Wprowadzenie do MATLAB

Małgorzata Strycharz , Magdalena Kwiatkowska

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki Automatyki Informatyki i Elektroniki
kierunek: Informatyka
rok II B grupa 6

http://student.uci.agh.edu.pl/~strychar/mownit.html
http://student.uci.agh.edu.pl/~magkwiat/mownit.html

1 Wiadomości wstępne

1.1 Krótka charakterystyka

MATLAB stanowi uniwersalne, interakcyjne środowisko programowe oraz jednocześnie wysokiego poziomu język programowania. Przeznaczeniem pakietu są przede wszystkim obliczenia naukowo - techniczne, inżynierskie oraz wizualizacje. Program ten pozwala na wykonywanie skomplikowanych obliczeń numerycznych z końcową wizualizacją otrzymanych wyników. Są one dostępne natychmiast, a pakiet oferuje możliwość przedstawienia ich w postaci dwu lub trzech wymiarowych wykresów.
MATLAB łączy analizę numeryczna, obliczenia macierzowe, przetwarzanie sygnałów i grafikę w łatwe do użycia środowisko, w którym zarówno problemy jak i ich rozwiązania zapisane są matematycznie bez uwzględnienia zasad tradycyjnego programowania.

1.2 Trochę historii

Nazwa MATLAB pochodzi od MATrix LABoratory i wiąże się z pierwotna koncepcja pakietu, polegającą na udostępnieniu użytkownikowi procedur LINPACK-a i EISPACK-a przy pomocy prostego interakcyjnego języka poleceń. Pierwsza wersja MATLAB-a została napisana w Fortranie w 1980. Jej autorem jest C.Moler. W piec lat później firma MathWorks Inc. wprowadziła na rynek wersje komercyjną, napisana w języku C. Wersja ta do dzisiaj została znacznie rozszerzona i obejmuje różne platformy sprzętowe począwszy od PC, a skończywszy na superkomputerach. Jeśli przetłumaczyć by pełną nazwę pakietu na język polski - „Laboratorium Macierzowe" oczywistym staje się to, iż użytkownik operuje tylko na jednym typie danych, a mianowicie na macierzach. Pociąga to za sobą fakt, że nawet pojedyncze liczby są reprezentowane w formie jednowymiarowej macierzy kwadratowej.

1.3 Dziedziny zastosowań

Ze względu na bardzo szerokie możliwości, zakres zastosowań pakietu obejmuje bardzo różnorodne dziedziny nauki i techniki. Można tu wymienić choćby elektronikę, telekomunikację, automatykę, ekonomię, ale również zagadnienia z meteorologii, biologii i medycyny. W dziedzinach algorytmów numerycznych algebry liniowej, analizy matematycznej i numerycznej, czy całkowania numerycznego i oczywiście rachunku macierzowego bardzo ważną role odgrywa dostęp do ich najnowszych implementacji, jest to jedna z bardzo znaczących zalet MATLAB'a. Przyjazny użytkownikowi interfejs (okienka, przyciski, menu) znacznie ułatwiają pracę z pakietem. Natomiast rozszerzalność pakietu przez importowanie własnych aplikacji napisanych w języku C lub Fortranie stanowi ważny i pożyteczny model pracy z pakietem.

2 Podstawowe zasady programowania

Praca w środowisku MATLAB-a przypomina pracę w typowym systemie operacyjnym. Polega na wydawaniu poleceń, które po zatwierdzeniu są wykonywane przez interpreter.

