CYFROWE MODELOWANIE ZAGADNIEŃ TECHNICZNYCH

Workbook

1. Wprowadzenie do systemu MATLAB

Szereg zagadnień technicznych jest dość trudnych do wyznaczenia bez wsparcia numerycznego. W praktyce naukowej i inżynierskiej wykorzystuje się różne systemy numeryczne zadaniem, których jest wykonanie określonych obliczeń i symulacji. Aby dokonać tych działań należy stworzyć model matematyczny zarówno badanego obiektu jak i jego zachowań, aby móc przeprowadzić symulację numeryczną.

Większość oprogramowania bazuje na gotowych rozwiązaniach, które nie zawsze w pełni odpowiadają na nasze potrzeby i oczekiwania. Nie oznacza to jednak, że należy umniejszać przydatność takich produktów. Większą swobodę w tworzeniu modeli i wykonywaniu symulacji numerycznych dają programy, które umożliwiają skonstruowanie i napisanie własnego modelu analizowanego obiektu. Przykładem takiego rozwiązania jest MATLAB - pakiet obliczeniowy firmy MathWorks przeznaczony do wykonywania różnorodnych obliczeń numerycznych.

Sercem pakietu jest 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 MATLAB - **MAT**rix **LAB**oratory. Pakiet posiada obszerne biblioteki dodatkowych procedur umożliwiające rozwiązywanie typowych problemów obliczeniowych. Pakiet charakteryzuje się prostą budową okienkową, która ułatwia korzystanie z programu a łatwa i estetyczna wizualizacja wyników w postaci dwu- i trójwymiarowych wykresów dopełnia jego funkcjonalność. Dodatkową zaletą pakietu MATLAB jest możliwość przeprowadzenia obliczeń symbolicznych (na wzorach).

Podstawowe zasady pracy w systemie MATLAB

Duże i małe litery:

MATLAB rozróżnia duże i małe litery. Oznacza to, że np. A i a nie są tymi samymi zmiennymi. Wszystkie nazwy poleceń wprowadzanych po wyświetleniu na ekranie znaku „zaproszenia" >> muszą być pisane małymi literami. Instrukcją *casesen* można spowodować niewrażliwość na wielkość liter. Ponowne jej wywołanie przywraca stan początkowy.

Uzyskiwanie informacji pomocniczej systemu:

Instrukcja help <temat> powoduje wyświetlenie komentarza dotyczącego danego tematu. Pod nazwą <temat> wpisuje się na ogół nazwę procedury. Gdy napiszemy tylko help, nastąpi wyświetlenie tematów objętych tą instrukcją. Innymi ważnymi własnościami pracy w systemie MATLAB są:

- Średnik ; użyty po wyrażeniu lub instrukcji blokuje wyświetlanie wyniku na ekranie,
- Dwie (lub więcej) kropki .. na końcu linii oznaczają kontynuację,
- Przy edycji poleceń istnieją następujące możliwości:
  - % - oznacza, że dalsza część linii jest komentarzem,
  - ↑ - przywołanie poprzedniej linii
  - ↓ - przywołanie następnej linii
  - ← - przesunięcie kursora w lewo o jeden znak

- → - przesunięcie kursora w prawo o jeden znak
- Ctrl← - przesunięcie kursora o jedno słowo w lewo
- Ctlr→ - przesunięcie kursora o jedno słowo w prawo
- Home - przesunięcie kursora do początku linii
- End - przesunięcie kursora na koniec linii
- Esc - kasowanie całej linii
- Ins - przełączanie między wstawianiem, a wpisywaniem w miejsce starego
- Del - kasowanie znaku przed kursorem
- Backspace -kasowanie znaku na lewo od kursora.

