**P O L I T E C H N I K A  G D A Ń S K A**

WYDZIAŁ INŻYNIERII LĄDOWEJ

# PODSTAWY PROGRAMOWANIA

# W JĘZYKU MATLAB

ROBERT JANKOWSKI, IZABELA LUBOWIECKA, WOJCIECH WITKOWSKI


GDAŃSK 2002

# WSTĘP

- Niniejszy zeszyt przeznaczony jest dla studentów Wydziału Inżynierii Lądowej Politechniki Gdańskiej jako pomoc dydaktyczna do laboratorium z programowania w języku MATLAB, prowadzonego w ramach przedmiotu podstawy informatyki.
- W pierwszej części zeszytu omówiono podstawowe funkcje: operacje na macierzach, działania tablicowe, operatory logiczne oraz elementy algebry liniowej.
- W następnej części pokazano pracę w tzw. skryptach wraz z instrukcjami sterującymi oraz zastosowaniem funkcji.
- Kolejną część zeszytu poświęcono grafice dwu- i trójwymiarowej.
- W ostatniej części pokazano przykładowe programy z dziedziny mechaniki budowli, wytrzymałości materiałów i dynamiki.
- Działania poszczególnych instrukcji zobrazowano w postaci licznych przykładów przeznaczonych do samodzielnego wykonania.
- W zeszycie czcionką `courier` wyróżniono komendy języka MATLAB.
- Niektóre przykłady programów pochodzą z książki A. Zalewskiego i R. Cegieły pt. „MATLAB – obliczenia numeryczne i ich zastosowania”, Wydawnictwo Nakom, Poznań 1997.

# Środowisko i programowanie w języku MATLAB

- MATLAB - pakiet obliczeniowy firmy *MathWorks* jest przeznaczony do wykonywania różnorodnych obliczeń numerycznych.
- Serce pakietu stanowi interpreter języka umożliwiający implementację algorytmów numerycznych oraz biblioteki podstawowych działań na macierzach (odwracanie, dodawanie/odejmowanie, wartości własne itp.).
- Podstawowym typem danych jest macierz, stąd nazwa MATrix LABoratory.
- Pakiet posiada obszerne biblioteki dodatkowych procedur umożliwiające rozwiązywanie typowych problemów obliczeniowych.
- Prosta budowa okienkowa ułatwia korzystanie z programu.
- Łatwa i estetyczna jest wizualizacja wyników w postaci dwu- i trójwymiarowych wykresów.
- Dodatkową zaletą pakietu MATLAB jest możliwość przeprowadzenia obliczeń symbolicznych (na wzorach).

# Wprowadzenie do pracy w środowisku języka MATLAB

- Praca w środowisku języka MATLAB polega na wydawaniu poleceń, które po zatwierdzeniu wykonywane są przez interpreter.
- Większą liczbę instrukcji można zapisać w zbiorze tekstowym zwanym skryptem (pliki z rozszerzeniem *.m*).

# Przykłady poleceń

- Podstawienie:

```
» a=3;
```

powoduje utworzenie zmiennej *a* o wartości 3.
UWAGA:
Średnik po poleceniu powoduje, że wartość będąca wynikiem nie będzie wyświetlana na ekranie.

```
» b=sin(a)
b =
    0.1411
```

oblicza wartość funkcji sinus dla zmiennej *a*, wynik zapisuje do zmiennej *b* i wyświetla na ekranie.

- Jeżeli nie podano nazwy zmiennej to wynik działania jest umieszczany w standardowej zmiennej *ans*, np.:

```
» cos(pi/3)
ans =
    0.5000
```

- Utworzona (zdefiniowana) zmienna jest pamiętana od momentu utworzenia, aż do chwili jej usunięcia. Możliwa jest przy tym nie tylko zmiana wartości, ale również rozmiaru zmiennej.

Nazwy zmiennych i informacje o nich można uzyskać wywołując funkcje `who` i `whos`.

- Usunięcie zmiennej z pamięci:
  `clear a` - usuwa zmienną *a*;
  `clear` - usuwa wszystkie zmienne znajdujące się w pamięci.

- Zapisanie zmiennych na dysku:
  `save nazwa_pliku` (domyślnie przyjmowane jest rozszerzenie *.mat*).

- Wczytanie danych z pliku dyskowego:
  `load nazwa_pliku`

- Korzystanie z podręcznej pomocy podającej opis funkcji:
  `help nazwa_funkcji`

- Zawartość aktualnego katalogu można wyświetlić używając funkcji `dir` lub `ls`.

- Do zmiany katalogu służy polecenie:
  `cd nazwa_katalogu`

## Liczby rzeczywiste i ich formaty

- Podstawowym typem dla elementów macierzy wykorzystywanym przez MATLAB są liczby rzeczywiste.
- Maksymalną i minimalną wartość liczby rzeczywistej dodatniej można poznać za pomocą funkcji `realmax i realmin`.
- Do określenia sposobu, w jaki liczby rzeczywiste są przedstawione na ekranie służy polecenie `format postać_liczby`, gdzie *postać_liczby* określa postać, w jakiej liczby rzeczywiste będą wyświetlane na ekranie (np. `short, short e, long`).

***Przykład:***
Przedstaw liczbę 2,5 w różnej postaci używając funkcji `format`.

```
» format short
» 2.5
ans =
    2.5000
» format short e
» 2.5
ans =
    2.5000e+000
» format long
» 2.5
ans =
    2.50000000000000
```

# Macierze

- Definicja macierzy przez wyliczenie elementów:

  ***Przykład:***

```
» A=[2 2 2 1; 1 2 3 1];
```

  lub:

```
» A=[2 2 2 1
1 2 3 1]
A =
     2     2     2     1
     1     2     3     1
```

  Poszczególne elementy macierzy oddziela się spacjami, a wiersze średnikami lub umieszcza się je w oddzielnych liniach.

- Definicja macierzy przez wygenerowanie elementów:
  `A=[min:krok:max]`
  Polecenie generuje wektor poczynając od elementu o wartości *min*, kończąc na elemencie o wartości *max* z krokiem *krok*. Jeżeli parametr *krok* zostanie pominięty, przyjmuje się, iż *krok*=1.

  ***Przykład:***
  Wygeneruj macierz dwuwierszową o wyrazach od 1 do 10 w pierwszym wierszu i o wyrazach od 2 do 20 (co 2) w wierszu drugim.

```
» A=[1:10; 2:2:20]
A =
     1     2     3     4     5     6     7     8     9    10
     2     4     6     8    10    12    14    16    18    20
```

- Definicja macierzy wykorzystując elementy innych macierzy:

  ***Przykład:***
  Utwórz macierz **D** budując ją ze zdefiniowanych macierzy **A**, **B** i **C**.

```
» A=[1 4 1; 2 0 1];
» B=[3 1; 4 1];
» C=[1 2 2 0 1; 2 4 7 1 0];
» D=[A B; C]
D =
     1     4     1     3     1
     2     0     1     4     1
     1     2     2     0     1
     2     4     7     1     0
```

  UWAGA:
  Przy takim budowaniu macierzy należy pamiętać o zgodności wymiarów.

