MATLAB
Podstawowe polecenia

W MATLABie możliwe jest wykonywanie prostych obliczeń matematycznych. Działania (np. 1+1) należy wpisać w okienku poleceń na końcu naciskając klawisz „**enter**". Program MATLAB wydrukuje wynik obliczenia poniżej.

```
>> 1+10^2

ans =

   101

>> log(12)+sqrt(25)

ans =

    7.4849

>> factorial(123)

ans =

  1.2146e+205
```

MATLAB ma bardzo rozbudowaną pomoc, którą możemy uzyskać wybierając w menu pozycję Help (F1). Jeżeli chcemy sprawdzić, jakie funkcje matematyczne są dostępne dla użytkownika, to możemy skorzystać ze spisu treści pomocy (np. wybrać pozycję MATLAB → Mathematics → …) lub wyszukać w oknie pomocy polecenie samemu.

Przydatna jest możliwość definiowania zmiennych i wykonywania działań przy ich pomocy. Nazwy zmiennych (a także funkcji i macierzy) mogą być niemal dowolnie długie, ale nie mogą zawierać spacji. W najprostszy sposób zmienne definiuje się poprzez przypisanie im wartości.

```
>> a=1
a =
     1

>> b=2;
>> a+b*a^(1/b);
>> a+b*a^(1/b)
ans =
     3
```

Program wyświetla przypisaną zmiennej wartość, chyba że wiersz zakończony jest średnikiem. Wystąpienie średnika powoduje, że program nie wyświetla wyniku, ale wykonuje działanie w pamięci.

Możliwe jest też przeprowadzanie działań na macierzach. Zdefiniowanie macierzy danych wymaga użycia nawiasów kwadratowych lub zdefiniowania rozmiarów macierzy w nawiasach kwadratowych (np. `x[2,3]`). Spacje oddzielają wartości w wierszu, a średniki oddzielają kolejne wiersze.

```
>> aa = [1 2 3; 4 5 6];
>> aa
aa =
     1     2     3
     4     5     6

>> bb = [a b 10; 3 b a]
bb =
     1     2    10
     3     2     1
```

Można także przypisywać elementom macierzy wartości zdefiniowanych wcześniej zmiennych (tutaj: `a` i `b`). Przypisywanie wartości elementom macierzy jest jednak często kłopotliwe i czasochłonne.

Dlatego istnieje szereg funkcji, które ułatwiają przypisywanie wartości macierzom.

```
>> jedynki = ones(2,4);       %macierz 2x4 z samych jedynek
>> zera = zeros(1,2);         %macierz 1x2 z samych zer
>> sekwencja = 1:1:5;         %wektor poziomy 1x5 przyjmujący
                              %wartości ciągu arytmetycznego od 1 do 5
>> sekwencja = 1:5:10         %ale...
sekwencja =
     1     6

>> qq = eye(4);               %macierz jednostkowa o wymiarach 4x4
>> losowe = rand(2,4);        %macierz 2x4 liczb pseudo-losowych z rozkładu
                              %jednostajnego U(0,1)
>> losowe_n = randn(5,2);     %macierz 5x2 liczb pseudo-losowych z rozkładu
                              %normalnego N(0,1);
>> B = repmat(aa,2,3)         %powiela macierz w wierszach i kolumnach
B =
     1     2     3     1     2     3     1     2     3
     4     5     6     4     5     6     4     5     6
     1     2     3     1     2     3     1     2     3
     4     5     6     4     5     6     4     5     6
```

Parametry funkcji `ones`, `zeros`, `eye`, `rand`, `randn` zapisane w okrągłych nawiasach oznaczają odpowiednie rozmiary macierzy.

Symbolem `%` oddzielone są komentarze do działań. Komentarza program nie traktuje jak polecenia i nie próbuje go wykonywać. Komentarz obowiązuje do końca linijki (do symbolu przejścia do następnego wiersza).

