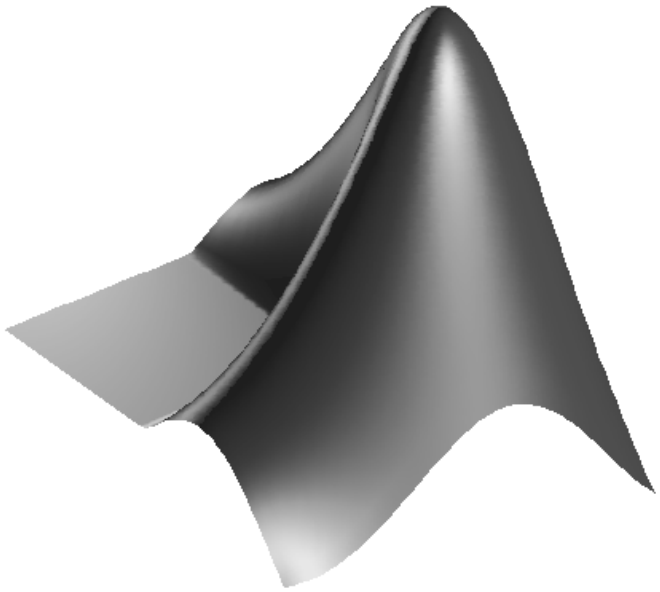

KURS MATLAB I

Rok 2005/2006, semestr letni

Uniwersytet Warszawski
Wydział Fizyki

Ryszard Buczyński, Rafał Kasztelanic

Spis Treści

Wstęp 3
Opis środowiska Matlaba 4
Operacje algebraiczne na wektorach i macierzach 9
Wizualizacja danych – Wykresy dwuwymiarowe 13
Wizualizacja danych – Wykresy trójwymiarowe 16
Podstawy programowania: skrypty i funkcje 19
Instrukcje w Matlabie 22
Inne przydatne funkcje 24
Rozwiązywanie równań nieliniowych 27
Rozwiązywanie układów równań liniowych 28
Interpolacja i aproksymacja funkcji 30
Podstawy statystyki w Matlabie 31

KURS MATLAB I

Rok 2005/2006 semestr letni, wymiar 15h

Prowadzący:

dr Ryszard Buczyński, ryszard.buczynski@mimuw.edu.pl,
dr Rafał Kasztelanic, kasztel@mimuw.edu.pl

Zakład Optyki Informacyjnej, Instytut Geofizyki, Wydz. Fizyki UW
Zajęcia odbywają się w Instytucie Geofizyki, ul. Pasteura 7, V piętro, pok. 508. (lub pok. 106)

Oprogramowanie:

Matlab, *The MathWorks, Inc.;* wersja 6.03; platforma UNIX/LINUX.

Charakterystyka kursu:

Poziom podstawowy, wymagana znajomość podstawowych pojęć matematycznych z zakresu algebry, analizy matematycznej i prawdopodobieństwa, znajomość programowania nie jest konieczna, ale mile widziana.

Forma zaliczenia:

Do zaliczenia Kursu na ocenę dostateczną lub zal. wymagane jest zaliczenie wszystkich ćwiczeń-laboratoriów. Ocena końcowa wystawiana jest przez prowadzącego na podstawie osiągniętej sprawności i postępów w posługiwaniu się MATLABem, oraz kreatywności studenta.
Obecność na wszystkich zajęciach jest obowiązkowa, dopuszczamy jedna nieobecność, przy większej ilości wymagane jest zwolnienie lekarskie. Nieobecność nie zwalnia studenta z zaliczenia poszczególnych zadań.
W czasie Kursu przewidziane są dwa krótkie (15 min.) kolokwia na godz. 7 i 14-tej.
Ostatnia godzina 15-ta jest przeznaczona na wystawianie ocen i ewentualne poprawki.

Spis omawianej problematyki:

1. Opis środowiska Matlaba
2. Operacje algebraiczne na wektorach i macierzach
3. Wizualizacja danych – Wykresy dwuwymiarowe
4. Wizualizacja danych – Wykresy trójwymiarowe
5. Podstawy programowania: skrypty i funkcje
6. Instrukcje w Matlabie
7. Inne przydatne funkcje
8. Rozwiązywanie równań nieliniowych
9. Rozwiązywanie układów równań liniowych
10. Interpolacja i aproksymacja funkcji
11. Podstawy statystyki w Matlabie

Literatura:

1. Matlab: Intro, Demo, manual online.
2. A. Zalewski R. Cegieła, Matlab – *Obliczenia numeryczne i ich zastosowania*, Wyd. Nakom, Poznań 1996.
3. B. Mrożek, Z. Mrożek, Matlab uniwersalne środowisko do obliczeń naukowo-technicznych, Wyd. PLJ, Warszawa 1996
4. B. Mrozek, Z. Mrozek, Matlab 6 – poradnik użytkownika.

TEMATY

OPIS ŚRODOWISKA MATLABA

Temat 1
Matlab: przeznaczenie oprogramowania i opis pakietu

Temat 2
Operowanie Matlabem w środowisku Linux

Okna: workspace, directory, history, array editor, editor, …

Temat 3
Różnice między wersjami Matlaba – funkcja *ver*

```
>> ver % podaje numer wersji Matlaba oraz numery zainstalowanych dodatków
```

Temat 4
Zapoznanie się z narzędziami wprowadzającymi Matlaba – funkcje *demo*, *peaks, bench*

```
>> demo % wyświetla dostępne przykłady
>> płaks % przykładowa funkcja 2 zmiennych
>> bench % sprawdzenie szybkości pracy Matlaba – benchmark
```

Temat 5
Poszukiwanie znaczeń funkcji i skryptów – funkcja *help*

```
>> help % wypisuje linki do wszystkich plików pomocy
>> help plot % wypisuje pomoc dotyczącą funkcji plot
```

Temat 6
Szukanie za pomocą słów kluczowy: *lookfor*

```
>> lookfor bessel % przeszukuje pliki pomocy szukając słowa kluczowego bessel
```

Temat 7
Znaczenie średnika na końcu polecenia

Średnik kończący komendę w Matlabie powoduje, że wynik działania danej komendy nie będzie wyświetlany na ekranie.