## Wymiar i wyświetlanie macierzy

- `[n,m]=size(A)` - zwraca liczbę kolumn *n* i wierszy *m* macierzy **A**;
- `n=length(B)` - zwraca wymiar wektora **B** (lub większy z wymiarów macierzy **B**);
- `A` lub `disp(A)` - pokazuje macierz **A** na ekranie;

## Funkcje wspomagające konstruowanie macierzy

- Definicja macierzy jednostkowej:

  ***Przykład:***
  Utwórz kwadratową macierz jednostkową **A** o wymiarze 3x3.

```
» A=eye(3)
A =
     1     0     0
     0     1     0
     0     0     1
```

- Definicja macierzy wypełnionej jedynkami:

  ***Przykład:***
  Utwórz macierz **A** o wymiarze 2x3 wypełnionej jedynkami.

```
» A=ones(2,3)
A =
     1     1     1
     1     1     1
```

- Definicja macierzy wypełnionej zerami:

  ***Przykład:***
  Utwórz macierz **A** o wymiarze 3x2 wypełnionej zerami.

```
» A=zeros(3,2)
A =
     0     0
     0     0
     0     0
```

## Dostęp do elementów macierzy

- Odwołanie do elementów:

  ***Przykład:***

```
» A=[1 2 3; 0 9 8; 1 1 0]
A =
     1     2     3
     0     9     8
     1     1     0
» A(2,3)        - odwołanie do elementu w wierszu 2 i kolumnie 3;
ans =
     8
» A(3,2)        - odwołanie do elementu w wierszu 3 i kolumnie 2
ans =
     1
```

- Wybór największego elementu

- `max(A)` - zwraca największy element wektora **A**. W przypadku gdy **A** jest macierzą, zwraca wektor wierszowy, którego elementami są maksymalne elementy z każdej kolumny **A**

***Przykład:***

```
» max(A)
ans =
   1     9     8
```

- Wybór najmniejszego elementu

- `min(A)` - zwraca najmniejszy element wektora **A**. W przypadku gdy **A** jest macierzą, zwraca wektor wierszowy, którego elementami są maksymalne elementy z każdej kolumny **A**

***Przykład:***

```
» min(A)
ans =
   0     1     0
```

- Obliczanie wartości średniej elementów

- `mean(A)` - zwraca średnią arytmetyczną elementów wektora **A**. W przypadku gdy **A** jest macierzą, zwraca wektor wierszowy, którego elementami są średnie arytmetyczne elementów z każdej kolumny **A**

***Przykład:***

```
» mean(A)
ans =
   0.6667    4.0000    3.6667
```

- Odwołanie do podmacierzy:

***Przykład:***

```
» A=[1 2 3 4 5 6; 0 9 8 7 6 5; 1 1 0 0 2 2]
A =
     1     2     3     4     5     6
     0     9     8     7     6     5
     1     1     0     0     2     2
» B=A(:,[1:3 5])          - utworzenie macierzy B poprzez pobranie z macierzy A
B =                         kolumn: 1-3 oraz 5
     1     2     3     5
     0     9     8     6
     1     1     0     2
» B=A([1 3],1:2:5)        - utworzenie macierzy B z elementów macierzy A leżących
B =                         na przecięciu wierszy 1 i 3 z kolumnami 1, 3 i 5
     1     3     5
     1     0     2
```

- Usuwanie wektora z macierzy:

***Przykład:***

```
» A=[1 2 3 4; 4 5 6 7]
A =
     1     2     3     4
     4     5     6     7
» A(2,:)=[ ]              - usuwa drugi wiersz z macierzy A
A =
     1     2     3     4
» A(:,1:2)=[ ]            - usuwa dwie pierwsze kolumny z macierzy A
A =
     3     4
```

## Działania na macierzach

- Suma i różnica macierzy

***Przykład:***
Zdefiniuj dwie macierze **A** i **B**, a następnie oblicz ich sumę, różnicę oraz dodaj do elementów macierzy **A** liczbę 2.

Definicja macierzy:

```
» A=[1 -1 2; -2 3 1]
A =
     1    -1     2
    -2     3     1
» B=[1 1 1; 0 -2 2]
```

```
B =
     1     1     1
     0    -2     2
```

Suma:

```
» A+B
ans =
     2     0     3
    -2     1     3
```

Różnica:

```
» A-B
ans =
     0    -2     1
    -2     5    -1
```

Dodanie do elementów macierzy **A** liczby 2:

```
» A+2
ans =
     3     1     4
     0     5     3
```

- Mnożenie macierzy

***Przykład:***
Zdefiniuj dwie macierze **A** i **B**, a następnie oblicz ich iloczyn oraz pomnóż elementy macierzy **A** przez 2.

Definicja macierzy:

```
» A=[1 1 0; 2 1 1]
A =
     1     1     0
     2     1     1
» B=[2; 2; 2]
B=
     2
     2
     2
```

Iloczyn macierzowy:

```
» A*B
ans =
     4
     8
```

Iloczyn macierzy przez liczbę:

```
» A*2
```

```
ans =
     2     2     0
     4     2     2
```

- Odwracanie i transpozycja

***Przykład:***
Zdefiniuj macierz **A**, a następnie wyznacz macierz odwrotną do niej i dokonaj transpozycji.

```
» A=[1 2 3; 0 9 8; 3 4 7]
A =
     1     2     3
     0     9     8
     3     4     7
»inv(A)                                  - zwraca macierz odwrotną do A
ans =
   -15.5000    1.0000    5.5000
   -12.0000    1.0000    4.0000
    13.5000   -1.0000   -4.5000
» A'                                     - transponuje macierz A
ans =
     1     0     3
     2     9     4
     3     8     7
```

- ***Przykład***
Zdefiniuj wektor kolumnowy **A**, a następnie oblicz sumę kwadratów elementów tego wektora.

```
» A=[1 2 3]'
A =
     1
     2
     3
» A'*A
ans =
     14
```

## Działania tablicowe

- Działanie tablicowe jest działaniem, które przekształca poszczególne elementy macierzy oddzielnie.

***Przykład:***
Zdefiniuj dwie macierze **A** i **B**, a następnie wykonaj działania mnożenia, dzielenia i potęgowania tablicowego.

Definicja macierzy:

```
» A=[5 -6 2; -2 4 1]
A =
     5    -6     2
    -2     4     1
» B=[5 2 2; -1 -2 1]
B =
     5     2     2
    -1    -2     1
```

Mnożenie tablicowe:

```
» A.*B
ans =
    25   -12     4
     2    -8     1
```

Dzielenie tablicowe:

```
» A./B
ans =
     1    -3     1
     2    -2     1
```

Potęgowanie tablicowe (podniesienie elementów macierzy **A** do drugiej potęgi):

```
» A.^2
ans =
    25    36     4
     4    16     1
```

## Algebra liniowa

- `det(A)` - obliczanie wyznacznika macierzy **A**
- `eig(A)` - obliczanie wartości własnych macierzy **A**
- `poly(A)` - obliczanie współczynników wielomianu charakterystycznego macierzy **A**
- `rank(A)` - obliczanie rzędu macierzy **A**
- `diag(A)` - wyznaczanie elementów leżących na głównej przekątnej macierzy **A**

- ***Przykład:***
  Zdefiniuj macierz **A** o wymiarze 4x4, a następnie wyznacz jej wyznacznik, wartości własne, współczynniki wielomianu charakterystycznego oraz zbadaj rząd macierzy.

```
» A=[1 3 0 -2; 2 0 3 -1; 0 5 0 0; 1 0 2 0];
» det(A)
ans =
     0
```

```
» eig(A)
ans =
   -4.5414
    4.0000
    1.5414
    0.0000
» poly(A)
ans =
    1.0000   -1.0000   -19.0000   28.0000   0.0000
» rank(A)
ans =
    3
```

- ***Przykład:***
  Rozwiąż układ równań liniowych:

$$\begin{cases} x+2y-z=3 \\ 3x-4y+2z=-5 \\ 5x-2y+3z=2 \end{cases}$$

  UWAGA:
  Układ ten można zapisać w postaci macierzowej: $\mathbf{A}\cdot\mathbf{X}=\mathbf{B}$, gdzie:

$$\mathbf{A}=\begin{bmatrix} 1 & 2 & -1 \\ 3 & -4 & 2 \\ 5 & -2 & 3 \end{bmatrix},\ \mathbf{X}=\begin{bmatrix} x \\ y \\ z \end{bmatrix},\ \mathbf{B}=\begin{bmatrix} 3 \\ -5 \\ 2 \end{bmatrix},$$

  dla której rozwiązanie ma postać: $\mathbf{X}=\mathbf{A}^{-1}\cdot\mathbf{B}$

```
» A=[1 2 -1; 3 -4 2; 5 -2 3];
» B=[3 -5 2]';
» X=inv(A)*B
X =
    0.2000
    2.3500
    1.9000
```

## Operacje na łańcuchach

- Uzupełniającym typem danych w języku MATLAB jest typ łańcuchowy (tekstowy). Do definiowania zmiennej tego typu stosuje się apostrofy, np.:

```
» s='MATLAB'
s =
MATLAB
```

- Na zmiennych typu łańcuchowego można dokonywać niektórych działań macierzowych, na przykład transpozycji:

```
» s'
ans =
M
A
T
L
A
B
```

- Zmienna typu łańcuchowego może zawierać nazwę instrukcji, którą można wykonać używając funkcji `eval`.