- obowiązuje zasada indeksowania macierzy począwszy od 1
- inicjalizacja zmiennych odbywa się automatycznie, wtedy, gdy po raz pierwszy przypisuje się im wartość
- stałe tekstowe zapisuje się w apostrofach
- wyróżniona jest predefiniowana zmienna *ans* - przyjmująca wynik ostatniej operacji, dla której nie określono zmiennej wynikowej
- w każdej chwili można zobaczyć listę zmiennych zdefiniowanych przez użytkownika przy użyciu komendy *who*
- usuniecie zmiennej o podanej nazwie następuje po wywołaniu komendy *clear* **nazwa-zmiennej**, gdy nie podano nazwy zmiennej to usuwane są wszystkie do tej pory zdefiniowane zmienne
- nazwy zmiennych powinny zaczynać się od dowolnej litery, po niej mogą nastąpić litery, cyfry lub podkreslenia w dowolnej ilości, ale rozróżniane jest tylko pierwsze 19 znaków
- duże i małe litery są rozróżniane
- funkcja przestaje być dostepna, jeśli utworzymy zmienną o nazwie identycznej jak nazwa tej funkcji, jest dostępna spowrotem przy ponownym uruchomieniu programu
- zawartość aktualnego katalogu uzyskujemy poleceniem *dir* **ścieżka-i-znaki-masek**
- do zmiany aktualnego katalogu służy polecenie *chdir* **nowy-katalog**
- wszystkie zmienne można zapisać na dysku poleceniem *save* **nazwa-pliku**
- odczytanie zapisanych danych umożliwia polecenie: *load* **nazwa-pliku**
- jeżeli po poleceniu wystepuje średnik, to wartość będąca wynikiem jego wykonania nie będzie wyświetlona na ekranie

3 Macierze i operacje na macierzach

3.1 Sposób przechowywania w pamięci

Macierze są podstawowym typem danych wykorzystywanym w MATLAB-ie. Dopuszcza on dwa sposoby przechowywania macierzy w pamięci:

- *gęsty* - przydzielana jest pamięć dla tablicy $n \times m$ liczb rzeczywistych
- *rzadki* - pamietane są tylko elementy niezerowe wraz z ich położeniem

3.2 Definiowanie macierzy

Najprostszą macierz tworzymy wymieniając jej elementy w nawiasach kwadratowych.
Przykład:

```
>>M=[1 2 .3 5 ; 5.1 3 2 1/3]
```

W efekcie otrzymamy macierz następującej postaci:

```
M=
    1.0000  2.0000  0.3000  5.0000
    5.1000  3.0000  2.0000  0.3333
```

Macierze można tworzyć także za pomocą funkcji bibliotecznych np.
eye(n) - daje w efekcie macierz jednostkową n-wymiarową
rand(m,n) - utworzy macierz mającą m wierszy i n kolumn oraz wypełni ją liczbami losowymi z przedziału <0,1>
ones(m,n) - utworzy macierz o podanych wymiarach i zainicjalizuje ją jedynkami
zeros(m,n) - utworzy macierz o podanych wymiarach i zainicjalizuje ją zerami

3.3 Dostęp do elementów macierzy

Do elementu macierzy możemy się odnieść podając nazwę macierzy oraz w nawiasach numery wiersza i kolumny.
Przykład:

```
>>M(1,1)=1;
```

Jeżeli wykroczymy poza wymiary macierzy, to automatycznie zostanie jej nadany odpowiedni wymiar, a brakujące pola zostaną zainicjalizowane zerami.
Przykład:

```
>>M(2,5)=3

M =
   1.0000  2.0000  0.3000  5.0000  3.0000
   5.1000  3.0000  2.0000  0.3333       0
```

3.4 Główne operatory macierzowe

Na macierzach można dokonywać następujących działań przy pomocy operatorów:

- dodawanie dwóch macierzy lub dodanie skalaru do macierzy

Przykład:

```
>>A+1;
>>A+B;
```

- odejmowanie macierzy lub odjęcie skalaru od macierzy
- mnożenie macierzy lub skalarów (przy możeniu macierzy A*B liczba kolumn macierzy A musi być równa liczbie wierszy macierzy B, w przeciwnym razie generowany jest komunikat o błędzie)

Przykład:

```
>> A=[1 1;2 2;3 3]\\
A=
   1.0000 1.0000
   2.0000 2.0000

>> B=[1;2]
B=
   1.0000
   2.0000

>>A*B
ans =
   3.0000
   6.0000
   9.0000
```

– potęgowanie macierzy

Przykład:

```
>>B^2;
```

3.5 Inne operacje na macierzach

Oprócz podstawowych operacji arytmetycznych, na macierzach można dokonywać szeregu innych działań.
Przykłady:

– mnożenie tablicowe macierzy o tych samych wymiarach - czyli mnożenie odpowiadających sobie elementów dwóch macierzy

```
>>A.*B
```

– potęgowanie tablicowe - podniesienie każdego elementu macierzy do podanej potęgi

```
>>A.^3
```