Temat 8
Symbole operatorów

=	Przypisanie wartości
[]	Tworzenie macierzy, list argumentów wyjściowych funkcji
()	Listy argumentów wejściowych funkcji, kolejność działań matematycznych
.	Kropka dziesiętna, część operatorów arytmetycznych
..	Katalog macierzysty
...	Kontynuacja polecenia jest w następnej linii
, .	Symbole separacji argumentów funkcji, indeksów, itp.
;	Koniec wiersza macierzy, koniec polecenia bez wypisywania odpowiedzi
%	Początek linii komentarza
:	Generowanie wektorów, indeksowanie macierzy
‘	Początek i koniec wprowadzania łańcuchów znakowych, transpozycja macierzy, sprzężenie macierzy
!	Komenda sytemu operacyjnego

Temat 9
Zmienne specjalne i stałe

ans	Zmienna robocza, automatycznie przyjmuje daną wartość, jeśli nie nadano jej nazwy
computer	Nazwa komputera, na którym działa Matlab
eps	Precyzja zmiennoprzecinkowa
flops	Licznik operacji zmiennoprzecinkowej
i, j	Jednostka liczby urojonej
inf	Nieskończoność
NaN	Wartość nieokreślona (zwykle oznacza wprowadzenie wartości nieliczbowej jako argumentu funkcji matematycznej)
nargin	Liczba argumentów wejściowych funkcji
nargout	Liczba argumentów wyjściowych funkcji
pi	3.1415926....
realmax	Największa dostępna liczba rzeczywista
realmin	Najmniejsza dostępna liczba rzeczywista

Temat 10
Podstawowe funkcje matematyczne

abs	Wartość bezwzględna, moduł liczby zespolonej, wektor wartości znaków łańcucha
acos, acosh	Arcus cosinus, arcus cosinus hiperboliczny
acot, acoth	Arcus cotangens, ...
acsc, acsch	Arcus cosecans, ...
angle	Kąt fazowy dla liczby zaspolonej w radzianach
asec, asech	Arcus secans, ...
asin, asinh	Arcus sinus, ...
atan, atanh	Arcus tangens, ...
atan2	Arcus tangens, wynik w przedziale [-π, π]
ceil	Zaokrąglenie w górę, sufit
conj	Liczba sprzężona do liczby
cos, cosh	Cosinus, ...
cot, coth	Cotangens, ...
csc, csch	Cosecans, ...
exp	e do potęgi argumentu
fix	Zaokrąglenie w kierunku zera
floor	Zaokrąglenie w dół, podłoga
gcd	Największy wspólny podzielnik

imag	Część urojona liczby zespolonej
lcm	Najmniejsza wspólna wielokrotność
log	Logarytm naturalny argumentu
log10	Logarytm dziesiętny argumentu
real	Część rzeczywista liczby zespolonej
rem	Reszta z dzielenia
round	Zaokrąglenie do najbliższej liczby całkowitej
sec, sech	Secans, ...
sign	Znak funkcji
sin, sinh	Sinus, ...
sqrt	Pierwiastek kwadratowy
tan, tanh	Tangens, ...

Przykład:

```
>> abs(5+3i) % wyświetla wartość bezwzględną liczby zespolonej
```

Temat 11
Wprowadzanie zmiennych różnych typów

```
>> a=´łańcuch wprowadzany´; % zmienna łańcuchowa
>> z=3+2i; zmienna zespolona (część urojoną oznaczamy literą i lub j
```

Temat 12
Wprowadzanie precyzji wyświetlanych wyników – funkcja format

Do ustalenia precyzji wyświetlania wyników służy funkcja FORMAT.
UWAGA: Wszystkie obliczenia w MATLABie wykonywane są w podwójnej precyzji.

Polecenie	Wartość	Opis
Format short	3.1416	5 cyfr, reprezentacja stałoprzecinkowa
Format long	3.14159265358979	15 cyfr, reprezentacja stałoprzecinkowa
Format shortE	3.1416e+000	5 cyfr, reprezentacja zmiennoprzecinkowa
Format longE	3.141592653589793e+000	15 cyfr, reprezentacja zmiennoprzecinkowa
Format shortG	3.1416	5 cyfr, reprezentacja stało- lub zmiennoprzecinkowa
Format longG	3.14159265358979	15 cyfr, reprezentacja stało- lub zmiennoprzecinkowa
Format hex	400921fb54442d18	Liczba w układzie szesnastkowym
Format bank	3.14	2 liczby dziesiętne, np. złoty i grosze
Format rat	355/113	Przybliżona wartość liczby w postaci ułamka
Format +	+	Informacja o znaku liczby

Temat 13
Informacja i usuwanie zmiennych z przestrzeni roboczej – funkcje *who*, *whos*, *clear*

```
>> who % informacja o dostępnych zmiennych, same nazwy
>> whos % pełna informacja o dostępnych zmiennych
>> clear a % usunięcie z przestrzeni roboczej zmiennej a
>> clear all % usunięcie wszystkich zmiennych
```

Temat 14
Zmienne losowe w Matlabie

```
>> rand % rozkład równomierny w przedziale (0,1)
>> randn % rozkład normalny o odchyleniu standardowym 1 i wariancji 1
```

Po każdym uruchomieniu Matlaba funkcja rand startuje od tych samych wartości. Aby zacząć od innej wartości należy wywołać funkcję rand w następujący sposób:

```
>> rand('state',sum(100*clock)) % wartość początkowa na podstawie wskazań zegara
```

Temat 15
Informacje o operatorach – *help ops*

*	mnożenie macierzy
/	dzielenie macierzy (lewej przez prawą)
\	dzielenie macierzy (prawej przez lewą)
^	podnoszenie do potęgi
‘	sprzężenie macierzy
.*	mnożenie tablicowe
./	dzielenie tablicowe (lewej przez prawą)
.\	dzielenie tablicowe (prawej przez lewą)
.’	transpozycja macierzy
.^	tablicowe podnoszenie do potęgi

Temat 16
Operatory relacji

==	Relacja równości
~=	Relacja nierówności
<	Relacja mniejszości
>	Relacja większości
<=	Relacja mniejsze-równe
>=	Relacja większe-równe

Temat 17
Operatory logiczne

& (and)	Logiczne i
| (or)	Logiczne lub
~ (not)	Logiczne nie
Xor	Operacja exclusive or
any	Operacja logiczna - jeśli jakiś
all	Operacj alogiczna - wszystkie

Temat 18
Długie linie

```
>> x=1 + 1/2 + 1/3 + 1/4 + 1/5 + …
    1/6 + 1/7 + 1/8 + 1/9 + 1/ 10;
```

Temat 19
Kilka instrukcji w jednej linii

Poszczególne instrukcje oddzielamy przecinkiem.

