

Sprawozdanie z zajęć
laboratoryjnych z przedmiotu
„Synteza Układów Sterowania”

Wykonał:

*Radosław Łochowski
Automatyka i Robotyka
rok 2 grupa 2*

Zadanie 1:

Napisać m-plik funkcyjny przyjmujący trzy argumenty:

- ilość wierszy macierzy;*
- ilość kolumn macierzy;*
- liczbę dodatkową;*

M-plik powinien zwracać macierz.

Rozwiązanie:

„zadanie1.m” będzie miał następującą składnię:

```
function a=zadanie1(m,n,d);
for i=1:m
for j=1:n
if i+j==m
a(i,j)=d;
else
if i+j>=m+1 && i+j<=m+n
a(i,j)=0;
else
a(i,j)=j;
end
end
end
end
end
```

W „Matlabie” wywołujemy ją następująco, otrzymując poniższy wynik:

```
>> zadanie1(6,5,22)
```

```
ans =
```

```
1  2  3  4  22
1  2  3  22  0
1  2  22  0  0
1  22  0  0  0
22  0  0  0  0
0  0  0  0  0
```

```
>>
```

Zadanie 2:

Rozwiązać układ równań:

$$\begin{cases} 12x - 12y = 1 \\ 12x + 2y = 7 \end{cases}$$

Rozwiązanie:

```
>> [x,y] = solve('12*x-12*y=1','12*x+2*y=7')
```

```
x =
```

```
43/84
```

```
y =
```

```
3/7
```

W ułamkach dziesiętnych, wynik jest następujący:

```
>> 43/84
```

```
ans =
```

```
0.5119
```

```
>> 3/7
```

```
ans =
```

```
0.4286
```

```
>>
```

Zadanie 3:

Znaleźć rozwiązanie równania stopnia trzeciego:

$$2x^3 + 3x^2 + 2 = 0$$

Rozwiązanie:

```
>> rownanie=[2,3,0,2]
```

```
rownanie =
```

```
    2    3    0    2
```

```
>> wynik=roots(rownanie)
```

```
wynik =
```

```
-1.8064  
0.1532 + 0.7281i  
0.1532 - 0.7281i
```

```
>>
```

Program wyliczył trzy pierwiastki równania, w tym dwa zespolone.

Zadanie 4:

Narysować wykres funkcji:

$$y = x^2 - 2x + 1$$

Rozwiązanie:

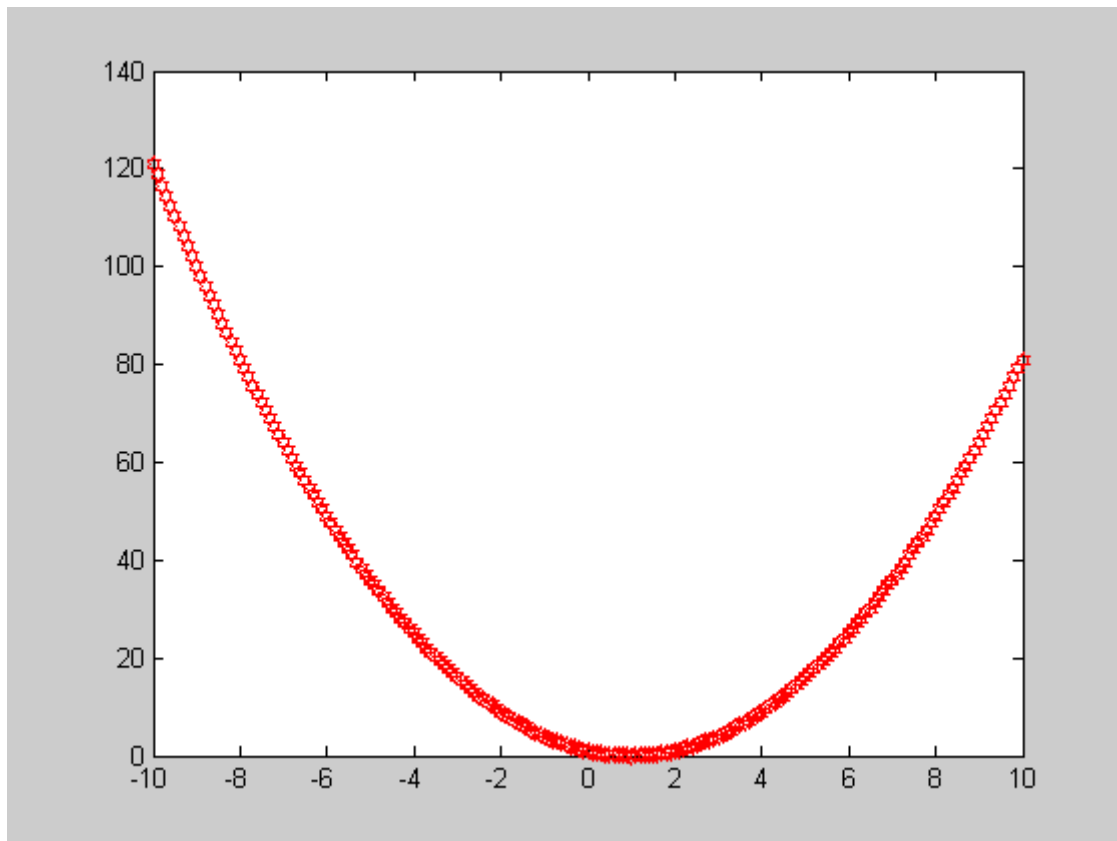
Dla przejrzystości zadania pomijam wypisywanie wyników, dodając na końcu linijek średnik „;”.

```
>> x=-10:0.1:10;
```

```
>> y=x.^2-2*x+1;
```

```
>> plot(x,y,'rh')
```

```
>>
```



