Grafika dwuwymiarowa

Najprostszą, a zarazem najczęściej wykorzystywaną funkcją do przedstawiania danych w sposób graficzny w języku MATLAB jest funkcja *plot (x,y)*, czyli wykreślenie przez program funkcji y=f(x).Zakres osi jest dobierany automatycznie, jeśli jednak użytkownik chce, to może zmienić zakres stosując funkcję *axis([xmin xmax ymin ymax])*, poszerzając lub ograniczając obszar rysowania wykresu.

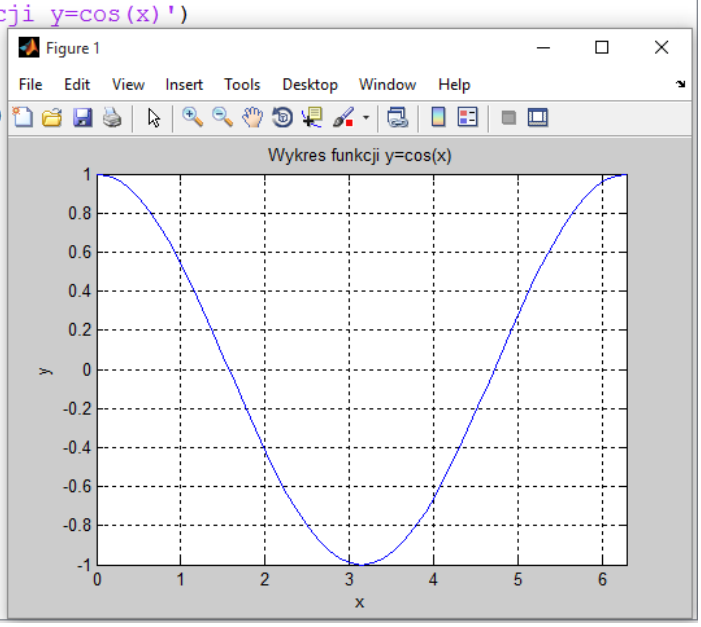
Innymi ważnymi funkcjami wykorzystywanymi przy tworzeniu skryptów używających grafiki w MATLAB-ie są następujące:

clf -czyści okno graficzne
close -zamyka okno graficzne

MATLAB posiada również opcje opisywania wykresów. Służą do tego następujące funkcje:

title('tekst') -tytuł wykresu
xlabel('tekst') -oznaczenie osi x
ylabel('tekst') -oznaczenie osi y
grid on/off -włącza/wyłącza siatkę
text(x,y,'tekst') -umieszcza *tekst* na wykresie w miejscu o współrzędnych *x* i *y*

```
1 -    x=0:pi/25:2*pi; %im mniejszy krok, tym większa dokładność wykresu
2 -    y=cos(x);
3 -    plot(x,y)
4 -    grid on
5 -    title('Wykres funkcji y=cos(x)')
6 -    xlabel('x')
7 -    ylabel('y')
8 -    axis([0 6.28 -1 1])
9 -    pause
10 -   close
```



Przykład zastosowania powyższych funkcji

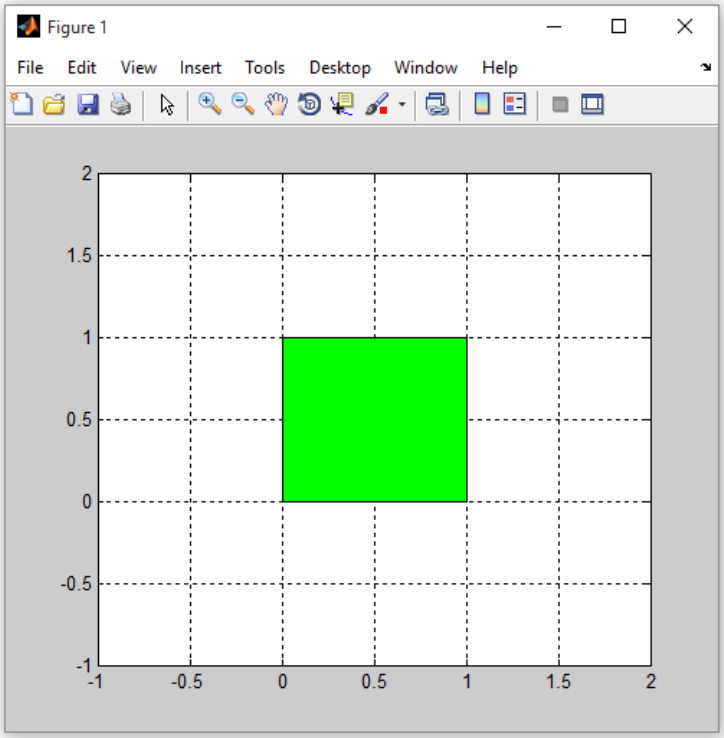
Innym sposobem na generowanie grafiki dwuwymiarowej w MATLAB-ie jest rysowanie linii i wielokątów, służą do tego funkcje:

line(x,y) -rysuje linię łączącą kolejne punkty o współrzędnych *x* i *y*

fill(x,y,'color') -rysuje wielokąt o wierzchołkach w punktach o współrzędnych *x* i *y*, a następnie wypełnia ten wielokąt kolorem. Zamiast *color* należy wpisać jeden z symboli kolorów:

k	-czarny
w	-biały
c	-cyjan
y	-żółty
m	-magenta
r	-czerwony
g	-zielony
b	-niebieski

```
1 —     line([0 0 1 1 0],[0 1 1 0 0])
2 —     fill([0 0 1 1],[0 1 1 0],'g')
3 —     axis([-1 2 -1 2])
4 —     grid on
```