***Przykład:***

```
» t=[0:0.2:1];
» s='sin(t)';
» eval(s)
ans =
    0   0.1987    0.3894    0.5646    0.7174    0.8415
```

- Można wysyłać na ekran wywołanie zachęty oczekujące na wprowadzenie przez użytkownika danej wartości lub łańcucha znaków, np.:

```
» a=input('Podaj wartość a: ')
Podaj wartość a:
```

lub:

```
» wzor=input('Podaj wzór funkcji f(x): ','s')
Podaj wzór funkcji f(x):
```

UWAGA:
Użycie parametru *'s'* w funkcji `input` powoduje, iż wprowadzona dana jest traktowana jako łańcuch znaków.

## Skrypty

- ***Przykład:***
Napisz skrypt (otwierając z menu *File* z opcji *New* plik *M-file*), który kreśli wykres wybranej przez użytkownika funkcji jednej zmiennej w przedziale $\langle 0, 4\pi \rangle$.

```
% skrypt rysuje wykres wybranej funkcji
```

```
x=[0:0.1:4*pi];
wzor=input('Podaj wzór funkcji jednej zmiennej f(x): ','s')
y=eval(wzor);
plot(x,y);   % kreślenie wykresu funkcji y=f(x)
```

Zapisz go pod nazwą *wykres.m*, a następnie uruchom wpisując w oknie komend jego nazwę:
```
» wykres
```

WSKAZÓWKA:
Podaj na przykład funkcję: `sin(x)+2*cos(2*x)`

## Operatory logiczne

- Operatory logiczne w języku MATLAB:

| | |
|---|---|
| == | równe |
| ~= | różne |
| < | mniejsze |
| > | większe |
| <= | mniejsze równe |
| >= | większe równe |
| & | i |
| \| | lub |

## Instrukcje sterujące

- Pętla FOR („dla”):

```
for zmienna_iterowana = macierz_wartości
   ciąg_instrukcji
end
```

Działanie pętli polega na wykonaniu *ciągu_instrukcji* dla kolejnych wartości *zmiennej_iterowanej*. Wartościami tymi są kolejne wektory kolumnowe pobrane z *macierzy_wartości* (jeżeli jest to wektor, to kolejno zostaną wykonane instrukcje dla danych elementów tego wektora).

***Przykład:***
Napisz skrypt, który generuje wektor **A** o wymiarze 1x5, którego elementy spełniają zależność:

$$A_i = \sqrt{1+i}$$

```
% Próba realizacji pętli FOR
for i=1:5
   A(i)=sqrt(1+i);          % pierwiastek kwadratowy
```

```
end
A
```

Zapisz go w pliku *petlafor.m* i uruchom.

Rozbuduj powyższy skrypt, aby generował macierz **A** o wymiarze 10x5, którego elementy spełniają zależność:

$$A_{ij} = \sqrt{1 + \frac{i}{j}}$$

```
% Próba realizacji pętli FOR
for i=1:10
   for j=1:5
      A(i,j)=sqrt(1+i/j);      % pierwiastek kwadratowy
   end
end
A
```

- Pętla WHILE („dopóki”):

```
while wyrażenie_warunkowe
   ciąg_instrukcji
end
```

Działanie pętli polega na wykonaniu *ciągu_instrukcji* dopóki *wyrażenie_warunkowe* jest spełnione.

***Przykład:***

```
% Próba realizacji pętli WHILE
i=0;
while i<100
   i=i+1
end
```

Zapisz skrypt w pliku *petlawhile.m* i uruchom go.

- Instrukcja warunkowa IF („jeżeli”):

```
if wyrażenie_warunkowe1
   ciąg_instrukcji1
elseif wyrażenie_warunkowe2
   ciąg_instrukcji2
else
   ciąg_instrukcji3
end
```

Działanie instrukcji jest następujące:
Jeżeli *wyrażenie_warunkowe1* jest spełnione, to wykonywany jest *ciąg_instrukcji1,* w

przeciwnym razie sprawdzane jest *wyrażenie_warunkowe2,* jeżeli jest ono spełnione wykonywany jest *ciąg_instrukcji2,* jeżeli nie, wykonywany jest *ciąg_instrukcji3.* Instrukcję warunkową IF można rozbudować dla większej liczby *wyrażeń_warunkowych* i odpowiadających im *ciągów_instrukcji.*

***Przykład:***
Napisz skrypt używając instrukcji warunkowej IF do zrealizowania wyboru jednej z dostępnych opcji (polecenie `menu`):

```
% Próba realizacji instrukcji IF
o=menu('Przykładowe menu', 'Opcja 1', 'Opcja 2', 'Opcja 3');
if (o==1)
   disp('Opcja 1')
elseif (o==2)
   disp('Opcja 2')
elseif (o==3)
   disp('Opcja 3')
end
```

Zapisz skrypt w pliku *instrukcjaif.m* i uruchom go.

## Funkcje

- W języku MATLAB istnieje możliwość definiowania własnych funkcji, jako elementów strukturalnych programu. Definicja funkcji ma następującą postać:

```
function[wartość_funkcji]=nazwa_funkcji(argument1,..,argumentN)
ciąg instrukcji
```

***Przykład:***
Napisz funkcję (otwierając z menu *File* z opcji *New* plik *M-file*) wyznaczającą wartość silni *n!*, gdzie *n* jest liczbą naturalną.

```
% Funkcja wyznacza watość n!
function[wynik]=silnia(n)
wynik=1;
for i=1:n
   wynik=wynik*i;
end
```

Zapisz ją pod nazwą *silnia.m*, a następnie uruchom wpisując w linii komend jej nazwę wraz z wartością argumentu *n* umieszczoną w nawiasie, np.:

```
» silnia(5)
ans =
    120
```

## Budowa strukturalna programu

- Skrypty, które stanowią większą całość nazywamy programami.
- W skrypcie możemy wywołać istniejące już (wcześniej zdefiniowane) inne skrypty lub funkcje.
- Polecenie `help nazwa_skryptu` wyświetla na ekranie tekst umieszczony w pierwszych liniach komentarza.

***Przykład:***
Napisz program, który wypisuje na ekranie informację o jego działaniu oraz imię i nazwisko autora, a następnie wyznacza wartość *n!* dla podanej przez użytkownika wartości *n*.
(Uwaga: użyta w poniższym przykładzie funkcja `round(n)` zaokrągla liczbę rzeczywistą *n* do liczby całkowitej)

```
% Program oblicza wartość silni n! dla wprowadzonej przez
% użytkownika wartości n

disp('Program oblicza wartość silni n! dla wprowadzonej przez')
disp('użytkownika wartości n')
disp(' ')
disp('Autor:')
disp('Imię i Nazwisko')
disp(' ')
n=input('Podaj wartość n: ');

%sprawdzenie czy n jest liczbą naturalną
while n<0 | n~=round(n)
   disp('Proszę podać liczbę naturalną')
   n=input('Podaj wartość n: ');
end

disp('Wartość n! wynosi:')
silnia(n)
```

Zapisz go pod nazwą *program.m* i uruchom.

## Grafika dwuwymiarowa

- Najczęściej spotykanym sposobem graficznej prezentacji danych w języku MATLAB jest wykres funkcji jednej zmiennej. Służy do tego funkcja `plot(x,y)`, gdzie y=f(x);
- Okno graficzne można wyczyścić wywołując funkcję `clf`;
- Zamknięcie okna graficznego odbywa się poprzez wywołanie funkcji `close`;
- Dodatkowe okna można otworzyć przy pomocy funkcji `figure`;
- Otworzyć jak i zamknąć można dowolne okno podając jego numer jako argument;
- W celu uzyskania kilku wykresów w jednym oknie należy wykorzystać funkcję

`subplot(m,n,p)`, gdzie:
*m* - liczba wykresów w pionie;
*n* - liczba wykresów w poziomie;
*p* - kolejny numer wykresu.