Sposób zapisywania działań na macierzach jest intuicyjny, a samo wykonywanie działań jest bardzo szybkie. Dostępne są operatory sumowania (`+`), odejmowania (`-`), mnożenia (`*`), dzielenia przez skalar (`/`), potęgowania (`^`), mnożenia i dzielenia macierzy element po elemencie (odpowiednio `.*` i `./`), transponowania (`'`) i inne.

```
>> aa+bb
ans =
     2     4    13
     7     7     7
>> aa./bb
ans =
    1.0000    1.0000    0.3000
    1.3333    2.5000    6.0000
```

```
>> aa.^bb                %do potęgowania elementu przez element potrzeba .^
ans =
           1           4       59049
          64          25           6
>> kron(aa,bb)
```

Jednak silnia (`factorial`), czy iloczyn Kroneckera (`kron(aa,bb)`) wymagają użycia funkcji. Możliwe jest łączenie pionowe i poziome macierzy.

```
>> [aa bb]
ans =
     1     2     3     1     2    10
     4     5     6     3     2     1

>> [aa; bb]
ans =
     1     2     3
     4     5     6
     1     2    10
     3     2     1
```

Odwołania do poszczególnych elementów macierzy dokonuje się wykorzystując nawiasy okrągłe. W komentarzach przedstawiono znaczenie poszczególnych poleceń.

```
>> aa(2,2)            %element z 2. wiersza i 2. kolumny
ans =
     5

>> aa(:,[2 3])        %elementy ze wszystkich wierszy i kolumn 2 i 3
ans =
     2     3
     5     6

>> aa(1,1:3)          %elementy z 1. wiersza i kolumn od 1 do 3
ans =
     1     2     3
```

Często potrzebna jest też wiedza na temat liczby wierszy i kolumn macierzy.

```
>> size(aa)           %liczba wierszy i kolumn
ans =
     2     3

>> size(aa,2);        %liczba wierszy i kolumn
>> size(aa,1);        %liczba wierszy i kolumn
```

Inne ważne funkcje często użyteczne przy budowaniu procedur i symulacji ekonometrycznych przedstawiono poniżej.

```
>> diag(eye(2))         %wektor elementów diagonalnych macierzy eye(2)
ans =
     1
     1
>> reshape(aa,1,6)      %przekształca i zmienia wymiary macierzy
ans =
     1     4     2     5     3     6
```

```
>> inv(v);              %odwrotność macierzy v
>> det(v);              %wyznacznik z macierzy v

>> cov(randn(100,2))    %macierz kowariancji z próby
ans =
    1.1104    0.0283
    0.0283    0.8887

>> sum(aa)             %suma elementów w kolumnach
ans =
     5     7     9
>> sum(aa,2)           %suma elementów w wierszach
ans =
     6
    15

>> min(aa)            %minimum w kolumnach
ans =
     1     2     3
>> min(aa,2)          %minimum z dwóch elementów dla całej macierzy aa
ans =
     1     2     2
     2     2     2
>> min(aa,[],2)       %minimum według 2. wymiaru (w wierszach)
ans =
     1
     4

>> max(aa);           %maksymalna wartość w kolumnach
>> mean(aa)           %średnia wartość w kolumnach
ans =
    2.5000    3.5000    4.5000
>> std(aa)            %odchylenie standardowe w kolumnach
ans =
    2.1213    2.1213    2.1213
>> corr(randn(100,3));   %macierz korelacji
```

Programy i funkcje

Praca w systemie obliczeniowym MATLAB najczęściej polega na pisaniu i wykonywaniu wielu poleceń. Można zautomatyzować ten proces poprzez napisanie w dowolnym edytorze tekstu kolejnych linijek poleceń i potraktowanie ich łącznie jak jednego programu. MATLAB wykona taki program linijka po linijce.

Pliki z programami i funkcjami napisanymi w języku MATLAB mają predefiniowane rozszerzenia „*.m”. Aplikacja MATLAB zawiera własny wbudowany edytor tekstu, który ułatwia pisanie i uruchamianie programów dzięki wbudowanym funkcjom (np. rozpoznawanie funkcji, słów kluczowych itp.).

Podobnie, możliwe jest definiowanie funkcji, które zawierać mogą szereg poleceń. Następnie, każda funkcja może być wywołana wielokrotnie, w dowolnym miejscu programu, za każdym razem przy użyciu pojedynczego polecenia. Funkcje mogą przyjmować argumenty i zwracać dowolną liczbę wartości w postaci macierzy, wektorów, skalarów itp. Funkcja rozpoczyna się od polecenia `function` i kończy się poleceniem `end` lub końcem pliku.