Zadanie 5:

Narysować wykres łamany oraz aproksymowany dla danych:

| | | | | | |
|----------|------------|------------|-----------|-----------|-----------|
| <i>X</i> | <i>1</i> | <i>2</i> | <i>5</i> | <i>7</i> | <i>9</i> |
| <i>Y</i> | <i>1,1</i> | <i>4,2</i> | <i>23</i> | <i>50</i> | <i>80</i> |

Rozwiązanie:

Zapisuję dwa wektory X i Y, zgodne z danymi z zadania. Następnie tworzę wykres łamany podpisując go oraz same osie, dodając również siatkę dla lepszej wizualizacji.

```
>> X=[1,2,5,7,9]
```

```
X =
```

```
1 2 5 7 9
```

```
>> Y=[1.1,4.2,23,50,80]
```

```
Y =
```

```
1.1000 4.2000 23.0000 50.0000 80.0000
```

```
>> plot(X,Y)
```

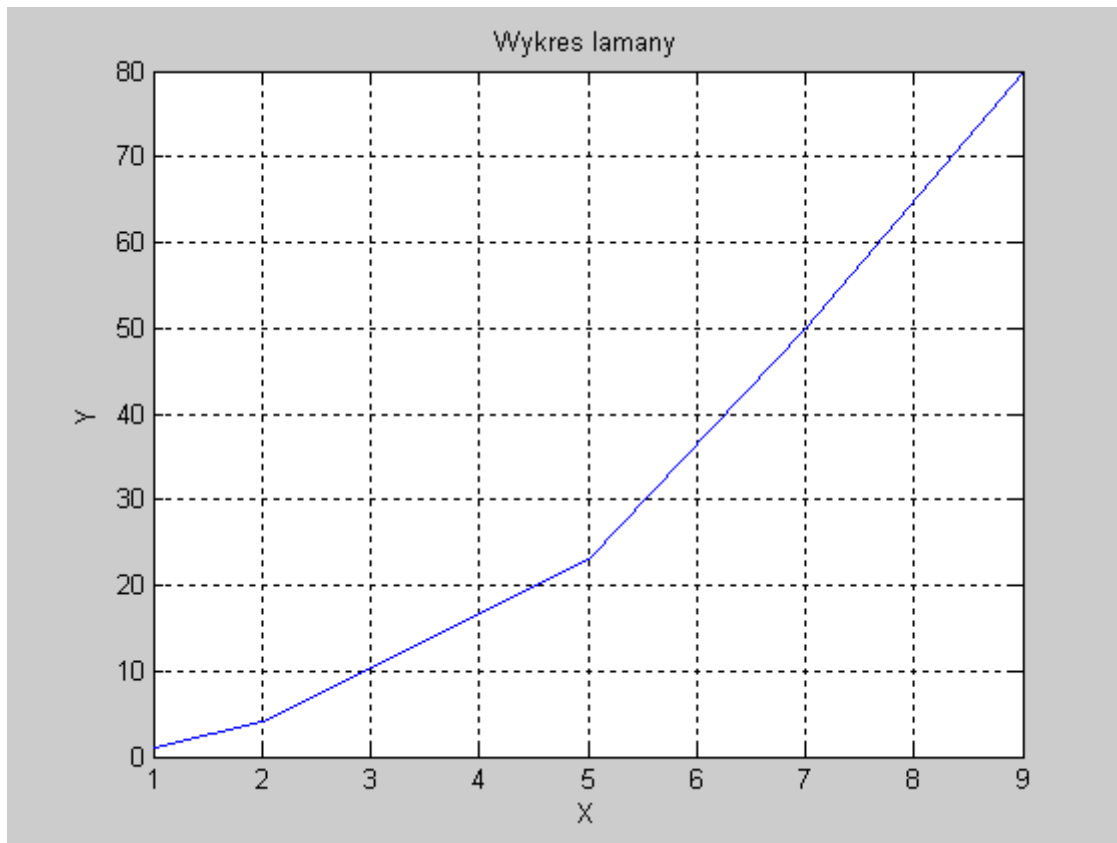
```
>> grid on
```

```
>> title('Wykres łamany')
```

```
>> xlabel('X')
```

```
>> ylabel('Y')
```

```
>>
```