– użycie dwukropka
A(:) - przestawia wszystkie elementy macierzy w wektor kolumnowy
A(:,i) A(i,:) - wypisuje odpowiednio i-tą kolumnę lub i-ty wiersz macierzy A
A(i:n) - wypisuje elementy macierzy od i-tego do n-tego
A(:,i:n) - wypisuje kolumny począwszy od i-tej do n-tej
A(i:n,:) - wypisuje elementy od i-tego do n-tego wiersza

3.6 Funkcje przekształcające macierze

Obok operatorów tablicowych MATLAB zawiera także funkcje tablicowe, które przekształcają każdy z elementów macierzy z osobna.
Przykład:

```
>>A=[1 2 4; 2 16 64];sqrt(A)
ans=
    1.0000 1.4142 2.0000
    1.4142 4.0000 8.0000
```

Elementami macierzy wynikowej są pierwiastki stopnia drugiego elementów macierzy A. Funkcje tablicowe można podzielić na cztery grupy:

– *trygonometryczne, hiperboliczne i odwrotne do nich* np **sin(A)** - sinus, **atan(A)** - arcus tangens, **cosh(A)** - cosinus hiperboliczny, **asinh(A)** - arcus sinus hiperboliczny

– *funkcje logarytmiczne, wykładnicze i potęgowe* np. **exp(A)** - funkcja wykładnicza **pow2(A)** - oblicza wartości potęgi liczby 2 dla wykładników będących elementami macierzy A, **log(A)** - logarytm naturalny elementów macierzy A

– *funkcje związane z obliczeniami w dziedzinie liczb zespolonych* np. **abs(A)** - macierz modułów elementów macierzy A, **real(A)** - macierz części rzeczywistych elementów macierzy A, **imag(A)** - macierz części urojonych elementów macierzy A

– *funkcje zaokrągleń i reszty z dzielenia* **ceil(A)** - zaokrągla elementy macierzy A w górę, **floor(A)** - zaokrągla elementy macierzy A w dół, **round(A)** - zaokrągla elementy macierzy do najbliższej liczby całkowitej, **rem(A,B)** - oblicza resztę z dzielenia odpowiadających sobie elementów macierzy A i B

4 Skrypty

W oparciu o doświadczenie programowania, Matlab daje użytkownikowi możliwość by pewne sekwencje instrukcji wykonywanych wielokrotnie i często wykorzystywanych umieszczać w skryptach. Dzięki temu coś zaprogramowane raz, może być następnie wykorzystywane wielokrotnie tylko przez odpowiednie wywołanie bloku. Skrypty stanowią narzędzie za pomocą którego w łatwy i szybki sposób można przejść od zadania do rezultatu. Skrypt, to po prostu zbiór tekstowy zawierający instrukcje (polecenia), które mają być wykorzystywane przez interpreter. Standardowym rozszerzeniem dla plików skryptów jest rozszerzenie ".m" . Plik ten nie musi spełniać żadnych dodatkowych wymogów formalnych, po prostu zapisane w nim polecenia muszą zachowywać poprawność składniową i semantyczną.

Przykład prostego skryptu wyznaczającego w 101 punktach wartości funkcji $x = \cos(t) + \sin(t)$ i kreślący jej wykres na przedziale $<0, 2\pi>$

```
t=[0:0.01:2*pi];
A=[1 1];
x=A*[cos(t); sin(t)];
plot(t, x)
```

Komentarze w skryptach to bardzo użyteczny sposób tworzenia pomocy związanej z danym skryptem. Pierwsze trzy linie komentarza zostaną wyświetlone na ekranie po wywołaniu polecenia *help* z nazwą pliku skryptu. Jest to pewnego rodzaju informacja o tym co dokładnie znajduje się w skrypcie i jakie właściwie funkcje spełnia.