Przykład:

```
>> x=2;, y=4;
```

Temat 20
Czyszczenie okna komend – funkcja *clc*

Temat 21
Wyprowadzanie na ekran tekstów – funkcja *disp*

```
>> disp('WYNIK: '), disp(2+2) % wyświetli WYNIK: 4
```

Dla znających składnię języka C wygodna może być w użyciu funkcja *fprintf()*

Temat 22
Wprowadzanie danych – funkcja *input*

Jeśli chcemy, aby użytkownik wprowadził jakąś zmienną stosujemy funkcję *input*

Przykład:

```
>> x = input('Podaj wartość: ')
```

```
Podaj wartość: 4
x = 4
```

Temat 23
Zapisywanie i wczytywanie zmiennych z pliku – funkcje *save*, *load*

Dokładny opis funkcji – help save, help load.

Wybrane polecenia:

```
>> save NazwaPliku x % zapisuje zmienną x w pliku NazwaPliku.mat
>> save NazwaPliku x -ascii % zapisuje zmienną x w pliku tekstowym NazwaPliku.mat
>> save NazwaPliku % zapisuje wszystkie zmienne w pliku NazwaPliku.mat
>> load NazwaPliku % wczytuje wszystkie zmienne z pliku NazwaPliku.mat
```

OPERACJE ALGEBRAICZNE NA WEKTORACH I MACIERZACH

Temat 24
Generacja macierzy za pomocą funkcji specjalnych Matlaba

eye	Macierz jednostkowa – z jedynkami na przekątnej
linspace	Wektor o wartościach rozłożonych równolegle
logspace	Wektor o wartościach rozłożonych logarytmicznie
meshgrid	Macierz dla wykresów 3D
ones	Macierz jedynek
rand	Macierz losowa o rozkładzie równomiernym
randn	Macierz losowa o rozkładzie normalnym
zeros	Macierz zer
compan	Macierz stowarzyszona
hadamard	Macierz Hadamarda
hankel	Macierz Hankela
hilb	Macierz Hilberta
invhilb	Odwrotna macierz Hilberta
magic	Kwadrat magiczny
pascal	Macierz Pascala
toeplitz	Macierz Toeplitza
vander	Macierz Vandermondea
gallery	Para małych macierzy testowych

Przykład:

```
>> x=ones(3); % macierz kwadratowa 3x3 z samymi jedynkami
>> y=zeros(5,2); % macierz zer o 5 wierszach i 2 kolumnach
>> z=rand(1,5); % wektor liczb losowych o 5 elementach
```

Temat 25
Generacja macierzy przy użyciu dwukropka

Przykład:

```
>> a=j:k % generuje wektor [j, j+1, ..., k-1, k]
>> a=j:i:k % generuje wektor [j, j+i,j+2i, ...,k]
```

Temat 26
Wybór elementów macierzy

Przykład:

```
>> x(:,i) % i-ta kolumna macierzy a
>> y(i,:) % i-ty wiersz macierzy y
>> z(i,a:b) % kolumny macierzy z(a) … z(b)
>> a(:) % całą macierz w postaci wektora kolumnowego
>> b(j:k) % wypisuje elementy macierzy A od elementu j do elementu k
```

Temat 27
Generacja wektorów, alokacja pamięci

Ze względu na czas wykonywania operacji dobrze jest przed przystąpieniem do obliczeń stworzyć odpowiednie macierze do przechowywania danych

Przykład:

```
>> x(5)=0; % wektor 5 elementowy wypełniony zerami
>> y(5,7)=0; % macierz 5x7 wypełniona zerami, lub y=zeros(5,7);
```

Temat 28
Znaczenie spacji, przecinka i średnika w generacji macierzy

```
>> x=[1 2 3] % wektor poziomy {1, 2, 3}, równoważne x=[1, 2, 3]
>> y=[1;2;3] % wektor pionowy {1, 2, 3}
```

Temat 29
Macierze wielowymiarowe

Przykład:

```
>> x=zeros(i,j,k); % 3-wymiarowa macierz zer o i wierszach, j kolumnach oraz k warstwach
```

Mając gotowe macierze mogę je składać w macierze o większej liczbie wymiarów. Służy do tego funkcja cat(dim,a,b,…) gdzie dim określa wzdłuż którego wymiaru dokonywane jest złożenie macierzy.

Przykład:

```
>> a=zeros(3); , b=ones(3); % dane początkowe
>> x=cat(1,a,b) % macierz 2-wymiarowa a nad b, równoważne [a; b]
>> y=cat(2,a,b) % macierz 2-wymiarowa b za a, równoważne [a b]
>> z=cat(3,a,b) % macierz 3-wymiarowa o warstwach a i b
```

Temat 30
Działania macierzowe i tablicowe

Operator	Operacja macierzowa	Operacja tablicowa
Dodawanie	+	+
Odejmowanie	-	-
Mnożenie	*	.*
Dzielenie lewostronne	\	.\
Dzielenie prawostronne	/	./
Potęgowanie	^	.^

Temat 31
Operacje na elementach wektora

max(x)	zwraca największą wartość w wektorze x. Jeśli x jest macierzą funkcja max(x) zwraca wektor gdzie kolejne elementy określają największe wartości w każdej z kolumn.
min(x)	zwraca wartość minimalną w wektorze x.
sum(x)	zwraca sumę wszystkich elementów wektora x.
prod(x)	zwraca iloczyn wszystkich elementów wektora x.
diff(x)	zwraca różnicę między kolejnymi elementami wektora x. [x(2)-x(1), x(3)-x(2), ...].

Uwaga: Jeśli x jest macierzą powyższe funkcje odnoszą się do poszczególnych kolumn macierzy x.

Temat 32
Wektoryzacja

Matlab optymalizowany jest do wykonywania działań na wektorach i macierzach. Jeśli to tylko możliwe należy dążyć do wykonywania obliczeń na wektorach lub macierzach.

Przykład:

```
% można tak – sposób iteracyjny
>> x=1:10; , y=zeros(10);
>> for i=1:10, y(i)=x(i)^2;, end

% lepiej jednak tak – sposób macierzowy
>> x=1:10;
>> y=x.^2;
```

Temat 33
Operowanie macierzami

flipdim	Wywinięcie macierzy wzdłuż danego wymiaru
fliplr	Wywinięcie macierzy w kierunku lewo-prawo
flipud	Wywinięcie macierzy w kierunku góra-dół
reshape	Zmiana rozmiaru macierzy, zmiana liczby wymiarów
rot90	Obrót macierzy o 90 stopni
squeeze	Usunięcie 1 wymiaru
tril	Macierz trójkątna dolna
triu	Macierz trójkątna górna

Temat 34
Inne przydatne funkcje

size	Podaje rozmiar macierzy
Numer	Liczba elementów w macierzy
end	Ostatni element macierzy, wektora, …
Isequal	Sprawdza czy macierze są sobie równe (funkcje typu is*)
a>0	Macierz zerojedynkowa. Jedynki tam gdzie a>0
any(a>0)	1 gdy jakiś element macierzy >0
find	Znajduje elementy spełniające dane kryterium

Przykład:

```
>> [x,y]=size(a)
>> m=numel(a)
>> b=a(5:end) % elementy wektora od 5 do końca
>> b=(a>0)
>> c=find(a>2)
```

Temat 35
Macierze rzadkie

Macierzą rzadką nazywamy macierz, której przeważająca większość elementów równa jest 0 a tylko nieliczne mają wartość znaczące. W takim przypadku ze względów pamięciowych wygodnie jest pamiętać macierz nie jako tablicę, ale poszczególne liczby wraz z adresami.