Teraz powstanie wykres aproksymowany dla powyższych wartości (korzystam z zadeklarowanych wcześniej wektorów X oraz Y).

```
>> wspolczyn2stopn=polyfit(X,Y,2)
```

```
wspolczyn2stopn =
```

```
1.0295 -0.3631 0.4584
```

```
>> osiks=0:0.1:9;
```

```
>> wartosci=polyval(wspolczyn2stopn,osiks);
```

```
>> plot(X,Y,'rh',osiks,wartosci)
```

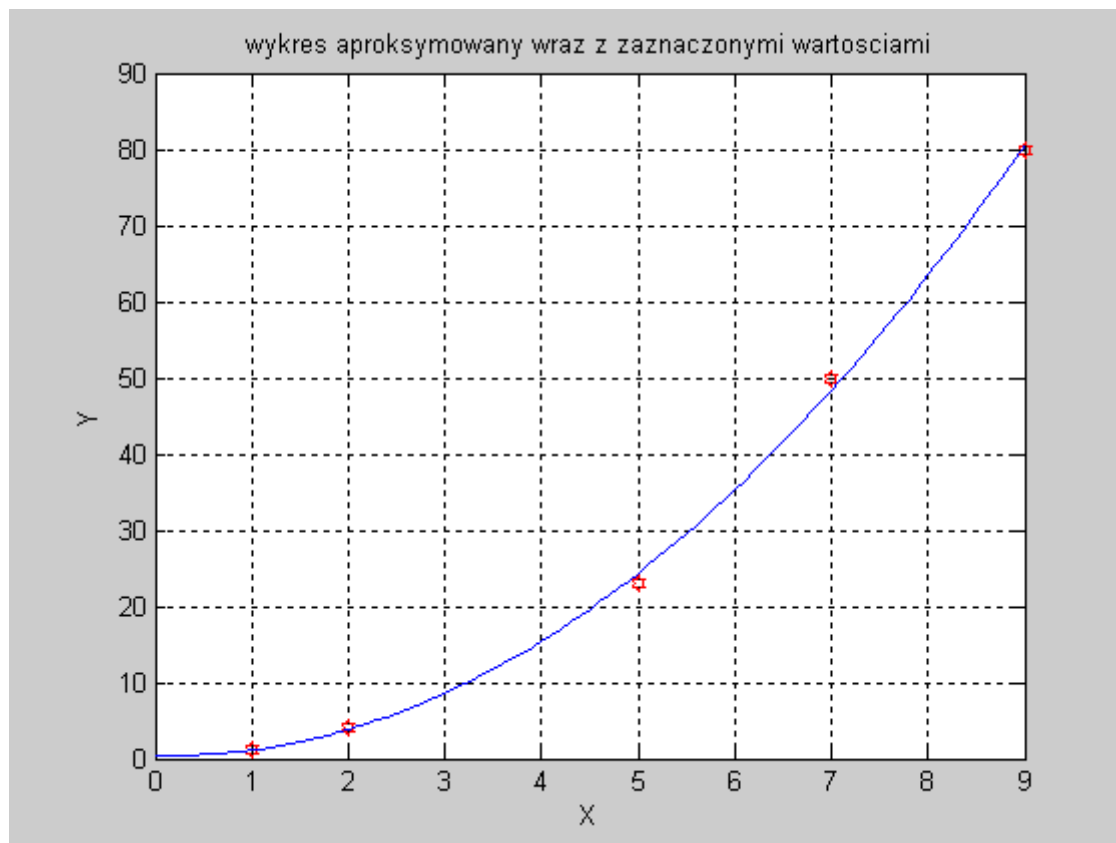
```
>> grid on
```

```
>> title('wykres aproksymowany wraz z zaznaczonymi wartosciami')
```

```
>> xlabel('X')
```

```
>> ylabel('Y')
```

```
>>
```