5 Funkcje

Jedną z własności MATLAB-a jest jego proceduralność. Cechę tę warunkuje możliwość tworzenia własnych funkcji. Zatem Matlab jest programem, który nie tylko daje użytkownikom możliwości wykorzystywania funkcji napisanych przez innych (biblioteki i funkcje standardowe), ale pozwala na samodzielne ich tworzenie. By zdefiniować funkcję, należy podobnie jak przy skryptach umieścić ją w pliku z rozszerzeniem ".m". Należy jednak pamiętać, by nazwa tego pliku była identyczna z nazwą definiowanej funkcji. Jest to wynikiem tego, iż po wykryciu nazwy funkcji pakiet rozpoczyna przeszukiwanie po nazwach dostępnej listy plików. Podobne, jak w skryptach, zastosowanie komentarza umożliwi wyświetlenie informacji o funkcji poleceniem *help*.
Ogólna postać definicji funkcji:

function *wartości-funkcji* = *nazwa-funkcji*(parametr1,..., parametrN)
komentarz-dokumentujący
ciąg-instrukcji

Zmienne używane w ciele funkcji są zmiennymi lokalnymi, są utworzone podczas jej działania i usuwane z chwilą zakończenia wykonywania funkcji. Aby dana zmienna istniejąca we wspólnej pamięci stała się dostępna, trzeba zadeklarować ją jako globalną (poleceniem *global*), a informacje o tym trzeba powtórzyć w ciele funkcji. Argumenty funkcji Matlab-a są przekazywane przez wartość. Wiąże się to z tym, iż po wywołaniu funkcji z parametrami wejściowymi, operacje wykonywane będą na ich kopiach. Istnieją w Matlab-ie zmienne standardowe *nargin* oraz *nargout*, zawierają one liczbę przekazywanych parametrów, odpowiednio wejściowych i wyjściowych. Właśnie na tych zmiennych może opierać się sposób wywołania danej funkcji.
Przykład funkcji *suma* - pobiera ona dwa parametry i zwraca ich sumę:

```
function z=suma(a, b)
%z = x+y;
%dodaje liczby, wektory, macierze, itp.
z = x+y;
```

6 Instrukcje

6.1 Operatory

Podstawowe operatory arytmetyczne to:
+ operator dodawania
- operator odejmowania
∗ operator mnożenia
\ - operator dzielenia

Operatory logiczne i ich znaczenie:
and - logiczne i
not - logiczne nie
or - logiczne lub
xor - logiczne xor
any - wieloargumentowy or
all - wieloargumentowy and

Operatory relacji:
eq - równe
ne - różne
lt - mniejsze niż
gt - większe niż
le - mniejsze lub równe
ge - większe lub równe

Podane operatory są w większości dwuargumentowe. Jednoargumentowymi operatorami są **any** i **all** i w ich przypadku operacje logiczne odbywają się na wektorach. Wyrażenia są wartościowane tak, że wartość **false** jest reprezentowana przez 0, natomiast wartość **true** przez pozostałe liczby.
Przykłady:

```
>>xor([1 3] [2 0])
ans =
    0 1

>>any([-2 0 0 0])
ans =
    1

>>all([1 -3 2 ; 1 0 3 ; 1 2 3])
ans =
    1 0 1

>>[2 3 ; 1 1] lt [0 1 ; 3 4]
ans =
    0 0
    1 1
```

6.2 Instrukcja warunkowa

Instrukcja warunkowa MATLAB-a ma postać:

if wyrażenie-warunkowe1
ciąg-instrukcji1 **elseif** wyrażenie-warunkowe2
ciąg-instrukcji2
...
else
ciąg-instrukcjiN
end

Wykonanie tej instrukcji polega na wykonaniu *ciągu instrukcji* związanego z *wyrazeniem warunkowym*, którego wartość jest macierzą o samych elementach niezerowych, co ma wartośc logicznej prawdy w MATLAB-ie. Jeśli nie zachodzi żaden z tych warunkow, wykonywany jest ciąg instrukcji po słowie kluczowym **else**. Sekwencje **elseif**... i **else**... są opcjonalne.

6.3 Instrukcje iteracyjne (pętle)

Instrukcja **for**
Ogólna postać instrukcji iteracyjnej **for** przedstawia się następująco:

for zmienna-iterowana=macierz-wartości,
ciąg-instrukcji
end

Działanie tej instrukcji to wykonywanie *ciągu-instrukcji* dla kolejnych wartości *zmiennej-iterowanej*. *Zmienna iterowana* przyjmuje wartosci równe wektorom kolumnowym pobieranym kolejno z *macierzy-wartości*.

Przykład:

```
for i=1:N
  for j=1:M
    A(i,j)=sqrt(i/j);
  end
end
```

Wartością wyrażenia 1:N (podobnie 1:M) jest wektor wierszowy postaci [1, 2, 3, ...,N].