Przykład:

```
>> a= sparse([1 2 2 4 4],[3 1 4 2 4],1:5) % tworzy macierz rzadką o elementach: (2,1)->2,
        (4,2)->4, (1,3)->1, (2,4)->3, (4,4)->5
```

Aby przekształcić macierz rzadką w macierz w postaci normalnej normalną korzystamy z funkcji *full()*.

WIZUALIZACJA DANYCH
WYKRESY DWUWYMIAROWE

Temat 36
Wykresy dwuwymiarowe funkcji – funkcja *plot*

plot(X) – rysuje wektor X w funkcji indeksu, w przypadku macierzy traktuje ją jak zestaw wektorów
plot(X,Y) – wykreśla wektor Y w funkcji wektora X, Gdy X lub Y jest macierzą to wektor jest rysowany odpowiednio w funkcji kolumn lub rzędów.
plot(X,Y,S) – wykreśla jak funkcja plot(X,Y) ale dodatkowo pozwala wybierać kolor, rodzaj linii i symbole punktów – patrz tabela poniżej.

Kolor
y – yellow
m – magenta
c – cyan
r – red
g – green
b – blue
w – white
k – black

Symbole punktów
. – point
o – circle
x – x-mark
+ – plus
* – star
s – kwadraty
d – romb
v – trójkąt w dół
^ – trójkąt w górę
< – trójkąt w lewo
> – trójkąt w prawo
p – pięciokąt
h – sześciokąt

Rodzaj linii
- – ciągła
: – kropkowana
-. – kropka-kreska
-- – kreskowana

Przykład:

```
>> plot(1:10,y) % wykreśla wektor od 1 do 10 w funkcji wektora y
>> plot(1:10,y, 'bx') – j.w. ale dodatkowo wykreśla go w kolorze niebieskim zaznaczając
      punkty krzyżykami.
>> plot(1:10,x, 'bx ', 1:10,y, 'r*') – wykreśla dwa wykresy na jednym
```

Temat 37
Wykresy dwuwymiarowe funkcji – funkcja *fplot*

fplot(F,P) – funkcja wykreśla funkcję F daną w postaci łańcucha w przedziale P.
Listę funkcji matematycznych predefiniowanych w MATLABie można uzyskać poprzez polecenie >> help elfun (funkcje podstawowe) i >> help specfun (funkcje specjalne)

Przykład:

```
>> fplot('2*sin(x) ',[0 2*pi]) % funkcja 2*sin(x) w przedziale od 0 do 2π
```

Temat 38
Wykresy dwuwymiarowe funkcji – funkcja *ezplot*

Bardziej ogólnymi funkcjami służącymi do rysowania wykresów także dla dwu zmiennych są funkcje typu ez*. Jedną z nich jest funkcja ezplot.
ezxplot(F,P) – funkcja wykreśla funkcję F w przedziale P.
Inne funkcje należące do kategorii ez* to: ezcontour, ezmesh, ezmeshc, ezpolar.

Przykład:

```
>> ezplot(x, 2*y, [0,2*pi]) % wykres funkcji parametrycznej typu x=x(t), y=y(2t), t∈[0,2π]
```

Temat 39
Wykresy dwuwymiarowe funkcji – funkcje *hist*, *starsi*, *bar*, *steam*

hist(x, m)	Wykreśla histogram z podziałem na m przedziałów.
stairs()	wykreśla wektor w postaci schodków od największego do najmniejszego elementu
bar(x)	wykreśla wektor w postaci słupków (bar)
stem(x)	wykreśla wektor w postaci linii pionowych (ystem)

Uwaga: Wywołanie n=hist(X) nie wyświetla wykresu, ale zlicza ilość elementów wektora w 10 równych przedziałach. Przedziały są tworzone na podstawie najmniejszej i największej wartości wektora

Temat 40
Rysowanie wielu wykresów na wspólnym wykresie graficznym – funkcja *hold*

```
>> hold on % wstrzymuje czyszczenie okna graficznego
>> hold off % przywraca tryb domyślny (każdorazowe czyszczenie okna)
>> ishold % testuje tryb rysowania wykresów
```

Temat 41
Otwieranie wielu okien graficznych – funkcje *figure*, *close*, *clg*, *cla*

```
>> figure % otwiera nowe okno graficzne
>> figure(n) % uaktywnia okno graficzne o danym parametrze,
>> close % zamyka okno aktywne lub okno z zadanym parametrem.
>> cla % czyści bierzący wykres
>> clf % czyści aktywne okno graficzne
```

Temat 42
Wykreślanie niezależnych wykresów w jednym oknie graficznym – funkcja *subplot*

Funkcja subplot służy do podziału okna graficznego na mniejsze fragmenty. Podziału można dokonać albo w układzie macierzowym albo podając dokładne wymiary wykresu.