Zadanie 6:

Dany jest element inercyjny drugiego rzędu o stałych czasowych $T_1 = 2$, $T_2 = 4$ i wzmacnieniu $k = 1$;
- narysować odpowiedź skokową,
- narysować odpowiedź impulsową,
- przekształcić do postaci: zero/biegun/wzmacnienie.

Rozwiązanie:

Tworzę dwa elementy inercyjne rzędu pierwszego. Mnożąc je, otrzymam element inercyjny rzędu drugiego o zadanych parametrach. Wyznaczę postać ze wzmacnieniem, zerami i biegunami. Następnie wykreślę żądane charakterystyki.

```
>> pierwsza=tf([1],[4,1])
```

Transfer function:

```
1
-----
4 s + 1
```

```
>> druga=tf([1],[2,1])
```

Transfer function:

```
1
-----
2 s + 1
```

```
>> inercyjny2=pierwsza*druga
```

Transfer function:

```
1
-----
8 s^2 + 6 s + 1
```

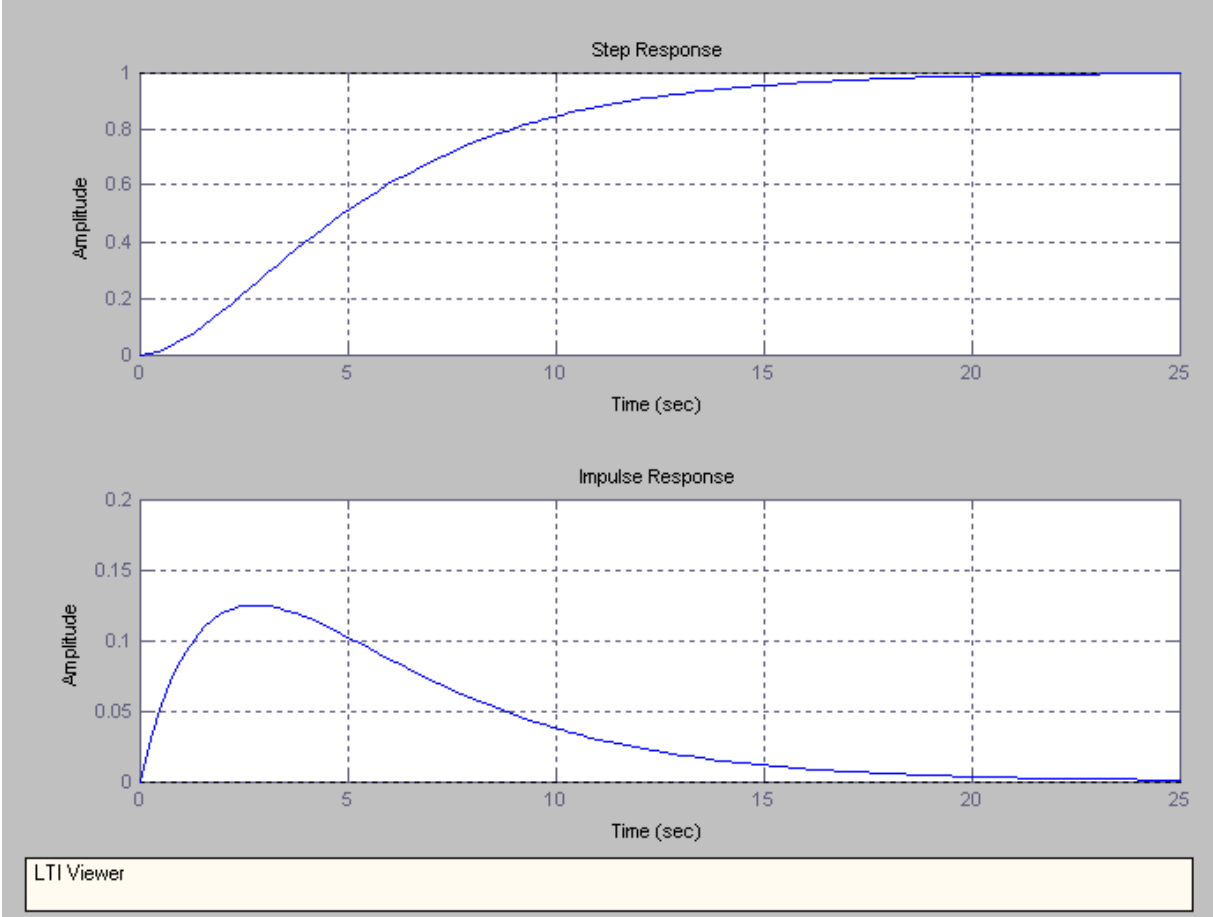
```
>> postacZBW=zpk(inercyjny2)
```

Zero/pole/gain:

```
0.125
-----
(s+0.5) (s+0.25)
```

```
>> ltiview({'step','impulse'},inercyjny2)
```

```
>>
```



Zadanie dodatkowe:

Wykorzystanie instrukcji programu „Matlab” w służbie człowiekowi.

Rozwiązanie:

♣ Program 1

```
function p=czyszczenie
cd c:\ % zmienia katalog roboczy na "c:\"
clc; % czyszczenie okna [command window]
clear; % czyszczenie pamięci z zapisanych wcześniej zmiennych
A=[1,2,3,4;5,6,7,8;8,7,6,5;4,3,2,1]; % wpisanie przykładowej macierzy
p=size(A); % wyliczenie wymiarów macierzy i zwrot wartości
```

♣ Program 2

```
function y = drukuj
help plot; % drukuje pomoc do komendy [plot()]
x=-5:0.1:5; % użycie Colon :
y = x.^2+4;
y2 = x.^2+2*x+7;
plot(x,y,'r',x, y2,'co'); % korzystam z funkcji [plot()]
grid on; % wykorzystanie [grid on]
title('Przykładowy podpis do wykresu') % tworzy podpis do wykresu
xlabel('Etykieta osi X') % podpisanie osi x
ylabel('Etykieta osi Y') % podpisanie osi y