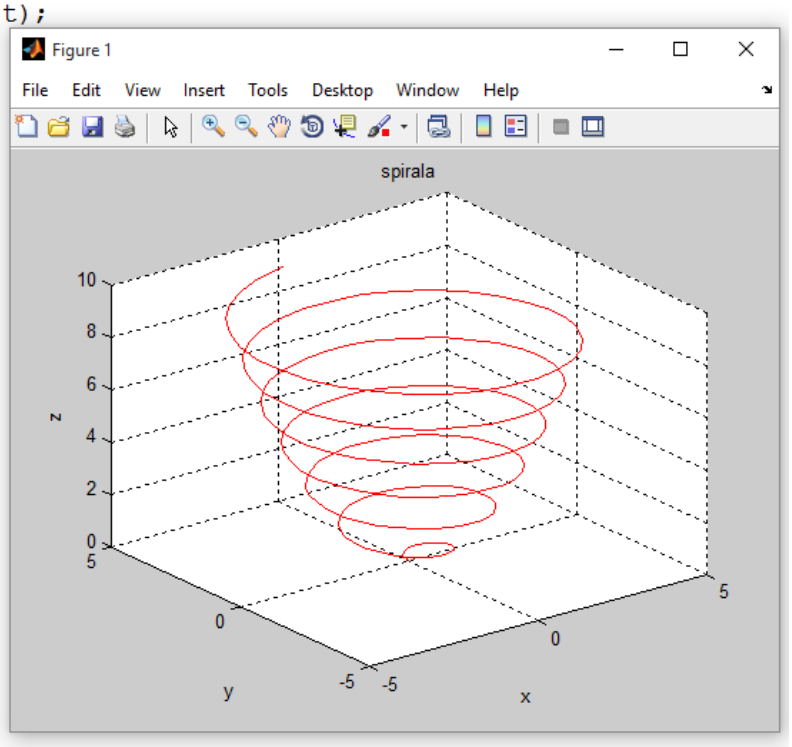


Przykład rysowania wykorzystując funkcje line i fill

Grafika trójwymiarowa

Analogicznie jak przy grafice dwuwymiarowej, tak i w tym przypadku najprostszym sposobem do rysowania wykresów, wykorzystuje się funkcję *plot3(x,y,z)*. Jak widać na poniższym przykładzie, można tu wykorzystywać funkcje takie same, bądź analogiczne do tych, używanych przy grafice dwuwymiarowej.

```
1 -    t=0:0.1:6*pi;
2 -    x=sin(2*t).*sqrt(t);
3 -    y=cos(2*t).*sqrt(t);
4 -    z=0.5*t;
5 -    plot3(x,y,z,'r')
6 -    grid on
7 -    xlabel('x');
8 -    ylabel('y');
9 -    zlabel('z');
10 -   title('spirala');
```



Przykład wykorzystania funkcji plot3

Jeżeli jednak użytkownik chce wykreślić jakieś powierzchnie w środowisku trójwymiarowym, to należy najpierw wyznaczyć płaszczyznę x y, w obrębie której będzie znajdowały się wykreślone powierzchnie.

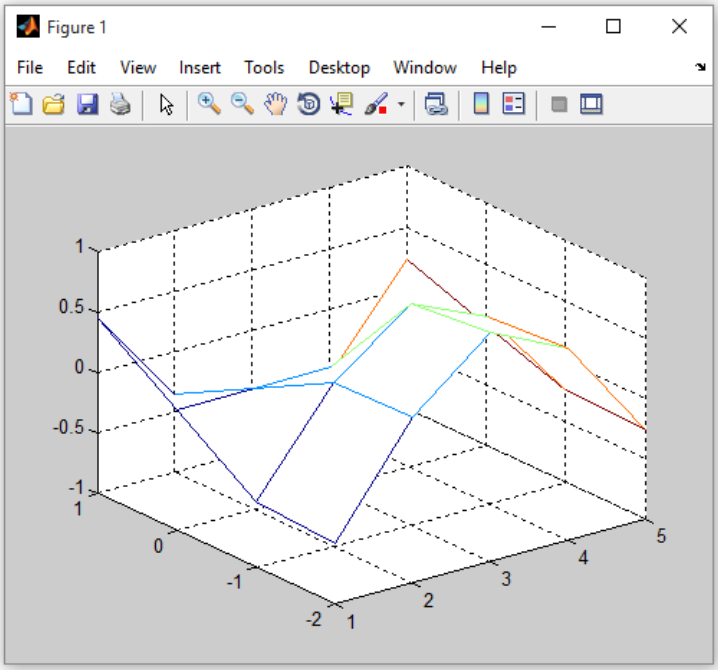
```
>> x=1:5;
>> y=-2:1;
>> [X,Y]=meshgrid(x,y)

X =

     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5


Y =

    -2    -2    -2    -2    -2
    -1    -1    -1    -1    -1
     0     0     0     0     0
     1     1     1     1     1

>> [X,Y]=meshgrid(1:5,-2:1)

X =

     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5


Y =

    -2    -2    -2    -2    -2
    -1    -1    -1    -1    -1
     0     0     0     0     0
     1     1     1     1     1
```