- Skala wykresu dobierana jest automatycznie. Chcąc ją zmienić, trzeba wywołać funkcję `axis([xmin xmax ymin ymax])` i jako argument podać wektor określający nowe parametry osi.
- Wykres można opisać podając nazwy zmiennych, tytuł, itp.

`title('tekst')` - tytuł rysunku;
`xlabel('tekst')` - opis osi x;
`ylabel('tekst')` - opis osi y;
`text(x,y,'tekst')` - umieszcza *'tekst'* w dowolnym punkcie o współrzędnych (x,y);
`grid` - włącza lub wyłącza siatkę;

***Przykład***:
Napisz skrypt kreślący przykładowy wykres wraz z opisem.

```
% Skrypt kreśli przykładowy wykres wraz z opisem
x=[0:pi/20:2*pi];
y=sin(x);
plot(x,y)
title('Wykres funkcji sin(x)')
xlabel('x')
ylabel('f(x)')
text(2.5,0.7,'f(x)=sin(x)')
grid
```

Zapisz go pod nazwą *wykresopis.m* i uruchom.

## Rysowanie

- Istnieją funkcje pozwalające na tworzenie dowolnych rysunków z linii i wielokątów.

`line(x,y)` - rysuje linię łamaną łącząc wierzchołki punktów wyznaczonych przez elementy wektorów **x** i **y**;
`fill(x,y,'c')` - rysuje wielokąt o wierzchołkach w punktach wyznaczonych przez elementy wektorów **x** i **y** wypełniony kolorem określonym przez argument *c* według poniższego opisu kolorów:

*y* - żółty
*m* - fioletowy
*c* - turkusowy
*r* - czerwony
*g* - zielony
*b* - niebieski
*w* - biały

*k* - czarny

***Przykład***:
Narysuj trójkąt o wierzchołkach w punktach (0,1), (3,4), (4,2) używając funkcji `line` oraz `fill` z wypełnieniem w kolorze niebieskim.

```
» line([0 3 4 0],[1 4 2 1])
» fill([0 3 4],[1 4 2],'b')
```

## Grafika trójwymiarowa

- Większość funkcji języka MATLAB generujących rysunki trójwymiarowe służy do kreślenia powierzchni. W praktyce definiując powierzchnię trzeba się ograniczyć do skończonego zbioru punktów należących do obszaru.

| | |
|---|---|
| `[x,y]=meshgrid(X,Y)` | - tworzy macierze **x** i **y** opisujące położenie węzłów prostokątnej siatki pobierając wartości z wektorów **X** i **Y**. |
| `mesh(x,y,z)` | - rysuje siatkę powierzchni opisanej przez macierze **x**, **y** i **z**. |
| `surf(x,y,z)` | - rysuje kolorową powierzchnię opisaną przez macierze **x**, **y** i **z**. |
| `surfl(x,y,z)` | - rysuje kolorową powierzchnię opisaną przez macierze **x**, **y** i **z** uwzględniając na niej odbicie światła. |
| `plot3(x,y,z)` | - rysuje krzywą w przestrzeni opisaną przez wektory **x**, **y** i **z**. |

***Przykład:***
Napisz skrypt kreślący siatkę wartości funkcji $f(x,y)=\sin(x)\cdot\sin(y)\cdot\exp(-x^2-y^2)$ w przedziale $\langle -\pi,\pi \rangle$.

```
% Skrypt rysuje siatkę wartości funkcji
clf
[x,y]=meshgrid(-pi:0.2:pi,-pi:0.2:pi)
z=sin(x).*sin(y).*exp(-x.^2-y.^2)
mesh(x,y,z)
```

Zapisz go pod nazwą *wykres3d.m* i uruchom.

Rozbuduj powyższy skrypt o rysowanie kolorowej powierzchni poprzez dodanie na końcu polecenia:
`surf(x,y,z)`
lub:
`surfl(x,y,z)`

***Przykład:***
Napisz skrypt kreślący krzywą w przestrzeni trójwymiarowej:

```
% Skrypt kreśli krzywą w przestrzeni trójwymiarowej
x=[0:0.1:10];
y=2*cos(x);
z=sin(2*y);
plot3(x,y,z)
grid
title('Wykres krzywej w przestrzeni trójwymiarowej')
xlabel('x')
ylabel('y')
zlabel('z')
```

Zapisz go pod nazwą *krzywa3d.m* i uruchom.

- Wykreślone powierzchnie można poddać cieniowaniu używając funkcji:

```
shading flat
shading interp
shading faceted
```

***Przykład:***
Napisz skrypt:

```
% Skrypt rysuje powierzchnie poddane cieniowaniu
clf
[x,y]=meshgrid(-3.5:0.7:3.5);
z=sin(x).*sin(y)+4*exp(-(x-0.5).^2-(y-0.5).^2);

%Wykres w trybie flat
subplot(1,3,1)
surf(x,y,z)
shading flat
title('flat')

%Wykres w trybie interp
subplot(1,3,2)
surf(x,y,z)
shading interp
title('interp')

%Wykres w trybie faceted
subplot(1,3,3)
surf(x,y,z)
shading faceted
title('faceted')
```

Zapisz go pod nazwą *powierzchnie.m* i uruchom.

- Inne elementy rysunków, takie jak: opisy, etykiety, linie pomocnicze wykonuje się podobnie,

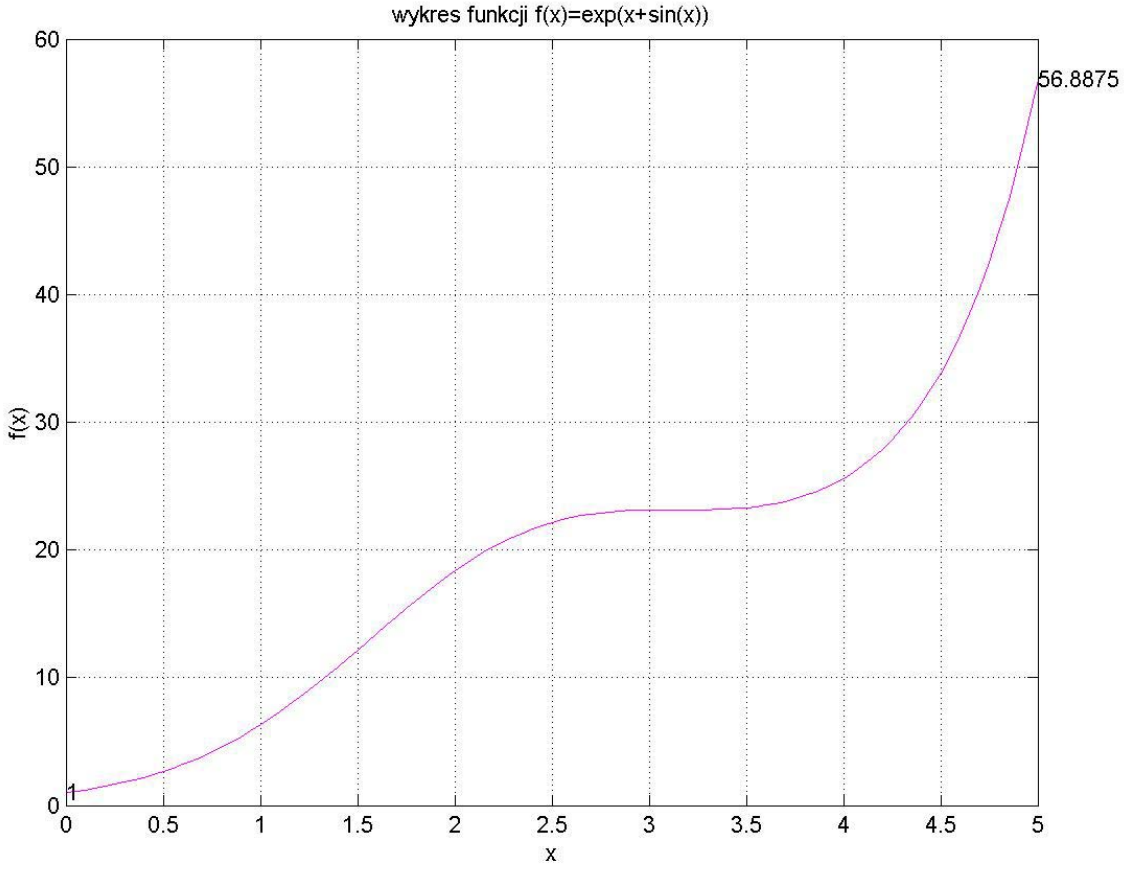
jak w grafice dwuwymiarowej. Dodatkowo jednak należy zdefiniować elementy dotyczące trzeciego wymiaru, np.:

```
text(x,y,z,'tekst');
```