legend('wykres 1', 'wykres 2') % legenda do wykresów
```

♣ Program 3

```
function A=macierz(i);
if i==1
    A=zeros(5); % generuje macierz zawierającą elementy 0
else
    if i==2
        A=ones(4); % generuje macierz zawierającą elementy 1
    else
        A=eye(3); % generuje macierz zawierającą elementy 1 na głównej przekątnej
    end
end
end
end
```

♣ Program 4

```
function koncowa=macierz2(A);
i=sum(A); % oblicza sumę elementów w poszczególnych kolumnach
j=sum(i);
```

```

i=dec2bin(j); % zamienia liczbę dziesiętną na binarną
j=log(i); % oblicza jej logarytm
i=sum(j);
j=log10(i); % oblicza logarytm dziesiętny z sumy logarytmów
if j<=1 % wykorzystanie jednego z operatorów relacji i alternatywy
    i='01001';
else % wykorzystanie "jeżeli nie, to..."
    i='10110';
end
end
koncowa=bin2dec(i); % końcowa liczba jest dziesiętna (konwersja liczby binarnej)

```

♣ Program 5

```

function a=macierze(A);
i=rank(A); % oblicza rząd macierzy
B=eye(5); % tworzy macierz diagonalną
B(3,4)=i; % przypisuje elementowi o indeksach (3,4) wartość rzędu macierzy
pomoc=max(B); % oblicza maksima poszczególnych kolumn
a=max(pomoc'); % oblicza maksimum wiersza
if a>=3 % jeżeli maksimum jest większe od 2, to...
    a=min(pomoc'); % oblicza minimum wiersza
end % kończy działanie alternatywy

```

♣ Program 6

```

function a=przypadek(operacja,n);
switch(operacja) % wykorzystanie operacji [switch()]
    case 'cosinus'
        a=cos(n); % liczenie cosinusa
    case 'sinus'
        a=sin(n); % sinusa
    case 'tangens'
        a=tan(n); % oraz tangensa
    otherwise
        while n<=30 % wykorzystanie komendy [while(), disp()]
            disp('Policz głębie sam!');
            n=n+1;
        end % korzystam z [end]
end