Instrukcja **while**
Powyższa instrukcja ma następującą postać:

while
wyrażenie-warunkowe,
ciąg-instrukcji,
end

Jej wykonanie powoduje wykonywanie *ciągu-instrukcji* do momentu, w którym wartość *wyrażenia-warunkowego* ma wartość logiczną prawda, czyli macierz, która jest wartością wyrażenia warunkowego ma wszystkie elementy niezerowe.

6.4 Instrukcje return, break

Instrukcja **break** powoduje przerwanie wykonywania pętli, przy czym opuszczany jest tylko jeden poziom zagłębienia pętli. Instrukcja **return** powoduje bezwarunkowe opuszczenie danej funkcji lub skryptu i powrót do miejsca jego/jej wywołania.

6.5 Obsługa wyjatków

Istnieje mechanizm, wzorowany na obowiązującym w językach obiektowych, służący do wychwytywania wyjątków. Postać bloku jest następująca:

try,instr1,instr2,...,instrN,
catch instr1,instr2,...,instrN,
end

Jego obecność zabezpiecza przed skutkami podania np. złego typu parametru itp.

7 Wybrane metody numeryczne w MATLAB-ie

7.1 Interpolacja

Procedury MATLAB-a realizują interpolację za pomocą następujących metod:

- interpolacja wielomianami pierwszego i trzeciego stopnia
- interpolacja za pomocą funkcji sklejanych

Dużą rodzinę metod interpolacji stanowi interpolacja za pomocą wielomianów. Powszechnie znanym i wykorzystywanym jest wielomian interpolacyjny Lagrange'a. Podstawową wadą interpolacji za pomocą wielomianów jest skłonność do występowania silnych oscylacji między węzłami interpolacji dla wielomianów interpolacyjnych wysokich stopni. Rozwiązaniem tego problemu może być interpolacja za pomocą funkcji sklejanych.

Funkcja **interp1**
Postać wywołania: $yi = interp1(x, y, xi, 'metoda')$

Wykonuje ona interpolację funkcji jednej zmiennej w punktach, które określa wektor xi. Węzły interpolacji określone są parametrami x i y. Parametr opcjonalny umożliwia wybranie metody interpolacji.

Dostępne metody:

- 'linear' - interpolacja przy pomocy funkcji łamanej
- 'spline' - interpolacja funkcjami sklejanymi trzeciego stopnia
- 'cubic' - interpolacja wielomianami trzeciego stopnia

7.2 Aproksymacja wielomianowa

Do stosowania tej metody aproksymacji w pakiecie MATLAB służy funkcja biblioteczna **polyfit**, której wywołanie ma postać:
$a = polyfit(x, y, r)$

x,y - wektory danych
r - stopień wielomianu

Funkcja ta dla wektorów danych x i y znajduje współczynniki wielomianu stopnia r przybliżającego najlepiej w sensie średniokwadratowym zależność między serią danych x a y.

7.3 Generowanie liczb pseudolosowych

Standardową możliwością MATLAB-a jest generowanie liczb pseudolosowych o rozkładzie jednostajnym i normalnym. Służą do tego funkcje **rand** i **randn**.
Funkcja **rand** generuje liczby losowe o rozkładzie jednostajnym na przedziale <0,1>, a **randn** - liczby pseudolosowe o rozkładzie normalnym o średniej 0 i wariancji 1.
Przykladowe postacie wywołań:

$rand(), randn()$ - zwracają liczbę pseudolosową
$rand(n, m), randn(n, m)$ - zwracają macierz $n \times m$ zawierającą liczby pseudolosowe.

7.4 Wielomiany i miejsca zerowe

Poniżej zaprezentowano kilka funkcji MATLAB-a służących do znajdowania zer funkcji jednej zmiennej i pozwalających wygodnie operować na wielomianach.

$x = fzero('F', x0, eps, w)$

Ta funkcja oblicza miejsce zerowe funkcji o podanej nazwie. Zaczyna obliczenia od punktu startowego x0. Parametr *eps* jest opcjonalny i okresla zadaną dokładność, a parametr *w* gdy ma wartość niezerową, powoduje wyświetlanie pośrednich rezultatów na ekranie.

$a = poly(r)$

Wyznacza współczynniki wielomianu o pierwiastkach podanych w wektorze r uszeregowane wg malejących potęg zmiennej x.