Przykład:

```
>> subplot(m,n,p) % dzieli okno graficzne na M kolumn i N wierszy (M,N<9). P oznacza
      numer aktualnego wykresu. Można też wywołać jako subplot(mnp)

>> subplot('position',[lewy dolny szerokość wysokość]) % w aktywnym oknie graficznym
      tworzy nowy wykres w zadanym podoknie. Lewy, dolny – współrzędne lewego dolnego
      rogu podokna. Szerokość, wysokość – rozmiary podokna. Wszystkie rozmiary podaje
      się w stosunku do całości okna unormowanego do 1, np.: [0.5 0.5 0.5 0.5]
```

Temat 43
Skalowanie wykresów – funkcje *axis* i *log*-i

axis('auto')	domyślny tryb skalowania
axis([xmin, xmax, ymin, ymax])	wykreśla wykres w zadanych przedziałach osi X i Y
axis('off')	ukrywa osie
axis('on')	przywraca wyświetlanie osi
axis('equal')	osie mają proporcjonalne jednostki na obu osiach X i Y

loglog(x)	skala logarytmiczna na obu osiach
semilogx(x)	skala logarytmiczna na osi X
semilogy(x)	skala logarytmiczna na osi Y

Temat 44
Opisywanie wykresów

```
>> plot(x,y, 'r ') % wykres funkcji
>> title('To jest wykres') % Tytuł wykresu
>> grid off % wyłączenie wyświetlania siatki
>> xlabel('oś X') % podpis osi X
>> ylabel('oś Y') % podpis osi Y
>> text(2,4, 'tu jest punkt') % tekst wstawiony w punkcie (2,4)
```

Temat 45
Niestandardowe znaki w opisie wykresów

Do wypisywania niestandardowych znaków wykorzystywana jest składnia TeX.

```
>> text(1,1, '\alpha^{3/2}') % wypisanie w punkcie (1,1) tekstu α^{3/2}
```

Temat 46
Zmiana pozostałych parametrów funkcji graficznych.

```
>> plot(x,y, 'LineWidth',4, 'MarkerSize',10)
```

Informacje o poszczególnych elementach wykresu można znaleźć w helpie, np: *help line*

Temat 47
Modyfikacja wykresów w oknie graficznym

WIZUALIZACJA DANYCH
WYKRESY TRÓJWYMIAROWE

Temat 48
Funkcja meshgrid

Funkcja meshgrid – tworzy macierze opisujące położenie węzłów siatki prostokątnej. Służy do przygotowania danych niezbędnych do stworzenia większości wykresów 3D.

Przykład:

```
>> [X,Y]=meshgrid(x,y); % tworzy macierze X, Y na podstawie wektorów z węzłami siatki x, y
>> [X,Y]=meshgrid(x); % j.w. ale y=x
>> [X,Y,Z]=meshgrid(x,y,z) % tworzy 3 macierze wykorzystywane do wykresów
      volumetrycznych
```

Temat 49
Funkcja *mesh*

Mesh(X,Y,Z) – funkcja mesh rysuje siatkę opisaną przez macierze X,Y,Z. Gdzie macierze X, Y podają współrzędne punktów siatki a dane w macierzy Z określają wartość funkcji w punkcie (x,y).
Mesh(X,Y,Z,c) – c – indeksy kolorów w aktualnej mapie kolorów.

Przykład:

```
>> [x,y] = meshgrid(-3:.125:3); % generacja siatki
>> z = peaks(x,y); % tworzenie wartości funkcji w punktach (x,y)
>> mesh(x,y,z) % tworzy wykres 3D
```

Temat 50
Inne wykresy 3D typu oparte na funkcji meshgrid

contour3	Wykres konturowy
ezmesh	Wykres siatkowy
ezsurf	Wykres – powierzchnia
mesh	Wykres siatkowy
meshc	Wykres jak mesh + poziomice
meshz	Wykres jak mesh + zasłony na końcach
ribbon	Wykres wstążkowy
Surf	Wykres powierzchniowy
Surfc	Wykres powierzchniowy + poziomice
Surfl	Wykres powierzchniowy + cieniowanie
Waterfall	Wykres plasterkowy

Temat 51
Inne wykresy 3D

bar3	Wykres słupkowy
ezplot3	Wykres parametryczny
isosurface	Izowarstwy dla danych 3D
plot3	Linia w 3 wymiarach
scatter3	Wykres typu scatter
Slice	Przekrój przez wykres wolumetryczny

Przykład:

```
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t) % linia śrubowa
>> ezplot3('sin(t)','cos(t)','t',[0,6*pi]) % linia śrubowa
>> a=rand(5); generowanie danych
>> bar3(a) % wykres słupkowy
```

Temat 52
Obiekty 3D

cylinder	Generacja walca
Elipsoid	Generacja elipsoida
fill3	Generacja wielokąta
sphere	Generacja kuli

Przykład:

```
>> sphere % wyświetla sferę
>> [x,y,x]=sphere; % zwraca 3 macierze z danymi do wyrysowania kuli za pomocą funkcji
      mesh lub surf
```

Temat 53
Widoki wykresów 3D

zlabel	Opis osi z
view	Zmiana domyślnego punktu obserwacji
view(azymut, elewacja)	Okresla punkt obserwacyjny za pomocą azymutu i elewacji
view(x,y,z)	Okresla punkt obserwacji w układzie kartezjańskim
view(2)	Obserwacja azymut=0, elewacja=90
view(3)	Domyślny punkt obserwacji: azymut=-37.5 , elewacja= 30

hidden on	Wyświetlanie ukrytych krawędzi
hidden off	Domyślny, ukrywa niewidoczne krawędzie

shading flat	Powierzchnia z dyskretnymi kolorami
shading intern	Powierzchnia z wypełnieniem kolorami interpolowanymi
shading faced	Powierzchnia z dyskretnymi kolorami i siatką

caxis	Przeskalowanie kolorów

Temat 54
Wizualizacja 3D

camlight	Definiuje oświetlenie we współrzędnych kamery
Light	Definiuje obiekt świecący
lightangle	Położenie kamery we współrzędnych sferycznych
lighting	Algorytm liczenia oświetlenia: flat, gouraud, phong, none
material	Określa właściwości odbiciowe materiału: shiny, dull, metal, default

Temat 55
Wizualizacja wolumetryczna

Dane trójwymiarowe możemy przedstawić albo przez wyświetlanie poszczególnych przekrojów, powierzchni o stałej wartości lub przepływów.

coneplot	Pole wektorowe
contourslice	Kontury w werstwach
isosurface	Powierzchnia o stałej wartości (izopowierzchnia)
slice	Płaszczyzna przekroju
streamline	Linie przepływu
streamparticles	Cząstki wraz z liniami przepływu
streamribbon	Wstęgi zgodne z przepływem
streamslice	Przepływ w warstwach lub na powierzchniach
streamtube	Przepływ pokazany za pomocą walcy

PODSTAWY PROGRAMOWANIA: SKRYPTY I FUNKCJE

Temat 56
Tworzenie skryptów

Skrypt jest plikiem tekstowym zawierającym zestaw funkcji i poleceń Matlaba. Pliki skryptowe mają rozszerzenie **.m**.
Pliki skryptowe można tworzyć w każdym edytorze tekstowym. Najwygodniej wykorzystać edytor Matlaba. Dostęp do edytora jest możliwy przez File -> New-> M-file lub przez odpowiednią ikonę.