Dwa sposoby utworzenia płaszczyzny x y

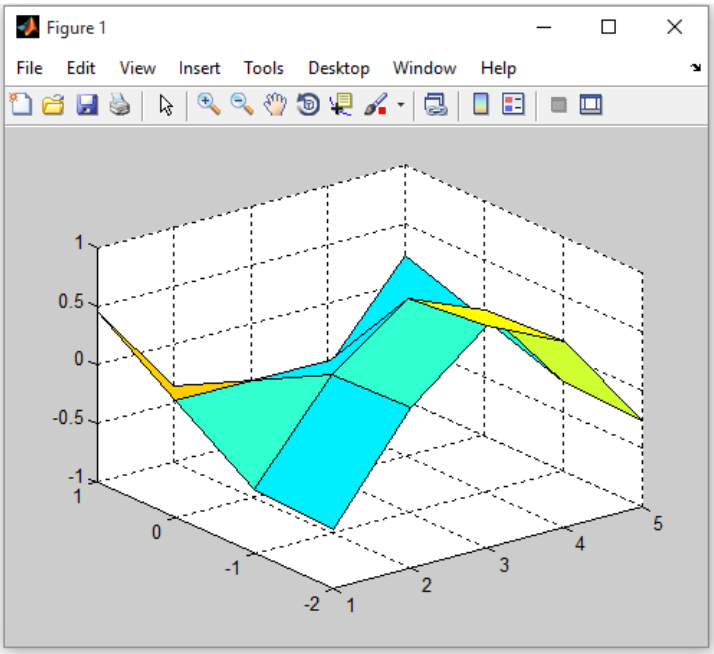
Po utworzeniu tej płaszczyzny można użyć funkcji *surf* lub *mesh* aby stworzyć grafikę trójwymiarową. Funkcja *mesh(X,Y,Z,c)* tworzy siatkę, łączącą punkty funkcji f(x,y) o kolorze *c* (kolory

są zależne od danych, jeżeli zostanie pominięty, to program przyjmuje c=Z), natomiast funkcja *surf(x,y,z)* tworzy siatkę, łączącą punkty funkcji f(x y) i koloruje przestrzenie między punktami. Można również użyć funkcji *surfl* z taką samą składnią, aby uwzględnić na powierzchni odbicie światła.

```
x=1:0.5:5;
y=-2:0.5:1;
Z=cos(X).*sin(Y);
c=X;
mesh(X,Y,Z,c)
```



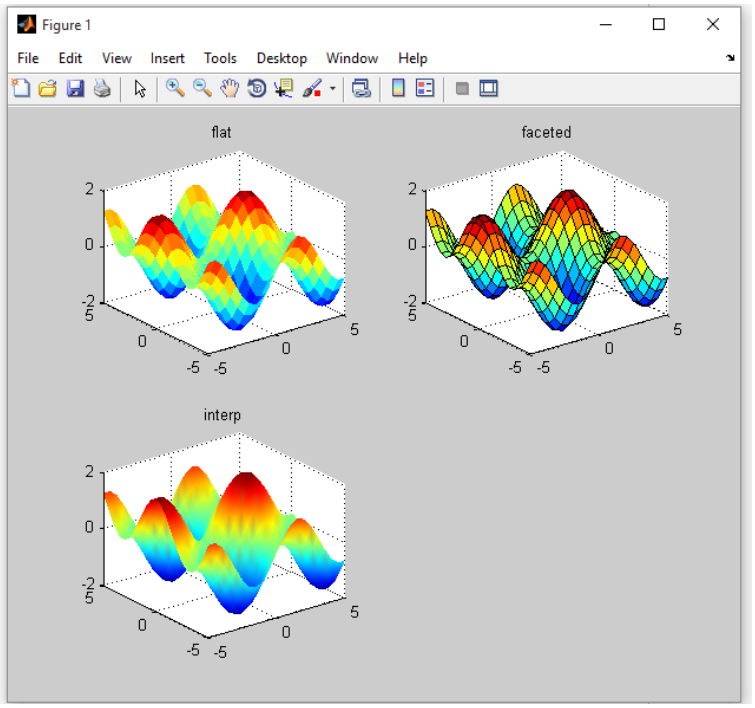

```
x=1:0.5:5;
y=-2:0.5:1;
Z=cos(X).*sin(Y);
surfl(X,Y,Z)
```



Przykłady użycia funkcji mesh i surfl

Możliwe jest również cieniowanie wykresów wykorzystując funkcję *shading*. Poniższy przykład ukazując również tworzenie kilku wykresów w jednym oknie graficznym, przedstawiając funkcję *subplot(pion,poziom,numer)*, gdzie: *pion* – liczba wykresów w pionie, *poziom* – liczba wykresów w poziomie, *numer* – numer wykresu.

```
1 -  [x,y]=meshgrid(-5:0.5:5);
2 -  z=sin(x)+cos(y);
3    %Cieniowanie flat
4 -  subplot(2,2,1)
5 -  surf(x,y,z)
6 -  shading flat
7 -  title('flat')
8    %Cieniowanie faceted
9 -  subplot(2,2,2)
10 - surf(x,y,z)
11 - shading faceted
12 - title('faceted')
13   %Cieniowanie interp
14 - subplot(2,2,3)
15 - surf(x,y,z)
16 - shading interp
17 - title('interp')
```



Przykład cieniowania wykresów

Graficzne metody prezentacji danych i wyników obliczeń

Najczęściej spotykanym sposobem graficznej prezentacji danych w języku MATLAB jest dwuwymiarowy 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.