Tryby użytkowania Matlaba

Można rozróżnić cztery tryby wykorzystywania Matlaba:

1) **Tryb bezpośredni**, czyli **wpisywanie wyrażeń i poleceń w oknie komend** IDE Matlaba. W trybie tym pojedyncza sesja polega na wpisaniu w oknie komend linii zawierającej jedną lub kilka komend (poleceń, instrukcji) oddzielonych przecinkami lub średnikami. Zakończenie tej linii klawiszem ENTER spowoduje natychmiastowe wykonanie zawartych w niej komend lub wyświetlenie komunikatu o błędzie. Tryb bezpośredni jest używany raczej jako pomocniczy - głównie przy uczeniu sie i testowaniu działania poszczególnych funkcji lub wartości zmiennych.

2) **Tryb pośredni, programowy** - to tworzenie programów **za pośrednictwem edytora**. Edytor Matlaba można uruchomić z menu File-New lub przez klikniecie ikony "kartki". Po wpisaniu programu w edytorze należy zapisać go do pliku. Nazwy plików podlegają takim samym regułom jak nazwy zmiennych (pierwszym znakiem musi być litera). Standardowo plik otrzyma rozszerzenie nazwy ".m" i będzie zapisany w folderze ustawionym jako bieżący. Program uruchamiamy albo z edytora (RUN) albo z okna komend przez wpisanie nazwy pliku bez rozszerzenia nazwy (.m). Programy można też pisać w innym prostym edytorze tekstowym jak "Notatnik" w MS Windows).

3) **Tryb graficzny**. Matlab posiada również środki do opracowywania programów posiadających **graficzny interfejs użytkownika** i wykorzystujących elementy okien dialogowych znane z MS Windows. Programy wprawdzie nadal są częściowo tworzone w edytorze jednak częściowo mogą być generowane w sposób półautomatyczny z wykorzystaniem narzędzia typu ***RAD*** (*Rapid Application Development*) nazywającego się ***GUIDE***. Polega to na wstawianiu elementów dialogu (suwaków, pól edycyjnych itp.) na formatki graficzne (*figure*) i modyfikowaniu ich cech.

4) **Tryb symulacyjny**. Rozszerzeniem Matlaba jest **SIMULINK** - pozwalający zestawiać z różnorodnych bloków funkcjonalnych modele "analogowe" układów dynamicznych mające postać schematów funkcjonalnych (blokowych) a następnie zadawać przebiegi wielkości wejściowych i obserwować przebiegi wielkości wyjściowych. SIMULINK pozwala wiec badać modele w sposób zbliżony jak na maszynach analogowych. Jest on wprawdzie dodatkowym elementem Matlaba, jednak prace z Simulinkiem można uznać za tryb czwarty. W Simulinku istnieje biblioteka modułów mechanicznych **SimMechanics** pozwalającą budować funkcjonujące modele układów mechanicznych. Dokumentacja SimMechanics jest na stronie: *http://www.mathworks.com/help/toolbox/physmod/mech/* natomiast filmy demonstrujące używanie są na: *http://www.mathworks.com/products/simmechanics/demos.html*

Podstawowe elementy języka Matlab

Nieżalenie od trybu w jakim używamy Matlaba – musimy znać jego język programowania – noszący tą sama nazwę co cały pakiet. MATLAB jest językiem programowania wysokiego poziomu, posiadającym m.in. różnorodne struktury danych, instrukcje sterujące wykonywaniem programów oraz wprowadzaniem i wyprowadzaniem danych, bogaty zbiór funkcji a także możliwości programowania obiektowego.

Tak jak w każdym języku programowania – w Matlabie treść programu tworzą głównie **instrukcje** zwane też **komendami** lub **poleceniami** a także **deklaracje** - zazwyczaj pomijane w prostszych programach. Podstawowymi składnikami instrukcji sa:
• **słowa kluczowe** np.: for, end, if, else, while,
• **stałe i zmienne** (w tym także złożone struktury danych),
• **wyrażenia** zawierające m.in. wywołania **funkcji.**

Wyrażenia skalarne i ich składniki

Wyrażenia - podobnie jak w innych językach - mogą zawierać:
• stałe (liczby)
• zmienne (nazwy zmiennych)
• operatory działań
• nawiasy
• funkcje

Jednak inaczej niż w innych językach - wyrażenia te dotyczą tablic (macierzy), które w szczególności mogą być skalarami (o wartościach będących pojedynczymi liczbami).