Temat 57
Opisywanie skryptów

Każdy skrypt powinien mieć krótki opis zawartości i działania. Opis umieszcza się za znakiem %. Ze względów praktycznych opis należy umieszczać za podwójnym znakiem procenta (%%).
Począwszy od Matlaba 7 znak %% oznacza nową fragment kodu.
Znaki %% oraz % są też inaczej traktowane w czasie konwersji skryptu do html-a.
Opis pliku można wywołać w Matlabie przy pomocy polecenia *help nazwa_skyptu.* Za opis pliku traktowane są pierwsze linie komentarza nieprzerwane liniami innego typu.

Przykład:

```
%% To jest test opisu skryptu piewszy_skrypt.m
%% Jestem w skrypcie
% czy widać tę linię?
a=5; % jakaś komenda
% czy widać tę linię
```

Temat 58
Zmienne w skryptach Matlaba

Skrypty do przechowywania zmiennych używają przestrzeni roboczej Matlaba. Z jednej strony nie trzeba definiować mu zmiennych, ale istnieje niebezpieczeństwo użycia i zamazania zmiennych istniejących już w przestrzeni roboczej.

Temat 59
Wypisywanie kroków wykonywanych w skrypcie na ekran.

Analogicznie do poleceń wypisywanych w Oknie Poleceń, polecenia wykonywane w skrypcie dają echo na ekranie. Aby przyśpieszyć pracę skryptów oraz dla zapewnienia uniwersalności (dobry nawyk dla programistów) należy wszystkie polecenia wykonywać z opcją ukrywania echa (o ile celem pliku nie jest narysowanie wykresu). Do ukrywania echa stosuje się średnik na końcu linii polecenia – patrz Temat 7.

Temat 60
Tworzenie funkcji

Funkcja tak jak skrypt jest plikiem tekstowym zawierającym zestaw funkcji i poleceń Matlaba i zaczynać się powinna od słowa kluczowego **function**. Pliki funkcji mają również rozszerzenie **.m**.

UWAGA: Ważne jest aby nazwa funkcji i nazwa pliku były takie same.

Pliki funkcji można tworzyć w każdym edytorze tekstowym. Najwygodniej wykorzystać edytor Matlaba.
Podstawową różnicą miedzy funkcją a skryptem jest sposób przechowywania danych. Skrypt czyni to w przestrzeni roboczej, natomiast funkcja przechowuje je poza przestrzenia roboczą, co pozwala na dublowanie nazw zmiennych z przestrzenią roboczą. Inaczej mówiąc funkcja jest hermetyczna i pokazuje na zewnątrz tylko dane wyjściowe, lub zmienne specjalnie udostępnione przy pomocy operatora global.

Temat 61
Szkielet funkcji

```
function [x,y,z]=nazwa_funkcji(a,b,c,d)
%% [x,y,z]=nazwa_funkcji(a,b,c,d)
%% Funkcja zwraca 3 wektory x,y,z dla danych parametrów wejściowych a,b,c,d

%%Koniec nazwa_funkcji.m
```

Funkcja powinna posiadać następujące elementy:

- nagłówek funkcji – definicje parametrów funkcji - argumentów, (a, b, c, d – w naszym szkielecie) oraz parametrów wyjścia - wartości, (x, y, z – w naszym szkielecie);
- komentarz z opisem do help-u – opisuje co funkcja robi, opisuje argumenty funkcji oraz wartości wyjściowe;
- analiza liczby parametrów wejściowych – moduł funkcji analizuje liczbę parametrów wejściowych, czy jest ich wystarczająco dużo do wykonania funkcji i czy ewentualnie można przyjąć wartości domyślne dla niepodanych parametrów (na razie się tym nie zajmujemy);
- analiza własności parametrów wejściowych – moduł funkcji sprawdza czy wartości wprowadzonych argumentów umożliwiają poprawne wykonanie funkcji (na razie się tym nie zajmujemy);
- implementacja algorytmu – zapewnia poprawność obliczeń numerycznych i przygotowuje wartości wyjściowe.

Temat 62
Postępowanie przy pisaniu funkcji

Najwygodniej najpierw napisać skrypt a po przetestowaniu przerobić go w funkcję.

Temat 63
Przykład funkcji i wywołania funkcji

Zawartość pliku przykład_1.m:

```
function [x,y]=przyklad_1(a,b)
%% [x,y]=przyklad_1(a,b)
%% Funkcja rysuje wykres funkcji y=a*cos(x+(pi/b))
%% zwraca 2 wektory x – wektor zmiennej x, y - wektor z wynikami funkcji
%% dla danych parametrów a, b. Funkcja rysuje wykres funkcji w aktywnym oknie.
x = 0:0.001:2*pi;
y = a.*cos(x+(pi./b));
plot(x,y);
%Koniec przyklad_1.m
```

Wywołanie funkcji z Okna Poleceń:

```
>> przyklad_1(2,2); % rysuje wykres funkcji
>> [ax,ay]=przyklad_1(2,2);  % oprócz wykresu wyprowadza do przestrzeni roboczej dwa
      wektory ax, ay.
>> parametr1=3;, parametr2=4;
>> [ax,ay]=przyklad_1(parametr1, parametr2); % j.w.
```

Temat 64
Pod funkcje

W jednym pliku zawierającym funkcję można umieścić więcej funkcji. Przy czym tylko pierwsza funkcja jest widoczna na zewnątrz. Wszystkie pozostałe funkcje mogą być wywoływane tylko w obrębie danego pliku.

Temat 65
Instrukcja *for*

Instrukcja for pozwala na powtarzanie wybranego fragmentu kodu określoną ilość razy.
Szablon instrukcji for (uwaga na przecinek):

```
....
for zmienna_iterowana = macierz_wartości ,
.....
Kod do wielokrotnego powtarzania
....
end
.....
```

Pętle w wybranych przypadkach można przerywać przy pomocy instrukcji **break**.