Przy zmiennych bardziej złożonych wykorzystuje się grafikę trójwymiarową. 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**

Aby móc wygenerować znane postaci funkcji matematycznych (szczególnie trygonometrycznych) w MATLAB posługujemy się predefiniowanymi w systemie poleceniami (tab. 4).

Tab. 4. 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
`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

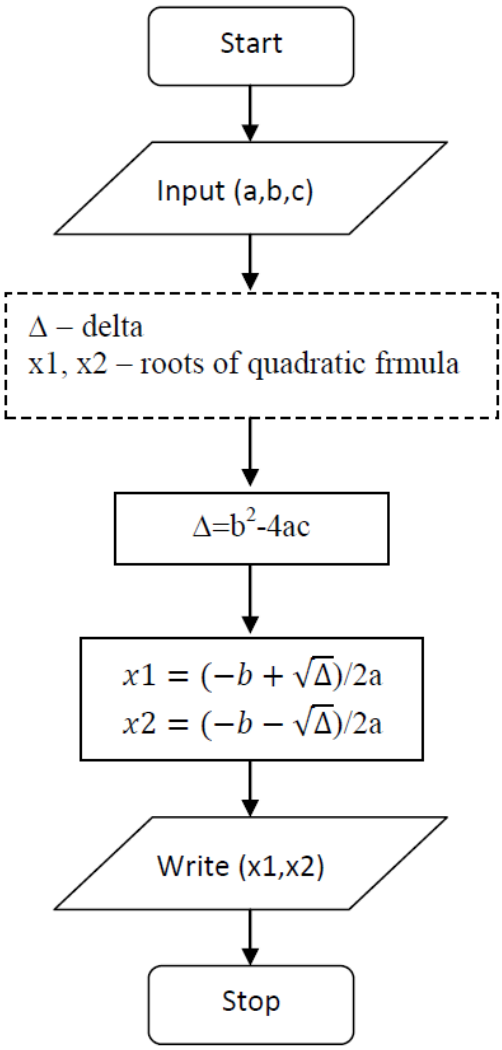
Algorytmy

Algorytm to jednoznaczny przepis prowadzący do rozwiązania zadania. Algorytm zapisany w języku programowania to program.

Cechy algorytmu:

- poprawność – poprawne wyniki dla dowolnych danych wejściowych,
- skończoność – uzyskanie wyniku po skończonej liczbie operacji,
- sprawność – uzyskanie wyniku w możliwie najkrótszym czasie i w możliwie najmniejszej ilości pamięci.

Jako przykład algorytmu zostanie wykorzystane zadanie rozwiązania równania kwadratowego ax2+bx+c=0.