$r = roots(a)$

Oblicza pierwiastki wielomianu W(x,a), którego wspólczynniki są podane w wektorze a i uszeregowane wedlug malejących potęg zmiennej x.

7.5 Całkowanie numeryczne

MATLAB udostepnia dwie funkcje realizujące numeryczne całkowanie oparte o dwie nieco różne, choć o zbliżonych podstawach metody:
quad - adaptacyjną kwadraturę opartą o regułę Simpsona
quad8 - adaptacyjną kwadraturę ośmioprzedziałową Newtona-Cotesa.
Pełne postacie wywołań:

$Q = quad(f, a, b, tol, trace)$
$Q = quad8(f, a, b, tol, trace)$

f- łańcuch zawierający nazwę całkowanej funkcji, która musi być umieszczona w odpowiednim skrypcie
a,b - liczby rzeczywiste określające przedział całkowania
tol - wymagana tolarancja względna
trace - opcjonalny parametr - jeśli jest nim macierz o wszystkich elementach niezerowych, to kreślony jest wykres przedstawiający wszystkie węzły kwadratury.

8 Grafika 2D w MATLAB-ie

Istnieje możliwość generowania dokładnych wykresów. Służą do tego funkcje biblioteczne, których część jest omówiona poniżej:

plot - służy do graficznej prezentacji danych matematycznych

Przykład:

```
x=-pi:0.1:pi;
y=sin(x);
plot(x,y)
```

8.1 Funkcja plot

Powyższe instrukcje generują nam wykres funkcji sinus w przedziale <-π,π>
Funkcja **plot** automatycznie ustawia skalę, w której rysowany jest wykres.
Przykładowe sposoby wywołań:

- **plot(y)** - wywoływana z jednym parametrem - wektorem y - rysuje wykres ciągu elementów wektora względem ich indeksów, to znaczy elementy wektora określają współrzędne y poszczególnych punktów, a współrzędne x są kolejnymi liczbami naturalnymi
- **plot(x y)** - wywołana z dwoma parametrami - wektorami x i y - rysuje wykres ciągu elementów drugiego wektora wzgledem pierwszego
- **plot(x,y,s)** - trzeci agrument jest stringiem oznaczającym sposób formatowania linii wykresu np. **plot([1 2],[1 2],'b*-')** - wykres w kolorze niebieskim z gwiazdkami w punktach, linia ciągła.

W następujący sposób można przy pomocy funkcji **plot** narysować wiele wykresów jednocześnie:
plot(x1,y1,x2,y2,...) plot(x1,y1,s1,x2,y2,s2,...)

8.2 Krótki przegląd innych funkcji

Funkcja **subplot**
subplot(m, n, p) - wywołana z trzema parametrami liczbowymi dzieli aktywny rysunek na m w poziomie i n w pionie części i w każdej z nich umieszcza nowy układ współrzędnych oraz uaktywnia p-ty z utworzonych układów. Parametry m i n muszą być liczbami naturalnymi z przedzialu od 1 do 9.

Funkcja **axis**
axis([xmin xmax ymin ymax]) - wymaga jednego parametru - czteroelementowego wektora, którego kolejne elementy określają zakresy skal na poszczególnych osiach układu współrzędnych.

Funkcje **xlabel** i **ylabel**
xlabel(tekst1)
ylabel(tekst2)
Funkcje wypisują łąńcuch znaków tekst1 pod osią x aktywnego układu współrzędnych, a łańcuch tekst2 obok osi y aktywnego układu współrzędnych.

Funkcja **text**
text(x,y,tekst) - wypisuje podany jako parametr tekst łańcuch znaków w aktywnym układzie współrzędnych, w miejscu określonym przez parametry x i y.

Funkcja **grid**
grid
grid on
grid off
Polecenie **grid on** powoduje naniesienie na wykres pomocniczej siatki współrzędnych. Polecenie **grid off** chowa siatkę. Polecenie bez parametrów powoduje przełączenie wyświetlania siatki.

Funkcja **loglog**
Rysuje wykresy używając skal logarytmicznych na obu osiach. Sposoby wywołania są takie jak funkcji **plot**, czyli np.
loglog(y)
loglog(x,y)
loglog(x,y,s)
loglog(x1,y1,x2,y2,...)
loglog(x1,y1,s1,x2,y2,s2,...)