Poszczególne funkcje zapisywane są zwykle w osobnych plikach. Poniższa funkcja przyjmuje dwa argumenty i zwraca trzy wartości.

```
function [x, y, z] = moja_funkcja(a, b)     %definiuje funkcję, argumenty:
a, b
    x = randn(1,1)*a;
    y = randn(1,1)*b;
    z = 1;
    disp('wykonuje procedurę:');
end                                         %koniec definicji funkcji
```

Wynik wywołania funkcji z okna poleceń jest następujący.

```
>> [x,y,z] = moja_funkcja(1, 1)
wykonuje procedurę:
x =
    0.8622
y =
    0.3188
z =
     1
```

Pętle i polecenia warunkowe

Wielokrotne wykonywanie sekwencji poleceń możliwe jest dzięki zastosowaniu pętli. Wyrażenie `for nazwa_zmiennej=start:step:stop … end` pozwala wykonywać grupę poleceń aż do momentu, kiedy zmienna `nazwa_zmiennej` osiągnie wartość `stop`. Zmienna `nazwa_zmiennej` na początku przyjmuje wartość `start` i przy kolejnym wykonaniu pętli zmienia się o wartość `step`.

```
    zz = 0;
    for v = 1:1:5;
        zz = [zz v];
    end
    zz
```

Wynik wywołania pętli jest następujący.

```
zz =
     0     1     2     3     4     5
```

Podobnie zdefiniowane jest wyrażenie `while warunek … end`. Wyrażenie to wykonuje pętlę kiedy `warunek` jest prawdziwy. Na wydruku poniżej warto zwrócić uwagę na sposób drukowania na ekranie tekstu i liczb razem w jednej linijce. Konieczna jest zamiana liczby na tekst.

```
    n = 10;
    f = 0;
    while n > 1
        n = n-1;
        f = f+n;
    end
```

```
    disp(['wynik = ' num2str(f)])
```

Wynik wywołania pętli jest następujący.

```
wynik = 45
```

Warto zwrócić uwagę na operator relacyjny ==, który oznacza równość zmiennych po obu stronach operatora. Warunek `i==4` jest prawdziwy, kiedy `i` równe jest 4. Inne operatory relacyjne to <, >, <=, ~=. Prawdziwe wyrażenie zwraca wartość 1, a fałszywe wyrażenie zwraca 0.

Polecenia warunkowe wykonywane są przy pomocy wyrażenia `if warunek … elseif warunek_2 … elseif warunek_n … else … endif`. Jeśli `warunek` jest prawdziwy to wykonywana jest grupa poleceń po tym warunku. Jeśli warunek jest fałszywy to sprawdzany jest `warunek_2` i gdy jest prawdziwy, to wykonywane są polecenia po tym warunku. Jeśli `warunek_2` jest fałszywy, to sprawdzany jest kolejny warunek, itd. Jeśli żaden warunek nie jest prawdziwy to wykonywane są polecenia po komendzie `else`. Wyrażenia `elseif warunek` i `else` nie są obowiązkowe.

```
    q = 1;
    if q==1;
        disp('q = 1');          %wydrukuje ten tekst
        q = q + 1;
    elseif q==2;                %nie sprawdzi tego warunku
        disp('q = 2');          %nie wydrukuje tego tekstu
    else;
        disp('q <> 1 i 2');     %nie wydrukuje tego tekstu
    end;
    disp(q);                    %wykrukuje 2
```

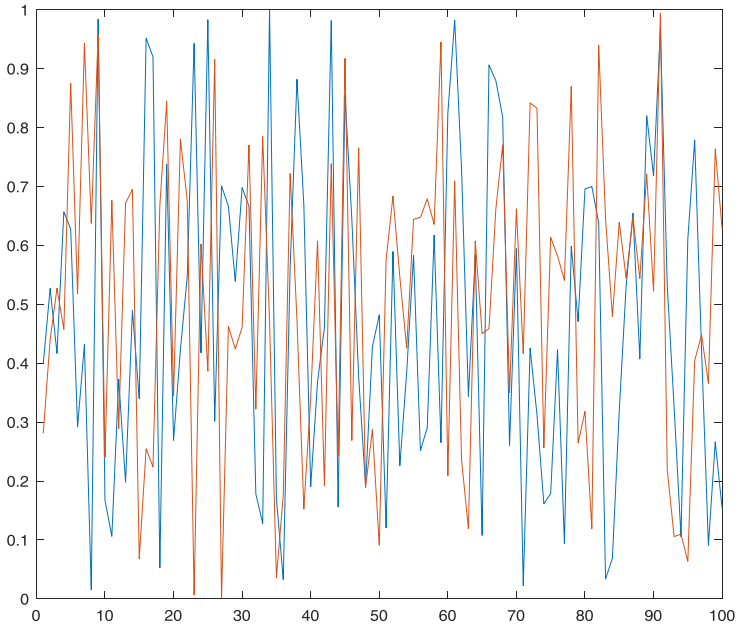
W trakcie wykonywania powyższego programu komputer wydrukuje następujący wynik.

```
q = 1
     2
```