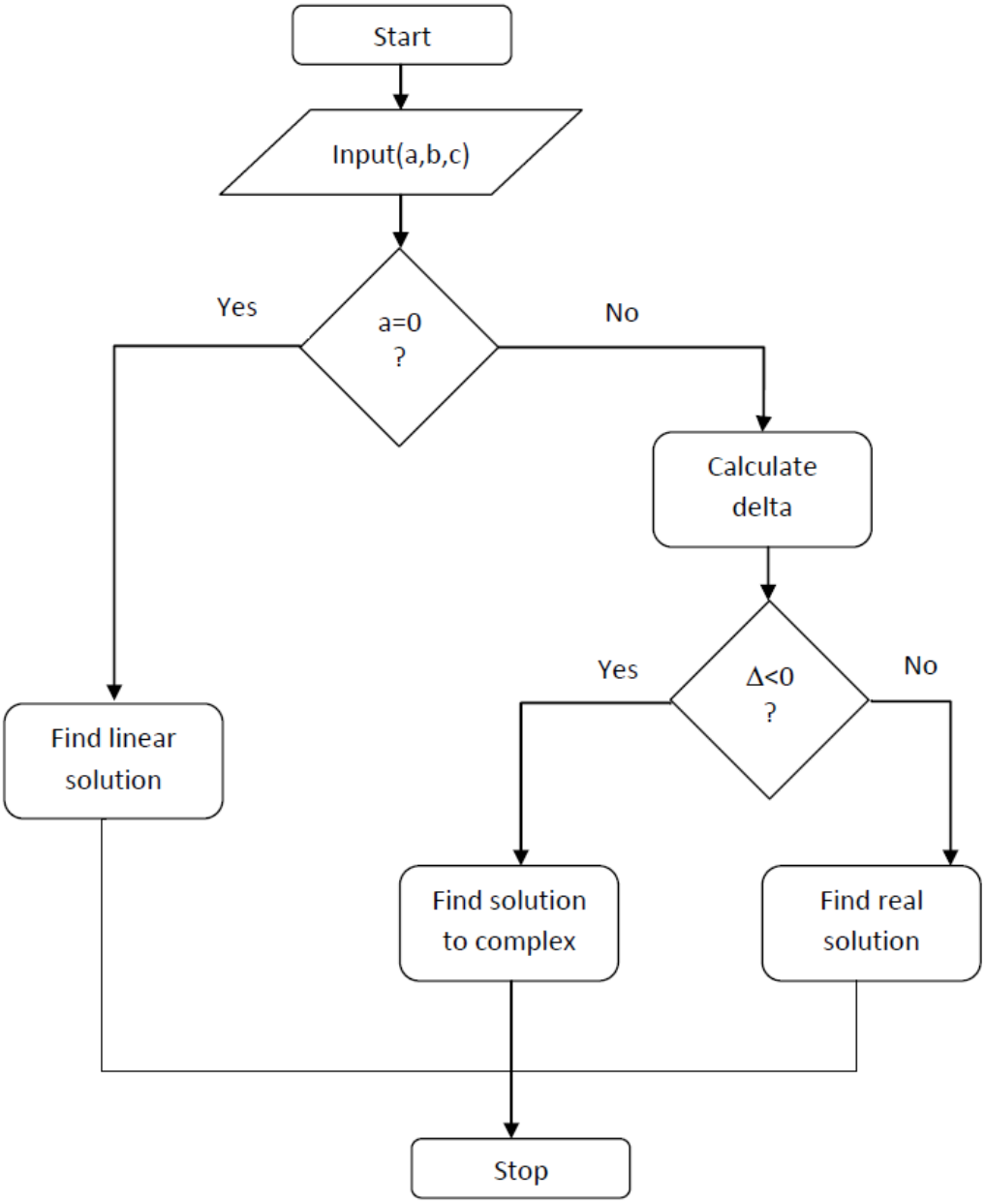


Algorytm wyznaczania równania kwadratowego (ogólny)

Ten program jest niepoprawny, ponieważ istnieją takie a, b i c, dla których nie ma rozwiązania:

- np. a=0 – błąd dzielenia przez 0,
- a=1, b=0, c=1 – rozwiązanie w dziedzinie liczb zespolonych (pierwiastek z liczby ujemnej)

Poprawną budowę algorytmu można przedstawić następująco (budowa modułowa)



Algorytm wyznaczania równania kwadratowego (prawidłowy)

Instrukcje i funkcje w MATLAB

W języku MALAB do wykonywania złożonych obliczeń np. iteracyjnych powszechnie wykorzystuje się instrukcje i funkcje. Zastosowanie tych niezwykle przydatnych narzędzi zostanie wytłumaczone za pomocą zadań rozwiązujących konkretne zadania inżynierskie.

Zad. 1. Policzyć charakterystyki geometryczne dla figury płaskiej złożonej z „n" elementów prostokątnych o wymiarach b_i oraz h_i (rys.1). charakterystyki te opisane są następującymi wzorami:

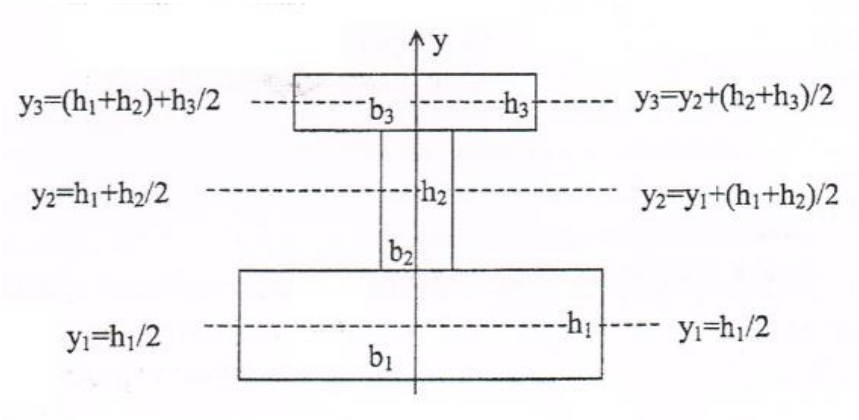
a) pole figury $A = \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} b_i\, h_i$

b) moment statyczny $S = \sum_{i=1}^{n} y_i A_i$ gdzie $y_1 = h_1/2$; $y_i = \sum_{k=1}^{i-1} h_k + \frac{h_i}{2} = y_{i-1} + \frac{(h_{i-1}+h_i)}{2}$;

c) środek ciężkości $y_c = S/A$;

d) moment bezwładności $I = I_o + I_{st} = \sum_{i=1}^{n} \frac{b_i h_i^3}{12} + \sum_{i=1}^{n} d_i^2 A_i$; gdzie $d_i = y_i - y_c$

Graficzny schemat obliczeniowy przedstawiono na rys. 3.



Zadanie to można wykonać na dwa sposoby:

1. korzystając z instrukcji **for**,
2. korzystając z operacji i funkcji macierzowych lub wektorowych.

Kod programu wg sposobu 1.

```
clc;
clear;
disp('Obliczanie charakterystyk geometrycznych przekroju');
%wczytywanie danych wektora b i h
b=input('podaj szerokosci [b1 b2 ... bn]:');
h=input('podaj wysokosci [h1 h2 ... hn]:');
n=length(b); %liczba elementów n
AC=0;
I0=0;
for i=1:n,
    A(i)=b(i)*h(i);
    AC=AC+A(i);
    I0=I0+b(i)*h(i)^3/12;
end
```

```
y(1)=h(1)/2;
SC=y(1)*A(1);
for i=2:n,
    y(i)=y(i-1)+(h(i-1))/2;
    S(i)=y(i)*A(i);
    SC=SC+S(i);
end
yc=SC/AC;
Ist=0;
for i=1:n,
    d(i)=y(i)-yc;
    Ist=Ist+A(i)*d(i)^2;
end
I=I0+Ist;
%wyslwietlanie wynikow
disp('rozwiązanie: A S yc I')
disp([AC SC yc I]);
```