Skalarne stałe liczbowe - postacie zapisu liczb

Podobnie jak w większości języków programowania **zapis liczb** w MATLABie może zawierać:

• początkowy znak plus (na ogół pomijany) lub minus

• **kropkę dziesiętną (NIE PRZECINEK!)** poprzedzającą cześć ułamkowa np.: -97.6397

• może być stosowana tzw. **notacja naukowa** w której **e oznacza "dziesięć do potęgi ..."** np.: **-1.60210e-23** oznacza: -1.60210 razy 10 do protegi -23 ($-1.60210*10^{-23}$)

• w zapisie liczb urojonych i zespolonych stosuje sie symbol jednostki urojonej "i" zawsze poprzedzony liczbą np.: **1i; 5.7- 3.149j; 2+3e5i.**

Należy pamiętać, że w MATLAB-ie nie powinno stosować się znaków narodowych, które wymagają dodatkowego kodowania, powoduje to komplikacje w działaniu interpretera.

Działania na liczbach

Najprostszym zastosowaniem MATLAB jest funkcja kalkulatora, nie wymagająca żadnej umiejętności programowania. Po wprowadzeniu wartości stałych (np. 3+5) i wprowadzeniu polecenia ENTER na ekranie pojawi się symbol odpowiedzi

```
>>3+5
ans=
        8
```

Operatory działań arytmetycznych na skalarach i ogólnie na macierzach są następujące:

Tab. 1. Operatory działań arytmetycznych

Operator	Objaśnienie
+	dodawanie
-	odejmowanie lub zmiana znaku
*	mnożenie
/	dzielenie
^	potęgowanie

Oprócz działań arytmetycznych istotnym elementem języka MATLAB są wyrażenia logiczne (warunki). Wyrażenia te buduje się przy pomocy **operatorów relacji** i **operatorów logicznych** (tab. 2 i 3). Operatory relacji określone są w zbiorze uporządkowanym np. w

zbiorze liczb. Operatory logiczne pozwalają tworzyć złożone wyrażenia logiczne. Najwyższy priorytet ma operator negacji (not), następnie pozostałe operatory logiczne (and, or). Niższy priorytet mają operatory arytmetyczne a najniższy mają operatory relacji. Z tego też powodu w wyrażeniach warunkowych należy używać nawiasów.

Tab. 2. Operatory relacji

Operator	Objaśnienie
<	mniejsze
<=	mniejsze lub równe
>	większe
>=	większe lub równe
= =	równe
~=	nierówne

Tab. 3. Operatory logiczne

Nazwa	Symbol	
negacja	not	~
koniunkcja	and	&
alternatywa	or	|

Macierze

Podstawowym typem organizacji danych w MATLAB jest macierz dwuwymiarowa. Szczególnymi odmianami macierzy są:

- skalar – macierz o wymiarach 1x1,
- wektor wierszowy – macierz o jednym wierszu,
- wektor kolumnowy – macierz o jednej kolumnie.

Macierze w MATLAB definiuje się na kilka sposobów. Pierwszym z nich jest wyliczenie elementów. Wiersze macierzy oddziela się średnikami, a poszczególne elementy spacjami.

Przykład:

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

Definicja macierzy przez wygenerowanie elementów:

```
A=[min:krok:max]
```

Polecenie generuje wektor poczynając od wartości *min* do elementu o wartości *max* z krokiem *krok*. Gdy parametr *krok* jest pominięty, to przyjmuje on wartość domyślną *krok=1*.