Grafika

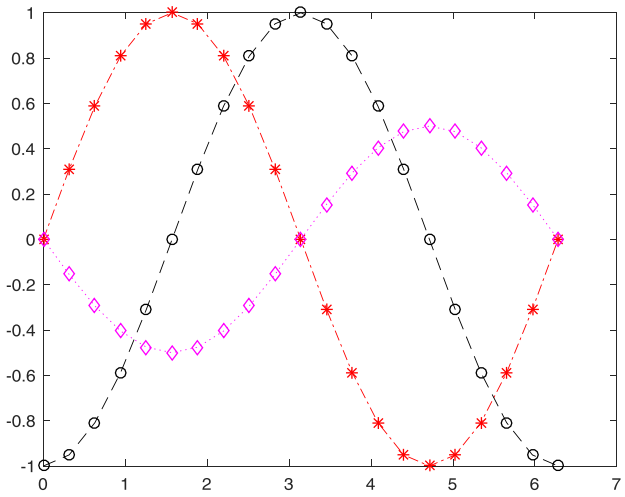
Tworzenie wykresów w Gaussie jest bardzo proste, a dostępne polecenia i opcje umożliwiają zmienianie wielu parametrów wykresów. Do rysowania wykresów liniowych przydatne jest polecenie `plot`.

```
    plot(1:100,rand(100,2));    %wykres dwóch zmiennych/szeregów
```



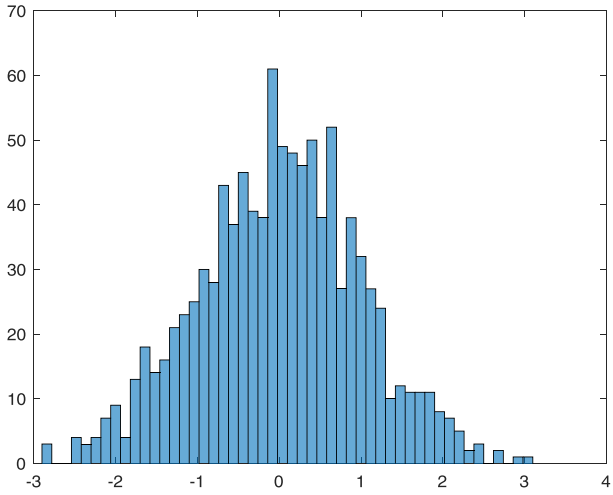
Skopiowanie wykresu do innego dokumentu lub nagranie do pliku możliwe jest z menu okna wykresu (odpowiednio, pozycja „Edit” → „Copy Figure” lub „File” → „Save As…”)
Można manipulować kształtem i kolorem linii poprzez dodanie dodatkowych opcji do polecenia `plot` (rodzaje linii: - , -- , : , -. oraz rodzaje punktów: + , o , * , . , x , s , d , ^ , v , < , > , p , h oraz kolory linii: r , g , b , c , m , y , k , w).

```
figure                          %rozpoczyna nowe okno z wykresem
    t = 0:pi/10:2*pi;           % wartości na osi x
    plot(t,sin(t),'-.r*')       % linie+kropki i czerwone gwiazdki
hold on                         % następne linie na tym samym wykresie
    plot(t,sin(t-pi/2),'--ko')  % przerywana linia, czarna, okręgi
    plot(t,sin(t-pi)*0.5,':md') % punkty, różowe, diamenty
hold off                        % koniec dorysowywania do wykresu
```



Możliwe jest między innymi rysowanie histogramu dla wektora obserwacji przy pomocy polecenia `histogram`.

```
>> histogram(rand(1000,1));       % histogram z rozkładu u(0,1)
>> histogram(randn(1000,1),50);   % histogram z rozkładu N z 50 przedziałami
```



Dostępne są funkcje do rysowania wykresów przedziałowych, z dwoma osiami y, 3D, typu „pudełkowego”, okrągłe itp. Szczegóły w oknie pomocy, np. po wpisaniu „graphics”.