Kod programu wg sposobu 2.

```
clc;
clear;
%wczytywanie danych wektorow b i h
disp('Obliczanie charakterystyk geometrycznych przekroju');
b=input('Szerokości [b1 b2 ... bn] : ');
h=input('Wysokości [h1 h2 ... hn] : ');
n=length(b);
y=h/2;
A=b.*h; %mnozenie tablicowe
for i=2:n,
   y(i)=y(i)+sum(h(1:i-1));
end
S=y.*A;
yc=sum(S)/sum(A);
I0=b.*h.^3/12;
d=y-yc*ones(size(y));
Ist=A.*d.^2;
I=sum(I0+Ist);
%wyświetlanie wyników
disp('rozwiazanie: A S yc I');
disp([A S yc I]);
```

UWAGA!! W obydwu powyższych programach dane wejściowe wprowadzamy tak, jak wektor wierszowy tj.: *Szerokości [b1 b2 ... bn] : [2 5 9 4]*

Zad. 2. Obliczyć wartości sił tnących i momentów zginających i wykreślić ich przebiegi dla belki wolnopodpartej obciążonej siłą skupioną.

Zadanie rozpoczynamy od utworzenia pliku o nazwie *belka.m*

>> save belka.m

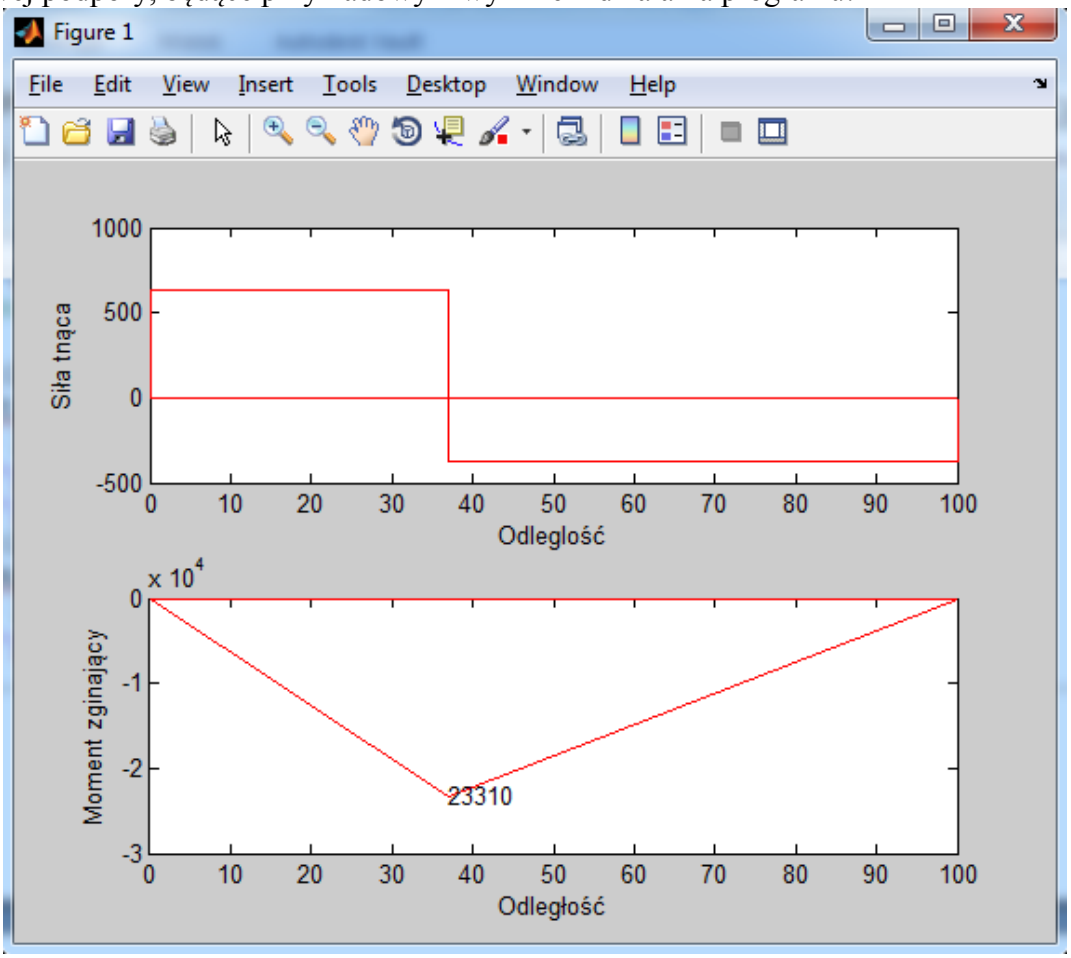
>> edit belka.m

W oknie edytora piszemy program jak poniżej.

```
%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('Odległość')
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
```

Wynik działania programu:

Poniżej przedstawiono wykresy sił tnących i momentów zginających dla belki wolnopodpartej o długości l=100 obciążonej siłą P=1000 przyłożoną w odległości x=37 od lewej podpory, będące przykładowym wynikiem działania programu:



Uwaga!! Wprowadzając dane należy pamiętać, że dla tego skryptu są to odpowiednio – *l i x* podane w [mm], natomiast *P* w [N].

Zad. 3. Wyznaczyć charakterystyki geometryczne i narysować rdzeń przekroju teowego/
Zadanie zaczynamy od utworzenia pliku *teownik.m*.

>>save teownik.m

>> edit teownik.m

W oknie edytora piszemy program jak poniżej.