## Przykłady programów w języku MATLAB

- ***Przykład 1:***

```
%Program rysuje wykres wybranej funkcji:
%f(x)=exp(x), f(x)=exp(x+sin(x)), f(x)=exp(x+log(x)),
%f(x)=exp(x+(3*x+1))
%w zadanym przez użytkownika przedziale
clear
clc
disp('           Program rysuje wykres wybranej funkcji:')
disp('f(x)=exp(x), f(x)=exp(x+sin(x)), f(x)=exp(x+log(x)),
f(x)=exp(x+(3*x+1))')
disp('            w zadanym przez użytkownika przedziale')
disp(' ')
disp('                 < naciśnij dowolny klawisz >')
pause

%wybieranie funkcji i przedziału
o=menu('wybierz
funkcje','f(x)=exp(x)','f(x)=exp(x+sin(x))','f(x)=exp(x+log(x))
','f(x)=exp(x+(3*x+1))');
disp(' ')
min=input('Podaj początek przedziału : ');
max=input('Podaj koniec przedziału : ');
while max<=min
   disp('     Podaj wartość większą niż początek przedziału !')
   max=input('Podaj koniec przedziału : ');
end
krok=(max-min)/100;
x=[min:krok:max];

%rysowanie wykresu
clf
if(o==1)
   y=exp(x);
   plot(x,y,'b-')
   title('Wykres funkcji f(x)=exp(x)')
elseif(o==2)
   y=exp(x+sin(x));
   plot(x,y,'m-')
   title('wykres funkcji f(x)=exp(x+sin(x))')
```




```
elseif(o==3)
   y=exp(x+log(x));
   plot(x,y,'r-')
   title('wykres funkcji f(x)=exp(x+log(x))')
elseif(o==4)
   y=exp(x+(3*x+1));
   plot(x,y,'g-')
   title('wykres funkcji f(x)=exp(x+(3*x+1))')
end
xlabel('x')
ylabel('f(x)')
grid
text(x(1),y(1),num2str(y(1)))
text(x(101),y(101),num2str(y(101)))

save wyniki.mat
```

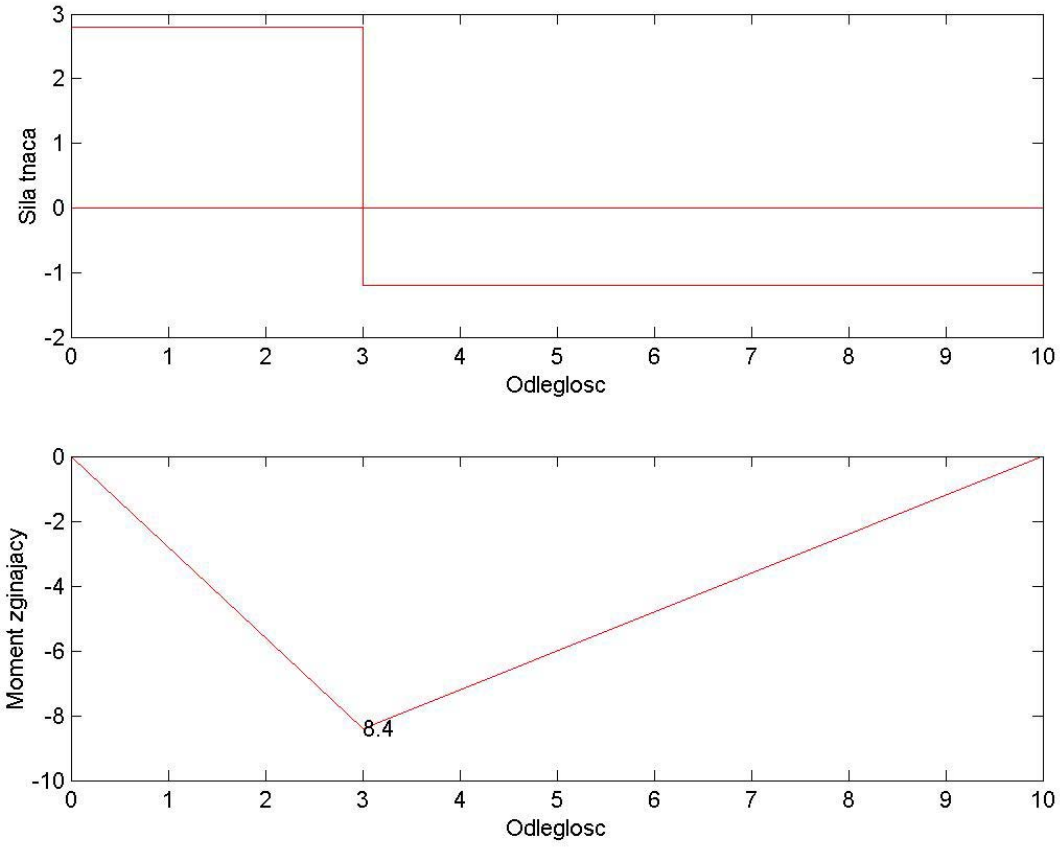
- ***Wyniki działania programu:***
  Poniżej przedstawiono wykres funkcji $f(x)=\exp(x+\sin(x))$ w przedziale $\langle 0,5\rangle$, będący przykładowym wynikiem działania programu:

- ***Przykład 2:***

```
%Program rysuje wykresy sil tnących i momentów zginających
%belki wolnopodpartej obciążonej siłą skupioną
%Dane do programu:
%       l - dlugość belki
%       P - wartość siły skupionej
%       x - odległość punktu przyłożenia siły od lewej podpory
clear
clc
disp('Program rysuje wykresy sil tnących i momentów zginających
belki wolnopodpartej')
disp('      obciążonej siłą skupioną przyłożoną w wybranym
punkcie belki')
disp(' ')

%wprowadzanie danych
l=input('Podaj długość belki wolnopodpartej l= ');
while l<=0
   disp('     !!! Długość musi być wartością dodatnią  !!!')
   l=input('Podaj długość belki wolnopodpartej l= ');
end
P=input('Podaj wartość siły skupionej P= ');
x=input('Podaj odległość punktu przyłożenia siły od lewej
podpory belki x= ');
while x<0 | x>l
   disp('     !!! Punkt przyłożenia siły musi się znajdować na
długości belki !!!')
   x=input('Podaj odległość punktu przyłożenia siły od lewej
podpory belki x= ');
end

%obliczanie reakcji
disp(' ');
disp('Reakcja na lewej podporze:');
Ra=P*(l-x)/l
disp('Reakcja na prawej podporze:');
Rb=P*x/l

%wartości sił wewnętrznych w wybranych punktach belki
disp('Maksymalny moment zginający:')
Mmax=P*x*(l-x)/l
i=[0 0 x x l l]';
T=[0 Ra Ra -Rb -Rb 0]';
M=[0 0 -Mmax -Mmax 0 0]';

%rysowanie wykresów
clf
```

```
subplot(2,1,1)
plot(i,T,'Color','red')
line([0 l],[0 0],'Color','red')
xlabel('Odleglość')
ylabel('Siła tnąca')
subplot(2,1,2)
plot(i,M,'Color','red')
line([0 l],[0 0],'Color','red')
xlabel('Odległość')
ylabel('Moment zginający')
text(x,-Mmax,num2str(Mmax))

save wyniki.mat
```

- ***Wyniki działania programu:***
  Poniżej przedstawiono wykresy sił tnących i momentów zginających dla belki wolnopodpartej o długości *l*=10 obciążonej siłą *P*=4 przyłożoną w odległości *x*=3 od lewej podpory, będące przykładowym wynikiem działania programu:



- ***Przykład 3:***

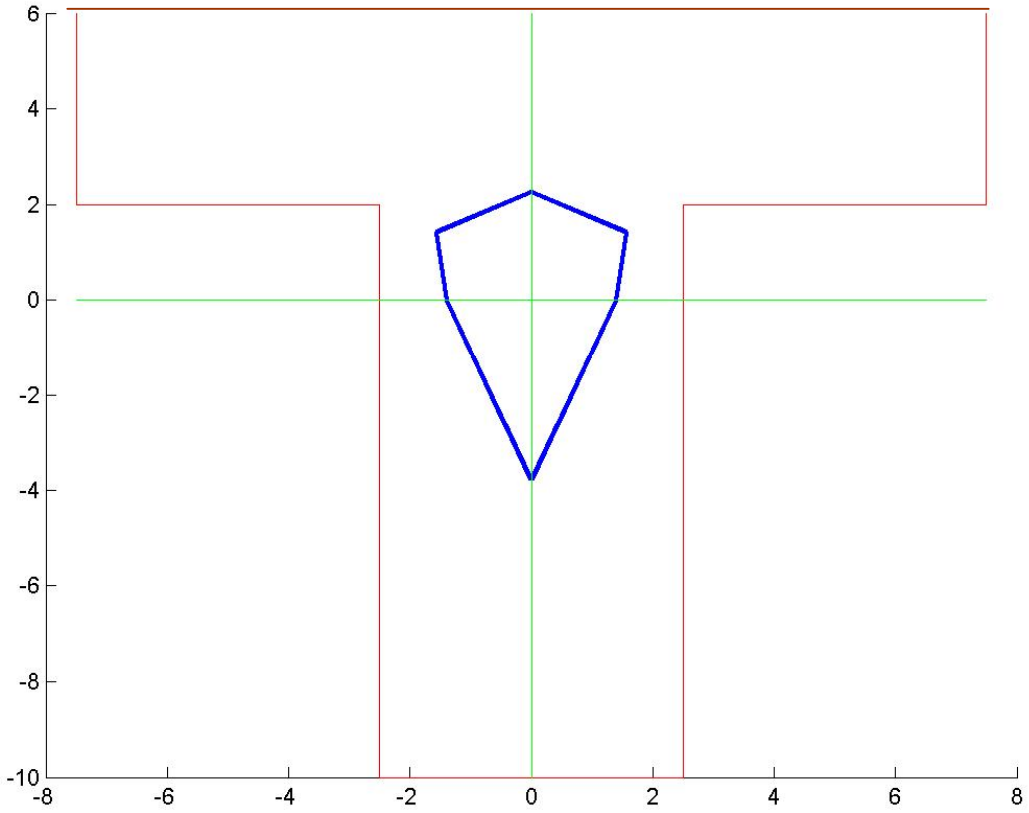
```
%Program oblicza charakterystyki geometryczne i rysuje rdzeń
przekroju teowego
%Dane do programu:
%       h - wysokość przekroju
%       b - szerokość półki
%       t - grubość środnika
%       d - grubość półki
clear
clc
disp('Program rysuje rdzeń przekroju teowego')
disp(' ')

%wprowadzanie danych
h=input('Podaj całkowitą wysokość przekroju h= ');
while h<=0
   disp('     Wysokość musi być wartością dodatnią!')
   h=input('Podaj całkowitą wysokość przekroju h= ');
end
b=input('Podaj szerokość półki b= ');
while b<=0
   disp('     Szerokość musi być wartością dodatnią!')
   b=input('Podaj szerokość półki b= ');
end
t=input('Podaj grubość środnika t= ');
while t<=0 | t>=b
   disp('     Grubość środnika musi być wartością dodatnią i
mniejszą od szerokości półki!')
   t=input('Podaj grubość środnika t= ');
end
d=input('Podaj grubość półki d= ');
while d<=0 | d>=h
   disp('     Grubość półki musi być wartością dodatnią i
mniejszą od wysokości przekroju!')
   d=input('Podaj grubość półki d= ');
end

%charakterystyki geometryczne przekroju
disp(' ')
disp('Pole powierzchni:')
A=b*d + (h-d)*t
Sx=b*d*d/2 + (h-d)*t*(d+(h-d)/2);
disp('Odległość środka ciężkości od góry przekroju')
yc=Sx/A
disp('Momenty bezwładności:')
```

```
Ix=b*d^3/12 + b*d*(yc-d/2)*(yc-d/2) + t*(h-d)^3/12 + t*(h-
d)*(d+(h-d)/2-yc)*(d+(h-d)/2-yc)
Iy=d*b^3/12 + (h-d)*t^3/12
disp('Kwadraty promieni bezwładności:')
ix2=Ix/A
iy2=Iy/A

%obliczanie wierzchołków rdzenia
u(1)=0;
v(1)=-ix2/yc;
u(2)=-iy2/(b/2);
v(2)=0;
e=(h-d)/(t-b);
x0=(yc+b*e-d)/(2*e);
u(3)=-iy2/x0;
y0=yc+b*e-d;
v(3)=-ix2/y0;
u(4)=0;
v(4)=-ix2/-(h-yc);
u(5)=-u(3);
v(5)=v(3);
u(6)=-u(2);
v(6)=0;
disp('Współrzędne wierzchołków rdzenia w układzie przechodzącym
przez środek ciężkości przekroju :');
[u' v']

%rysowanie przekroju i rdzenia
clf
x=[-b/2 b/2 b/2 t/2 t/2 -t/2 -t/2 -b/2 -b/2];
y=[yc yc yc-d yc-d yc-h yc-h yc-d yc-d yc];
line(x,y,'Color','red');
u(7)=u(1);
v(7)=v(1);
line(u,v,'LineWidth',2.5)
line([-b/2 b/2],[0 0],'Color','green');
line([0 0],[yc-h yc],'Color','green');

save wyniki.mat
```

- ***Wyniki działania programu:***
  Poniżej przedstawiono charakterystyki geometryczne i rysunek rdzenia przekroju teowego o całkowitej wysokości $h$=16, szerokości półki $b$=15, grubości środnika $t$=5 i grubości półki $d$=4, będące przykładowym wynikiem działania programu:

```
Pole powierzchni:
```

```
A =
    120
Odległość środka ciężkości od góry przekroju
yc =
    6
Momenty bezwładności:
Ix =
    2720
Iy =
    1250
Kwadraty promieni bezwładności:
ix2 =
    22.6667
iy2 =
    10.4167
Współrzędne wierzchołków rdzenia w układzie przechodzącym przez
środek ciężkości przekroju :
ans =
    0         -3.7778
   -1.3889     0
   -1.5625     1.4167
    0          2.2667
    1.5625     1.4167
    1.3889     0
```