Przykład:

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

Definiowanie macierzy przez wykorzystanie elementów innych macierzy.

Przykład: utworzyć macierz D budując ją z 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.

Funkcje wspomagające konstruowanie macierzy

1. definicja macierzy jednostkowej o wymiarach *nxn* lub *mxn:*

```
A=eye(n)
A=eye(m,n)
A=eye([m n])
```

2. definicja macierzy o wymiarach *nxn* lub *mxn* wypełnionej jedynkami:

```
A=ones(n)
A=ones(m,n)
A=ones([m n])
```

3. definicja macierzy o wymiarach *nxn* lub *mxn* wypełnionej zerami:

```
A=zeros(n)
A=zeros(m,n)
A=zeros([m n])
```

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

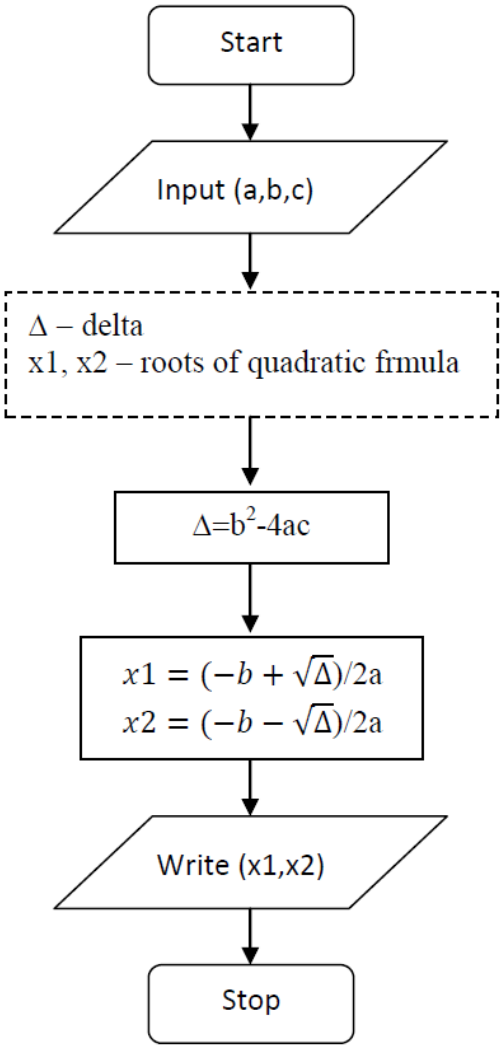
Algorytmy

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.