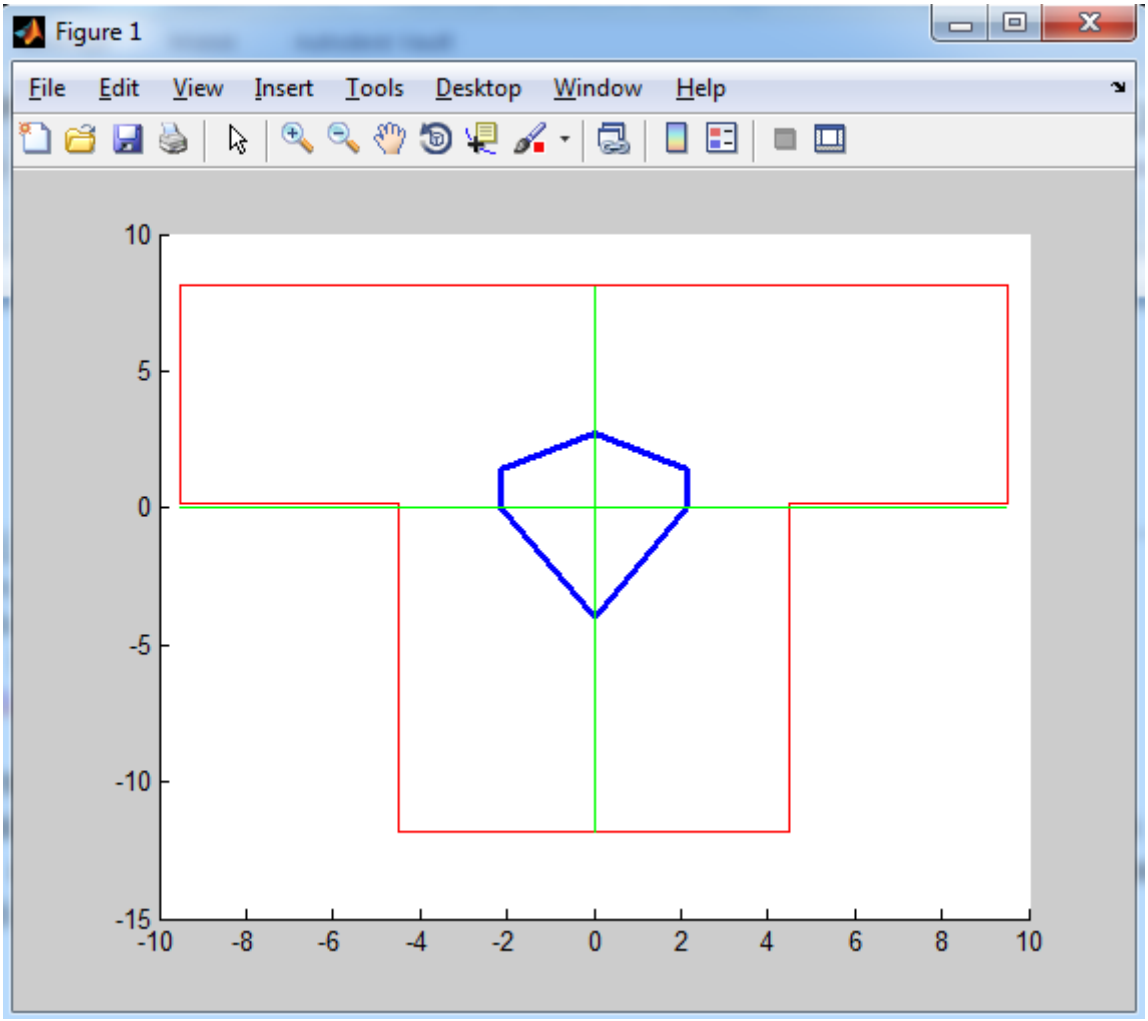
```
%Program oblicza charakterystyki geometryczne i rysuje rdzeń przekroju
teowego
%Dane do programu:
% h - wysokość przekroju
% b - szerokość półki
% t - grubość środnika
% d - grubość półki
clear
clc
disp('Program rysuje rdzeń przekroju teowego')
disp(' ')
%wprowadzanie danych
h=input('Podaj całkowitą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

Wynik działania programu:

Poniżej przedstawiono charakterystyki geometryczne i rysunek rdzenia przekroju teowego o całkowitej wysokości h=20 szerokości półki b=19, grubości środnika t=9 i grubości półki d=8, będące przykładowym wynikiem działania programu:

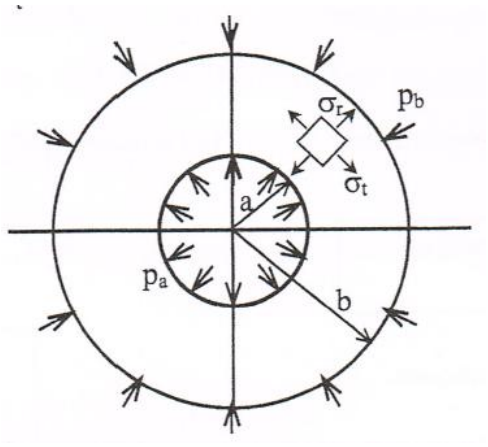


Wyniki obliczeń numerycznych zostaną zapisane w pliku wyniki.mat

Zad. 4. Rozwiązać zagadnienie Lame'go. Rura grubościenna o promieniu wewnętrznym a i zewnętrznym b obciążona jest ciśnieniem wewnętrznym p_a i zewnętrznym p_b (rys. 4). Obliczyć naprężenia promieniowe i obwodowe w funkcji promienia wg wzorów:

$$\sigma_r = \frac{a^2 p_a - b^2 p_b}{b^2 - a^2} - \frac{a^2 b^2 (p_a - p_b)}{(b^2 - a^2) r^2}$$

$$\sigma_t = \frac{a^2 p_a - b^2 p_b}{b^2 - a^2} + \frac{a^2 b^2 (p_a - p_b)}{(b^2 - a^2) r^2}.$$



Zagadnienie Lame'go

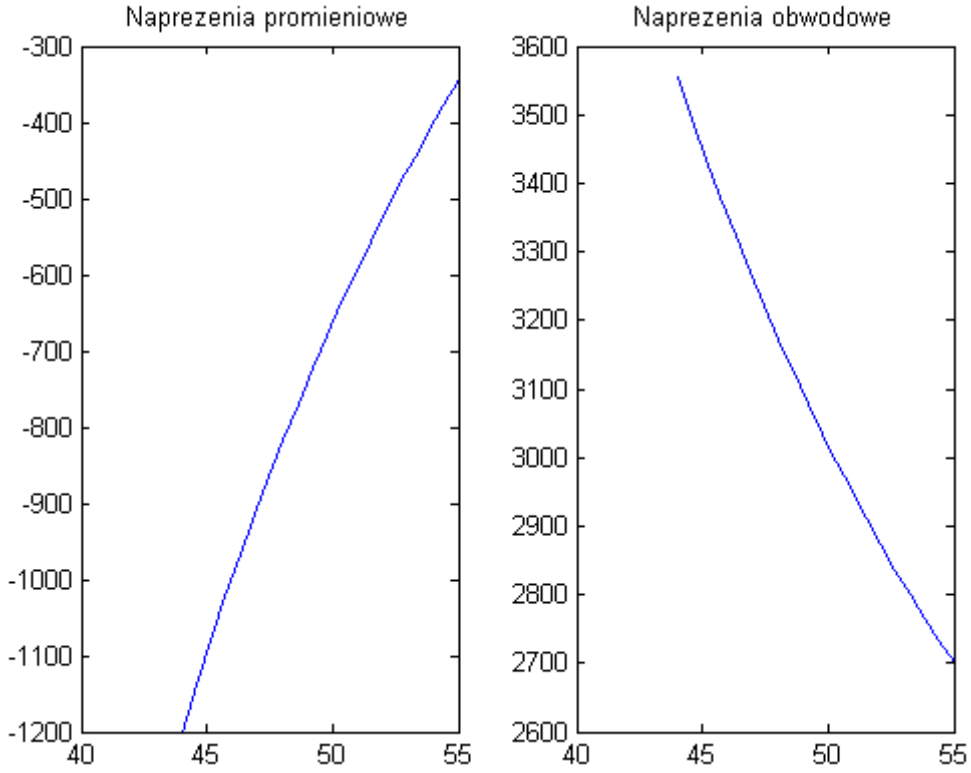
Tworzymy plik o nazwie lame.m. W edytorze kodu należ napisać poniższy kod.