Przykład:

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

Temat 66
Instrukcja *while*

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

```
....
while wyrażenie_warunkowe,
.....
Kod do wielokrotnego powtarzania
....
end
.....
```

Przykład:

```
licznik1=0; , licznik2=0; , suma=0; % definicja stałych
while (licznik1<10 & licznik2<10), % znak & oznacza and, opis – help ops
   licznik1=licznik1+0.1;
   licznik2=licznik2+0.2;
   suma=licznik1+licznik2;
end
```

Temat 67
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 wyrażenie_warunkowe_1
   Kod wersja 1
elseif wyrażenie_warunkowe_2
   Kod wersja 2
elseif wyrażenie_warunkowe_3
   Kod wersja 3
......
else
   Kod wersja N
end
```

Przykład:

```
%% y=a*x^2+b*x+c
a=1; , b=2; , c=3; % definicja stałych
wyznacznik=b^2-4*a*c; % np. wyznacznik równania kwadratowego
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
```

Temat 68
Instrukcje *break* i *return*

Obie instrukcje powodują przerwania wykonywania kodu. Funkcja **break** powoduje wyskoczenie z najgłębiej zagnieżdżonej pętli do wyższej pętli. Funkcja **return** powoduje natychmiastowe opuszczenie danej funkcji lub skryptu i powrót do miejsca jej wywołania.

Temat 69
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
```

INNE PRZYDATNE FUNKCJE

Temat 70
Funkcje pomiaru czasu

cputime	Czas CPU który upłynął od uruchomienia Matlaba (ogólnie do pomiaru czasu)
tic	Start stopera
toc	Zatrzymanie stopera
etime	Czas, który upłynął pomiędzy dwoma podanymi datami w formie wektorów
pause	Zatrzymanie na x sekund – zwykle oczekiwanie na odpowiedź użytkownika przy programach interakcyjnych

Temat 71
Testowanie funkcji – czas wykonywania funkcji – *tic* i *toc*

W przypadku testowania programów najwygodniej używać funkcji tic i toc.

Przykład:

```
tic
testowana_Funkcja
toc
```

```
>> Elapsed time is 2.188000 seconds.
```

Temat 72
Funkcje daty i czasu

Funkcje czasu i daty znajdują się w grupie funkcji timefun – help timefun

now	Aktualna data jako liczba dni od 01.01.0
date	Aktualna data i godzina jako zmienna łańcuchowa
clock	Aktualna data i godzina jako wektor

datenum	data jako liczba dni od 01.01.0
datestr	data jako zmienna łańcuchowa
datevec	Transformacja składników daty do postaci wektora

calendar	Kalendarz
weekday	oblicza dzień tygodnia dla podanej daty
eomday	zwraca liczbę dni w miesiącu w podanym roku i miesiącu
datetick	formatowanie daty

Temat 73
Funkcje w Matlabie – ciąg dalszy – zmienne globalne *global*

Przypomnienie: zmienne w funkcji są lokalne – nie widać ich na zewnątrz. Tak samo zmienne w obszarze roboczym są niewidoczne dla funkcji chyba, że są jej parametrem wejściowym. Nawet wtedy jednak są przekazywane przez wartość, także ich wartość modyfikowana wewnątrz funkcji wróci do wartości początkowej po wyjściu z funkcji. Jednak czasami takie ograniczenia nie są wygodne. Gdy chcemy, aby zmienne z przestrzeni roboczej były dostępne wewnątrz funkcji bez definiowania ich jako parametry funkcji, wtedy deklarujemy je jawnie w przestrzeni roboczej oraz w samej funkcji poprzez ***global***. Takie działanie jest jednak niebezpieczne, bo może dojść do konfliktu nazw pomiędzy funkcją i przestrzenią roboczą, lub niepożądaną zmiana ich wartości.

Przykład:

```
function [.....]=fun(....)
%% opis funkcji
global a1 a2 a3;
.....
% koniec funkcji
```

```
%% w przestrzeni roboczej
global a1 a2 a3;
a1=....
a2=.....
a3=.....
```

Temat 74
Funkcje w Matlabie – ciąg dalszy – funkcja *feval*

Często istnieje potrzeba, aby dana funkcja matlabowska (plik *.m) była w stanie przeprowadzić obliczenia dla dowolnych funkcji matematycznych zdefiniowanych poza plikiem *.m. Wtedy stosuje się funkcję *feval*.
Definicja funkcji feval:

```
>> y = feval(Nazwa_funkcji, x1 .....xn) % Nazwa_funkcji  - zmienna łańcuchowa
%% , x1 .....xn – zadane argumenty funkcji
```

Przykład:

```
y= feval('cos',[0:0.01:pi]);
```

Przykład: Funkcja suma_ciagu, która wylicza sumę n wyrazów dowolnego ciągu

```
function s=suma_ciagu(n,ciag)
%% s=suma_ciągu(liczba wyrazów, 'nazwa_funkcji')
i=[1:n];
s=sum(feval('ciag',i)
% koniec funkcji suma_ciagu
```

```
function [a]=ciag(n)
%% [a]=ciag(n) – tu definiuję jak wygląda n-ty wyraz ciągu
a=0.5 .^n
%koniec funkcji ciag
```

Temat 75
Operacje łańcuchowe

```
>> a='to jest lancuch';  % definicja łańcucha i przypisanie go zmiennej a
>> b='drugi';
>> c=strat(a, b);  % połączenie łańcuchów. Zmienna c = `to jest lancuch drugi`
>> d=[a b]; % j.w.

>> t=num2str(15.4); % Zamiana liczby na łańcuch
>> d=str2num(`15.4`); % Zmiana łańcucha na liczbę
```

Temat 76
Rekurencja

Rekurencja jest eleganckim, ale bardzo kosztownym sposobem programowania. Można ją stosować tam gdzie mamy do czynienia z zależnościami typu f(n+1)=g(f(n)).

Przykład:

```
function s=silnia(n)
%% Obliczanie silni metodą rekurencyjną
s=1;
for k=2:n,
   s=k*silnia(k-1); % funkcja wywołuje sama siebie
end
```

ROZWIĄZYWANIE UKŁADÓW RÓWNAŃ NIELINIOWYCH

Temat 77
Rozwiązywanie równań nieliniowych

Przy rozwiązywaniu równań poszukujemy pierwiastków równań, maksimów i minimów funkcji. Pierwiastki rzeczywiste równania, (czyli miejsca zerowe) –> f(x)=0.
W Matlabie funkcja ***fzero*** wyszukuje pierwiastek równania w pobliżu zadanej wartości zmiennej. Czyli do znalezienia wszystkich pierwiastków równania trzeba podać okolice gdzie ma on występować.
Minimum lokalne funkcji poszukuje się analogicznie do pierwiastków przy pomocy funkcji ***fminbnd***.
Maximum lokalne funkcji poszukuje się przez poszukiwanie minimum funkcji odwrotnej do danej, czyli -> -f(x).
W przypadku wielomianów wartości funkcji wyszukuje się poprzez funkcje ***roots(c)*****,** gdzie c jest wektorem współczynników wielomianu.
W celu obliczenia wartości wielomianu korzystamy z funkcji ***polyval(c,x)***, gdzie *x* jest liczbą, wektorem lub macierzą dla której liczymy wartości wielomianu.