Funkcja **fplot**
Typy wywołania:
fplot(f,granice)
fplot(f,granice,n)
Procedura rysuje wykres funkcji o nazwie określonej przez parametr *f*. Punkty, w których należy obliczyć wartość rysowanej funkcji, są dobierane automatycznie tak, aby wykres był dokładny, uwzględniał dynamikę zmian wartości funkcji, ale również, aby nie wymagał nadmiernej ilości obliczeń. Znaczenie parametrów funkcji:
f-łańcuch znaków stanowiący nazwę pliku zawierającego rysowaną funkcję
granice - dwuelementowy wektor opisujący granice przedziału, w jakim ma być narysowany wektor
n - opcjonalny - liczba określająca minimalną liczbę punktów, uwzględnianych przy sporządzaniu wykresu (domyślnie 25)

9 Grafika 3D w MATLAB-ie

Grafika trójwymiarowa jest rozumiana jako przedstawienie na płaszczyźnie ekranu monitora rzutów figur trójwymiarowych. Program MATLAB zawiera procedury obsługi grafiki typu 3D. Rzutowanie elementów figur trójwymiarowych na płaszczyznę ekranu oraz zasłanianie niewidocznuch krawędzi wykonuje system. Standardowo figury rzutowane są prostopadle. Poniżej wspomniane są niektóre funkcje służace do rysowania wykresów w przestrzeni trójwymiarowej.

Funkcja **plot3**
Sposoby wywołania:

- **plot3(x,y,z)**
- **plot3(x,y,z,s)**
- **plot3(x1,y1,z1,s1,x2,y2,z2,s2,...)**

Funkcja **plot3** stanowi trójwymiarowy analog funkcji **plot**. Wywołana z argumentami wektorowymi x,y i z jako parametrami, funkcja rysuje łamaną łącząc punkty, których współrzędne stanowią odpowiadające sobie elementy tych wektorów. Jeżeli x,y i z są macierzami o identycznych wymiarach rysowana jest jedna linia dla każdej kolumny. Parametr s pozwala określić kolor i typ linii według zasad przedstawionych przy prezentacji funkcji **plot**. Wiele linii na jednym wykresie można uzyskać wywołując funkcję z kilkoma trójkami lub czwórkami parametrów.

Funkcja **mesh**
Funkcja służy do tworzenia wykresów powierzchni.
Typy wywołania:

- **mesh(x,y,z,c)**
- **mesh(x,y,z)**
- **mesh(z,c)**
- **mesh(z)**

Procedura ta rysuje powierzchnię opisaną przez macierze x,y i z w postaci kolorowej siatki o oczkach wypełnionych kolorem tła. Elementy macierzy c określają kolory obwódek poszczególnych oczek. Elementy macierzy c są przeskalowywane w taki sposób, aby była wykorzystana cała mapa. Jeżeli macierz c zostanie pominięta, to przyjmowane jest c=z, czyli kolor obwódek zależy od współrzędnych ich końców.

Funkcja **surf**
Rysuje różnokolorową powierzchnię opisaną macierzami x,y i z. Parametry funkcji mają identyczne znaczenie jak parametry funkcji **mesh** z tą różnicą, że parametr c określa nie kolory "nici" siatki, lecz kolory, jakimi wypełnione są oczka. Funkcję **surf** wywołujemy:

- **surf(x,y,z,c)**
- **surf(x,y,z)**
- **surf(z,c)**
- **surf(z)**

10 Zakończenie

MATLAB-a może używać praktycznie każdy - zarówno zwolennik systemu Microsoft Windows, jak i UNIX-a. Na terenie placówek naukowych takich jak np ACK Cyfronet w Krakowie jest dostępny na odpowiednich serwerach.

Literatura

A.Zalewski, R.Cegieła.
„MATLAB - obliczenia numeryczne i ich zastosowania", Wydawnictwo NAKOM, Poznań 1999

J.Brzózka, L.Dorobczyński
„Programowanie w MATLAB", Wydawnictwo MIKOM, Warszawa 1998

B.Mrozek, Z.Mrozek.
„MATLAB, uniwersalne środowisko do obliczeń naukowo technicznych", Wydawnictwo PLJ, Warszawa 1996