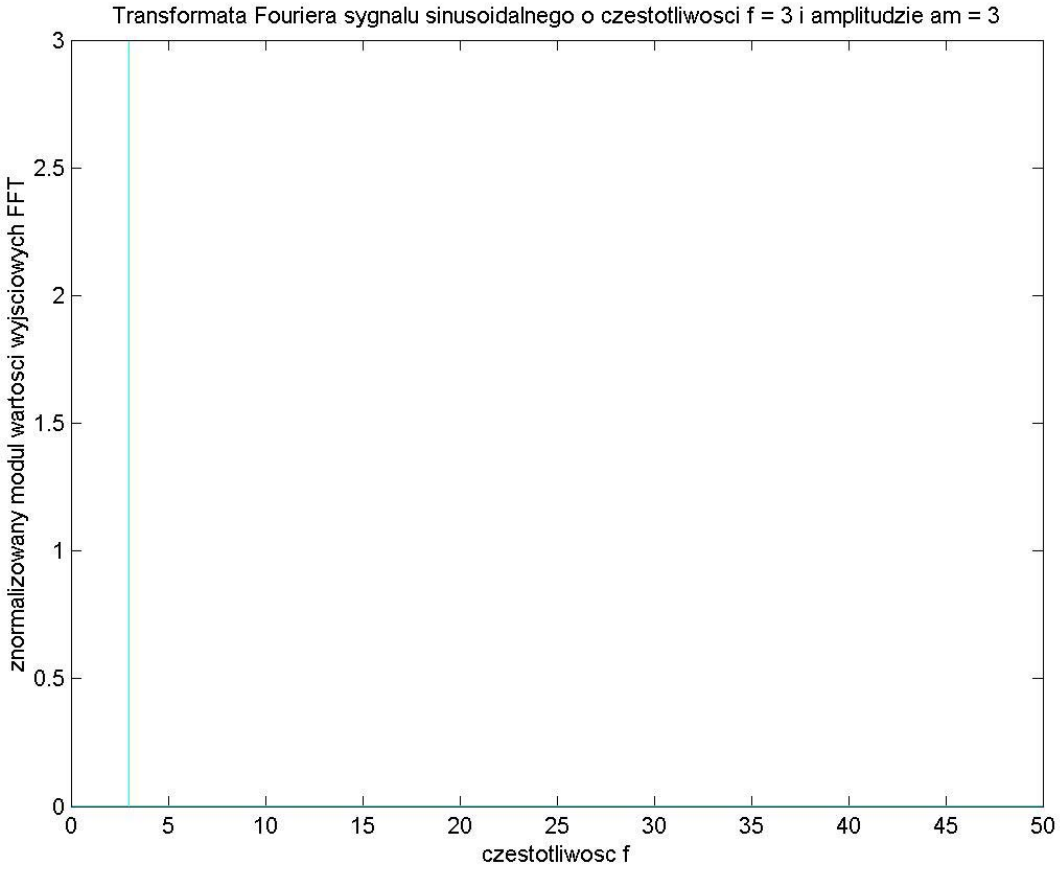



- ***Przykład 4:***

```
%Program oblicza szybka transformatę Fouriera sygnału
sinusoidalnego
%o częstości kołowej definiowanej przez użytkownika.
%Wykorzystuje się funkcję fft.
%Aby uzyskać dobre wyniki należy podzielić przedział czasu
%na dużą liczbę punktów. W programie pokazano jak poprawnie
%wyznaczyć oś częstości, oraz jak obliczyć amplitudę sygnału
%używając wyniku funkcji fft.


clear;
clc;

f = input('Podaj częstotliwość f: ');
while f<=0
      disp('     Podaj wartość większą od zera !')
      f=input('Podaj częstotliwość : ');
end
```

```
am = input('Podaj amplitudę sygnału am : ');
while am<=0
      disp('     Podaj wartość większą od zera !')
      am=input('Podaj amplitudę sygnału: ');
end


tk = 1500;%definicja wektora czasu
dt=0.01;%definicja kroku czasowego
t=(0:dt:tk);
n_t=length(t);%wyznaczenie długości wektora czasu

w = 2*pi*f;%obliczenie częstości
x = am*sin(w*t);%generacja sygnału sinusoidalnego

%************************************************************
%obliczenia

fx = fft(x);%obliczenie szybkiej transformaty
fx(1) = [];%usunięcie pierwszego elementu z wektora transf.
nx = length(fx);
base =inv(dt)*(0:(n_t/2-1))/n_t;%wyznaczenie osi częstotliwości
powerx = abs(fx(1:nx/2));%wyznaczenie widma
powerxn = 2*powerx./nx;%normalizacja odpowiedzi


%************************************************************
%wydruk odpowiedzi

plot(base,powerxn);
title(['Transformata Fouriera sygnału sinusoidalnego o
częstotliwości f = 'num2str(f)...
' i amplitudzie am = 'num2str(am)]);
xlabel('częstotliwość f');
ylabel('znormalizowany moduł wartości wyjsciowych FFT');
```

- ***Wyniki działania programu:***
  Poniżej przedstawiono wykresy będące przykładowym wynikiem działania programu:

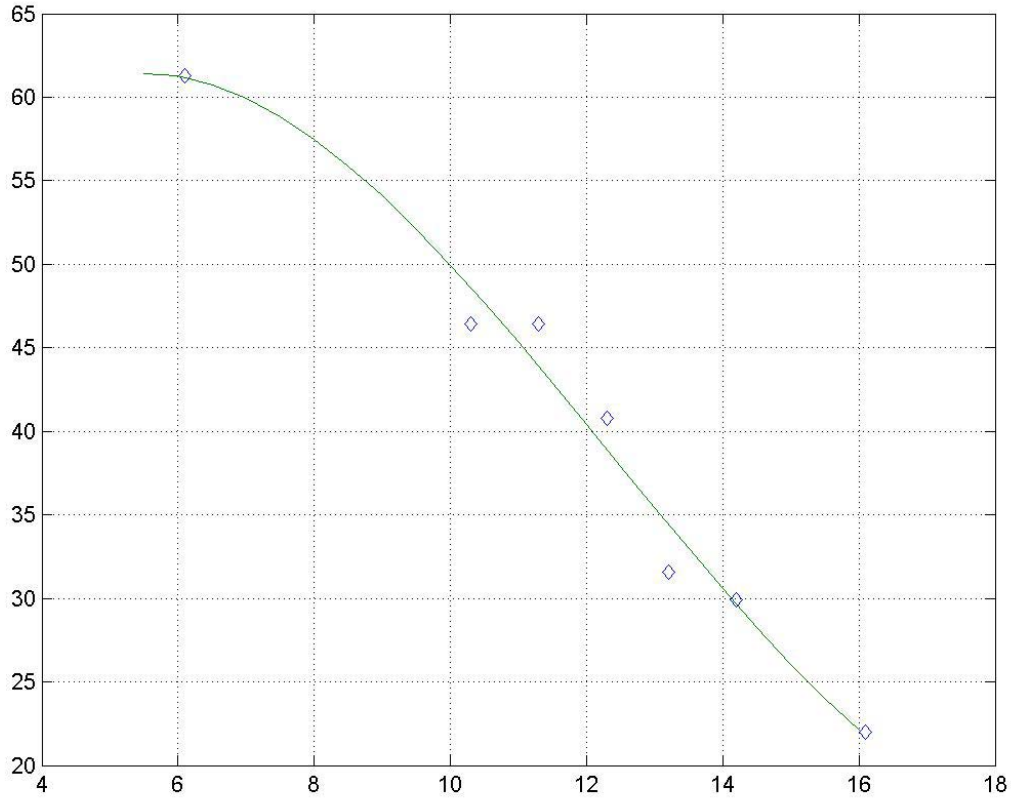



- ***Przykład 5***

```
%Program wyznacza współczynniki wielomianu aproksymującego
%zbiór danych, przy pomocy metody najmniejszych kwadratów.
%Wykorzystuje 2 funkcje:
%Pierwsza funkcja:
%polyfit(x,y,n) – funkcja zwraca współczynniki wielomianu,o
%zadanym stopniu n, „wpasowanego” w zbiór danych (x,y).
%Współczynniki te są ustawione od najwyższej potęgi.
%Druga funkcja:
%polyval(f,xi) – funkcja zwraca wartość wielomianu o
%współczynnikach zapisanych w tablicy f, w zadanym punkcie xi

clear
clc
x = [6.1 10.3 11.3 12.3 13.2 14.2 16.1];%zbiór rzędnych
y = [61.3 46.4 46.4 40.8 31.6 29.9 22];%zbiór wartości
n = 3;%przyjęty stopień wielomianu;
f = polyfit(x,y,n);%wyznaczenie tablicy współczynników
wielomianu f(x)
%wykreślenie wykresu wielomianu f(x)
p = min(x);%początek zbioru rzędnych xn
```

```
k = max(x);%koniec zbioru rzędnych xn
dx = (k-p)/50;%krok podziału zbioru xn
xn = (p:dx:k);%
for i=1:length(xn)
yn(i)=polyval(f,xn(i));% zbiór wartości wielomianu (fx)
end

plot(x,y,'rd',xn,yn)% wykreślenie wykresu
```

- ***Wyniki działania programu:***
  Poniżej przedstawiono wynik działania programu:



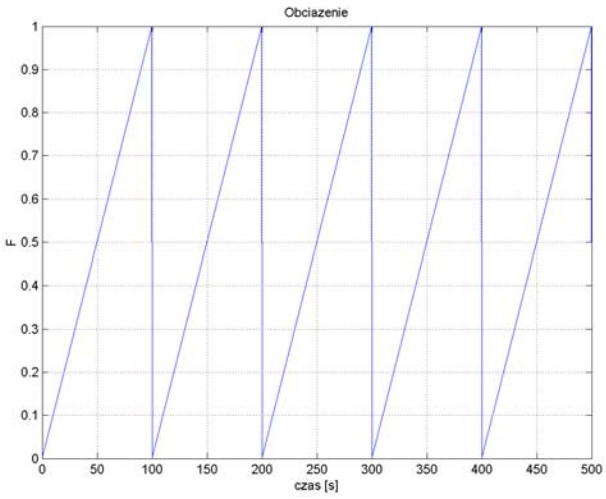

- ***Przykład 6:***

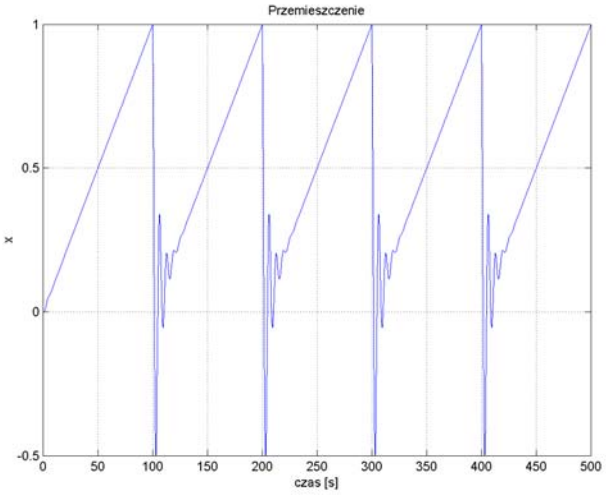
```
%UWAGA !Program ten program wymaga toolboxie %Control.
%Program oblicza odpowiedź układu dynamicznego o jednym stopniu
%swobody na trzy różne przypadki obciążenia:
%sinusoidalne, tzw "zęby piły" oraz impuls.
%Układ dynamiczny opisany jest liniowym równaniem różniczkowym
%w postaci: x''+2*ksi*w*x'+w^2*x=F, gdzie w jest częstością
%kołowa, ksi bezwymiarowym wsp. tłumienia a F obciążeniem.
%Program wykorzystuje procedurę lsim dostępną w toolboxie
```

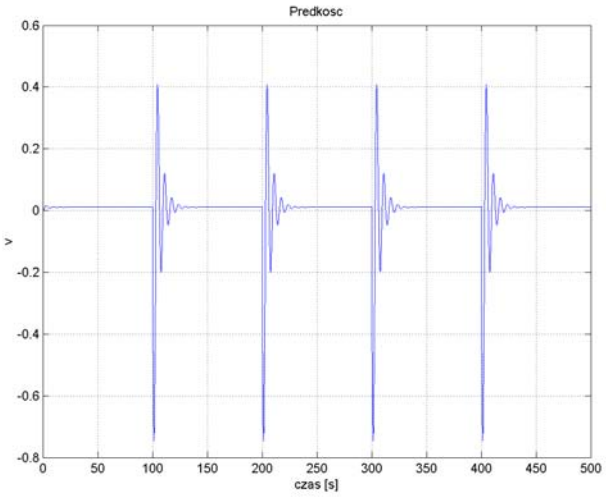
```
%Control.
%Procedura wymaga przepisania równania różniczkowego rzędu
%drugiego jako układu dwóch równań różniczkowych rzędu
%pierwszego (x'=Ax+BF, y=Cx+Du).W analizowanym przypadku A jest
%macierzą 2x2, B jest wektorem o wymiarze 2x1, C jest wektorem
%o wymiarze 1x2 a D jest wektorem 1x1. W programie macierze te
%są generowane jawnie.
%Można jednak wygenerować je automatycznie używając funkcji
%ord2 w postaci [a,b,c,d] = ord2(w,ksi) dostępnej w toolboxie
%Control.

clear
clc
%wybór funkcji obciążenia

o=menu('Wybierz funkcje obciążenia','Obciążenie F=sin(w*t)',...
    'Obciążenie "zęby piły"','Obciążenie impulsem');

disp(' ')
w=input('Podaj częstość kołową układu : ');
ksi=input('Podaj współczynnik tłumienia : ');

while w<=0
      disp('     Podaj wartość większą od zera !')
      w=input('Podaj częstość kołową : ');
end

dt = 0.1;% definicja kroku całkowania
T = (0:dt:500);%definicja wektora czasu
F = zeros(1,length(T));%inicjacja wektora obciążenia
X0 = [0 0]';%wektor warunków początkowych

if (o==1)%generacja obciążenia sinusoidalnego
   F = sin(w*T);

elseif (o==2)%generacja obciążenia typu "zęby piły"

   for i=0:4
       for z=1:1000
        F(z+i*1000)=0.001*z;
       end
   end

elseif (o==3)%w sekundzie dt*1000=100 uderzenie impulsem o
%wart.1

   F(1000)=1;
```

```
end

%*************************************************************
%generacja macierzy układu równań

a = [0 1;-w^2 -2*ksi*w];

b = [0 1]';

c = [1 0];

d = [0];

%*************************************************************
%obliczenia

[Yrozw, Xrozw] = lsim(a,b,c,d,F,T,X0);

%*************************************************************
%drukowanie odpowiedzi
figure(1);
plot(T,F);
grid;
title('Obciążenie');
xlabel('czas [s]');
ylabel('F');


figure(2);
plot(T,Xrozw(:,1));
grid;
title('Przemieszczenie');
xlabel('czas [s]');
ylabel('x');


figure(3);
plot(T,Xrozw(:,2));
grid;
title('Prędkość');
xlabel('czas [s]');
ylabel('v');
```

- ***Wyniki działania programu:***
  Poniżej przedstawiono wykresy będące przykładowym wynikiem działania programu:


Obciazenie
F
czas [s]


Przemieszczenie
x
czas [s]


Predkosc
v
czas [s]

## Funkcje matematyczne

| | |
|---|---|
| `sin(x)` | sinus |
| `cos(x)` | cosinus |
| `tan(x)` | tangens |
| `asin(x)` | arcus sinus |
| `acos(x)` | arcus cosinus |
| `atan(x)` | arcus tangens |
| `sinh(x)` | sinus hiperboliczny |
| `cosh(x` | cosinus hiperboliczny |
| `tanh(x` | tangens hiperboliczny |
| `asinh(x)` | arcus sinus hiperboliczny |
| `acosh(x` | arcus cosinus hiperboliczny |
| `atanh(x)` | arcus tangens hiperboliczny |
| `sqrt(x)` | Pierwiastek kwadratowy |
| `exp(x)` | $e^{x_{ij}}$ |
| `log(x)` | Logarytm naturalny |
| `log2(x)` | Logarytm przy podstawie 2 |
| `log10(x)` | Logarytm przy podstawie 10 |

*Funkcje związane z obliczeniami w dziedzinie liczb zespolonych*

| | |
|---|---|
| `abs(x)` | Macierz modułów elementów macierzy **x** |
| `angle(x)` | Macierz argumentów elementów macierzy **x** |
| `real(x)` | Macierz części rzeczywistych elementów macierzy **x** |

| | |
|---|---|
| `imag(x)` | Macierz części urojonych elementów macierzy **x** |
| `conj(x)` | Macierz o elementach sprzężonych z elementami macierzy **x** |

*Funkcje dodatkowe*

| | |
|---|---|
| `round(x)` | Zaokrągla elementy macierzy **x** do najbliższej liczby całkowitej |
| `rem(x,y)` | Oblicza resztę z dzielenia odpowiadających sobie elementów macierzy **x** i **y** |
| `gcd(a,b)` | Oblicza największy wspólny dzielnik liczb *a* i *b* |
| `lcm(a,b)` | Oblicza najmniejszą wspólną wielokrotną liczb *a* i *b* |