```
clc;
clear;
%wczytywanie danych
a=input('Podaj promień wewnętrzny a= ');
b=input('Podaj promień zewnętrzny b= ');
pa=input('Podaj ciśnienie wewnętrzne pa= ');
pb=input('Podaj ciśnienie zewnętrzne pb= ');
if a>=b
   disp('Blad!! powinno byc a<b!!')
else
   r=linspace(a,b,20);
   sr=(a^2*pa-b^2*pb)/(b^2-a^2)-a^2*b^2*(pa-pb)/(b^2-a^2)./r.^2;
   st=(a^2*pa-b^2*pb)/(b^2-a^2)+a^2*b^2*(pa-pb)/(b^2-a^2)./r.^2;
   disp('Naprezenia promieniowe : r, sr, st ');
   disp([r', sr', st']);
   subplot(1,2,1);
   plot(r,sr);
   title('Naprezenia promieniowe');
   subplot(1,2,2);
   plot(r,st);
   title('Naprezenia obwodowe');
end
```

Wynik działania kodu dla przykładowych danych

```
Podaj promień wewnętrzny a= 44
Podaj promień zewnętrzny b= 55
Podaj ciśnienie wewnętrzne pa= 1200
Podaj ciśnienie zewnętrzne pb= 344
Naprezenia promieniowe : r, sr, st
  1.0e+003 *

    0.0440   -1.2000    3.5556
    0.0446   -1.1386    3.4942
    0.0452   -1.0796    3.4352
```

0.0457	-1.0228	3.3784
0.0463	-0.9682	3.3237
0.0469	-0.9155	3.2711
0.0475	-0.8648	3.2203
0.0481	-0.8158	3.1714
0.0486	-0.7687	3.1242
0.0492	-0.7231	3.0787
0.0498	-0.6792	3.0347
0.0504	-0.6367	2.9923
0.0509	-0.5957	2.9513
0.0515	-0.5561	2.9117
0.0521	-0.5178	2.8733
0.0527	-0.4807	2.8363
0.0533	-0.4449	2.8004
0.0538	-0.4102	2.7657
0.0544	-0.3766	2.7321
0.0550	-0.3440	2.6996



Wynik działania zagadnienia Lame'go