```

♣ Program 7

```

function [wynik]=silnia;
clc; % czyszczenie...
clear; % wszystkiego
n=input('Podaj n: '); % czeka na wpisanie wartości dla zmiennej
if(n<0) % alternatywa
    disp('Liczba mniejsza od zera'); % użycie [disp()]
elseif n==0 % korzystam z [elseif]

```

```

silnia=1;
else % else
    licz=1;
    for i=1:n % wykorzystanie pętli [for()]
        licz=licz*i;
    end % zamknięcie pętli
disp('Wynik:');
wynik=licz;
end

```

♣ Program 8

```

function a=wykres(B,w);
x=tf([2*B,0],[1,2*B,sqrt(w)]); % tworzę transformatę funkcji
ltiview({'step';'impulse';'nyquist';'bode'},x); % tworzenie wykresów

```

♣ Program 9

```

function liczba2=zesp(zespolona);
A=abs(zespolona); % oblicza promień liczby zespolonej za pomocą [abs()]
fi=atan2(imag(zespolona),real(zespolona)); % oblicza kąt fi za pomocą [atan2()]
kat=rad2deg(fi); % zmienia kąt w radianach na stopnie za pomocą [rad2deg()]
liczba2=A*exp(fi*i); % zamienia z powrotem na postać kartezjańską (wykorzystuje exp())

```

♣ Program 10

```

function wynik=zera(wsp1,wsp2);
x=tf([2],[1*wsp1,3/wsp2,wsp1*wsp2/10,wsp1*3]); % transformuje funkcję
pzmap(x); % tworzy obraz zer i biegunów na płaszczyźnie zespolonej
text(-0.5,0.5,'Tu jest tekst'); % wykorzystanie komendy [text()]
bieg=pole(x); % wylicza bieguny funkcji przejścia
disp(bieg);
wynik=zpk(x); % drukuje wynik w postaci zera/bieguny/wzmocnienie

```

♣ Program 11

```

function a=magia(mac);
if mac==0 % jeżeli nie podamy żadnej konkretnej macierzy
    x=magic(45); % wykonuje "magiczną" macierz
    imagesc(x); % pokazuje wartości w macierzy w postaci kolorów. Najmniejsza liczba jest
    najzimniejszym kolorem, a najwyższa- najcieplejszym
else
    imagesc(mac); % pokaże naszą macierz w postaci barwnego obrazu
end

```

♣ Program 12

```

function a=oplacalnosc; % nie jeździć samochodem!
clc;
benz=input('Cena benzyny:\n');

```

```

gaz=input('Cena gazu:\n');
olej=input('Cena oleju silnikowego (4litry):\n');
dodatkowe=input('Koszt OC oraz przeglądu technicznego:\n');
rocznie=input('Rocznie przejeżdżasz kilometrów:\n');
przej=input('Liczba przejechanych w ostatnim czasie kilometrów:\n');
disp('Koszt:');
km1=0.8*gaz*10+0.2*benz*8.5+olej/40+dodatkowe/(rocznie/100);
km2=8.5*benz+dodatkowe/(rocznie/100);
km3=km1*przej/100;
v=round(km3); % zaokrągla do najbliższej całkowitej
disp('100km na zasilaniu gazem');
disp(km1);
disp('100km na zasilaniu benzyną');
disp(km2);
disp('Ostatnio zaplaciles');
disp(v);

```

♣ Program 13

```

function a=dziennik
get(0,'diary') % odczytuje aktualny stan dziennika
diary on; % włącza go, jeśli nieaktywny
diary('c:\dziennik.txt'); % zapisuje go na dysku w postaci pliku "dziennik.txt"

```

♣ Program 14

```

function x=sortowanie(A);
b=sort(A,1); % sortowanie wierszami macierzy
v=det(b); % oblicza wyznacznik macierzy
if v>=1 && v<=100 % wykorzystanie relacji i operatora logicznego
    disp(v);
    clc;
    lookfor('DIALOGMANAGER'); % szuka wśród m-plików zawierający łańcuch
"DIALOGMANAGER"
else
    disp(v);
    b=abs(v); % liczy wartość bezwzględną liczby
    disp(b);
    A=eye(5)+3; % tworzy macierz diagonalną i dodaje do niej 3
    v=log(b); % oblicza logarytm
    A(1,1)=v; % przypisuje elementowi (1,1) wartość logarytmu
    disp(A);
    A(5,3)=A(3,3)*3;
    A(1,:)=mean(A); % przypisuje pierwszemu wierszowi wartości średnich poszczególnych
kolumn
    disp(A);
end

```