Przykład:

```
>> x=fzero('sin',10); % szuka miejsca zerowego funkcji sinus w okolicach 10
>> m=fminbnd('sin',10,11); % szuka najmniejszej wartości funkcji sinus w przedziale (10,11)
>> p=[3,-2,4,1]; % definicja wielomianu p=3x^3-2x^2+4x+1
>> r=roots(p); % zwraca pierwiastki równania
>> w=polyval(p,2); % zwraca wartość wielomianu 3*2^3-2*2^2+4*x+1
```

ROZWIĄZYWANIE UKŁADÓW RÓWNAŃ LINIOWYCH

Temat 78
Rozwiązywanie układów równań liniowych

Matlab ma bardzo rozwinięte algorytmy rozwiązywania równań liniowych. W zależności od potrzeb można używać metod zaawansowanych (Matlab stara się dobrać metodę w tle) lub ręcznie poprzez Metodę Gaussa, uzyskiwanie rozkładu macierzy na macierze trójkątne itd.)

Układ równań linowych można zapisać wektorowo w postaci:
A*x=b, gdzie A macierz współczynników, x – wektor zmiennych [x1...xn], b – wektor wartości równań [b1...bm].
Uwaga: z rozwiązaniem układu równań nie ma problemu pod warunkiem, że układ nie jest sprzeczny, jest dobrze określony, i jest liniowo niezależny. W przeciwnym wypadku trzeba stosować bardziej zaawansowane metody obliczeń.
Do sprawdzania uwarunkowania macierzy służy funkcja ***cond(a)***. Duże wartości funkcji ***cond*** świadczą o złym uwarunkowaniu – to wpływa na dokładność obliczeń numerycznych.

Temat 79
Metody obliczania układów równań

- operator dzielenia lewostronnego: x=A\b – praktycznie jest tu stosowana metoda eliminacji Gaussa z częściowym wyborem elementu głównego
- przez mnożenie wektora wynikowego przez macierz odwrotna współczynników x=inv(A)*b

Przykład:

```
% rozwiązanie układu równań w postaci [a]*[x]=[b]
>> a = [ 1 -4  3; 3  1 -2; 2  1  1]; % definicja macierzy współczynników
>> b = [ -7; 14; 5]; % definicja wektora wyników
>> x = inv(a)*b; % rozwiązanie metodą odwrócenia macierzy
>> x = a\b; rozwiązanie metodą dzielenia lewotronnego
```

Uwaga: równanie x=b/A daje wynik rozwiązania układu równań w postaci x*A=b.

Temat 80
Eliminacja Gaussa

Podstawowa metodą rozwiązywania układów liniowych jest metoda eliminacji Gaussa – tzw. rozkład LU. Polega on na znalezieniu macierzy L i U takich, że A=L*U, gdzie U jest macierzą trójkątną górną, a L macierzą trójkątną dolną.
W Matlabie eliminację Gausa przeprowadza funkcja ***lu***.
Przy wywołaniu [L,U]=lu(A) U jest macierzą trójkątną górną, ale L nie zawsze będzie macierzą trójkątną dolną.
Przy wywołaniu [L,U, P]=lu(A) U jest macierzą trójkątną górną, L nie zawsze będzie macierzą trójkątną dolną, a P macierzą permutacji (zmienia kolejność wierszy w macierzy A). Zachodzi tu zależność L*U=P*A.

Temat 81
Inne funkcje związane z układami równań liniowych

det	wyznacznik macierzy
inv	odwrotność macierzy
eig	wartości własne
chol	rozkład Cholesky'ego, rozkład macierzy A na macierz L i L' takie, że A=L'*L

INTERPOLACJA I APROKSYMACJA FUNKCJI

Temat 82
Interpolacja

Interpolacją nazywamy zadanie znalezienia krzywej przechodzącej przez zadane punkty. Te zadane punkty nazywa się węzłami interpolacji.
W Matlabie stosuje się kilka metod interpolacji: wielomianami pierwszego i trzeciego stopnia, metodą najbliższych sąsiadów oraz za pomocą funkcji sklejanych. Interpolacje stosuje się do tzw. zagęszczania tabel. Np. mamy tabelę z krokiem dla osi x równym 1, a chcemy stworzyć tabelę z krokiem 0.2.

Temat 83
Funkcja *Interp1*

yi=interp1(x, y, xi, 'metoda') gdzie:
x, y – wektory współrzędnych węzłów interpolacji,
xi – wektor punktów na osi X dla których będą obliczane interpolowane wartości yi
metoda:

'linear'	funkcja łamana
'spline'	funkcja sklejana 3-go stopnia
'cubic', 'pchip'	wielomian 3-go stopnia
'nearest'	funkcja najbliższego sąsiedztwa

Przykład:

```
% Interpolacja funkcji sinus, na wykresie węzły zaznaczone są punktami, dodatkowo
% rysowana jest wzorcowa funkcja.
>> x=0:10; y = sin(x); xi = 0:.25:10;
>> yi = interp1(x, y, xi);
>> plot(x, y, 'o', xi, yi, sin(xi))
```

Temat 84
Funkcje interp2, interp3

Funkcje interpolujące w 2 i 3 wymiarach.

Przykład:

```
>> zi=interp2(x,y,z,xi,yi); % x, y, z – dane funkcji, xi, yi – nowe zagęszczone punkty
>> vi=interp3(x,y,z,v,xi,yi,zi); % x, y, z, v – dane funkcji, xi, yi, zi – nowe zagęszczone punkty
```

Temat 85
Aproksymacja – funkcja *polyfit*

Aproksymacja oznacza przybliżanie tzn. zastępowanie jednych wartości innymi, wygodniejszymi, z jakich względów. Matlab pozwala na aproksymację wielomianem.

Przykład:

```
p=polyfit(x,y,r); % x, y – serie danych, r – zadany stopień wielomianu przybliżającego
```

PODSTAWY STATYSTYKI W MATLABIE

Temat 86
Funkcje Statystyczne

Dostęp do opisu funkcji statystycznych: *help datafun*

max	Element maksymalny
min	Element minimalny
mean	Wartość średnia
median	Wartość medialna
std	Odchylenie standardowe
var	Wariancja
sort	Sortowanie kolumn wg różnych wartości
sortrows	Sortowanie kolumn wg różnych wartości
sum	Sumowanie elementów
prod	Mnożenie elementów
hist	Histogram
histc	Histogram ważony
cumsum	Zwraca wektor kolejnych skumulowanych sum elementów
cumprod	Zwraca wektor kolejnych skumulowanych iloczynów elementów
corrcoef	Współczynniki korelacji
cov	Macierz kowariancji