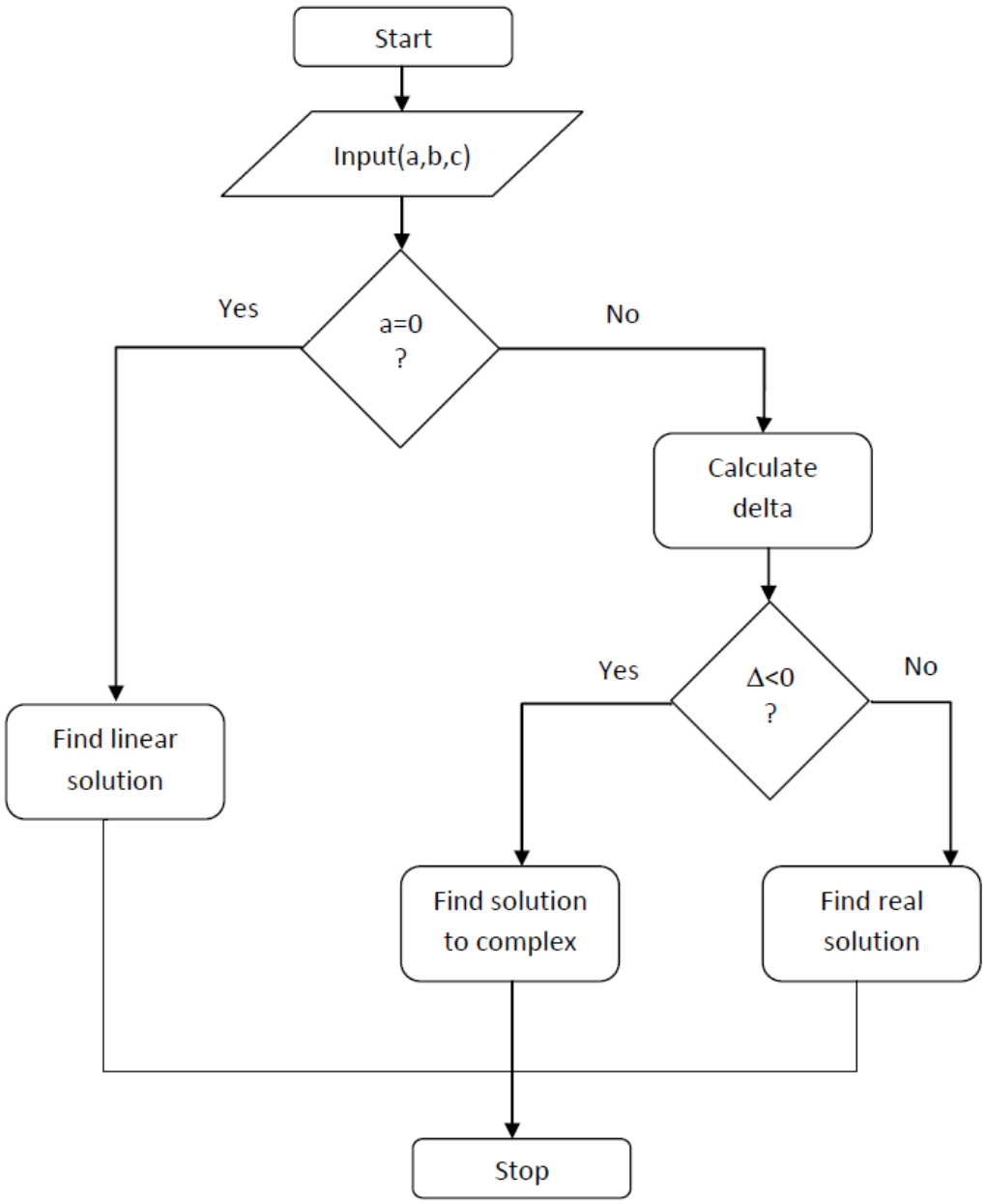


Rys. 1. Algorytm wyznaczania równania kwadratowego (ogólny)

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)



Rys.2. 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:

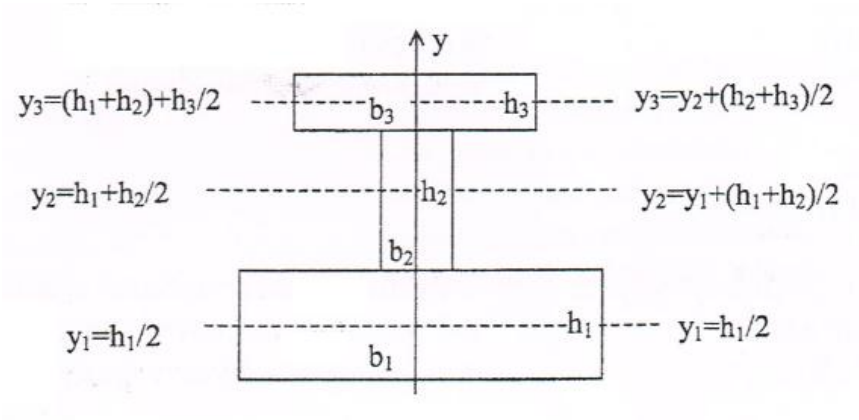
a) pole figury $A = \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} b_i\, h_i$

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 1.

```
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

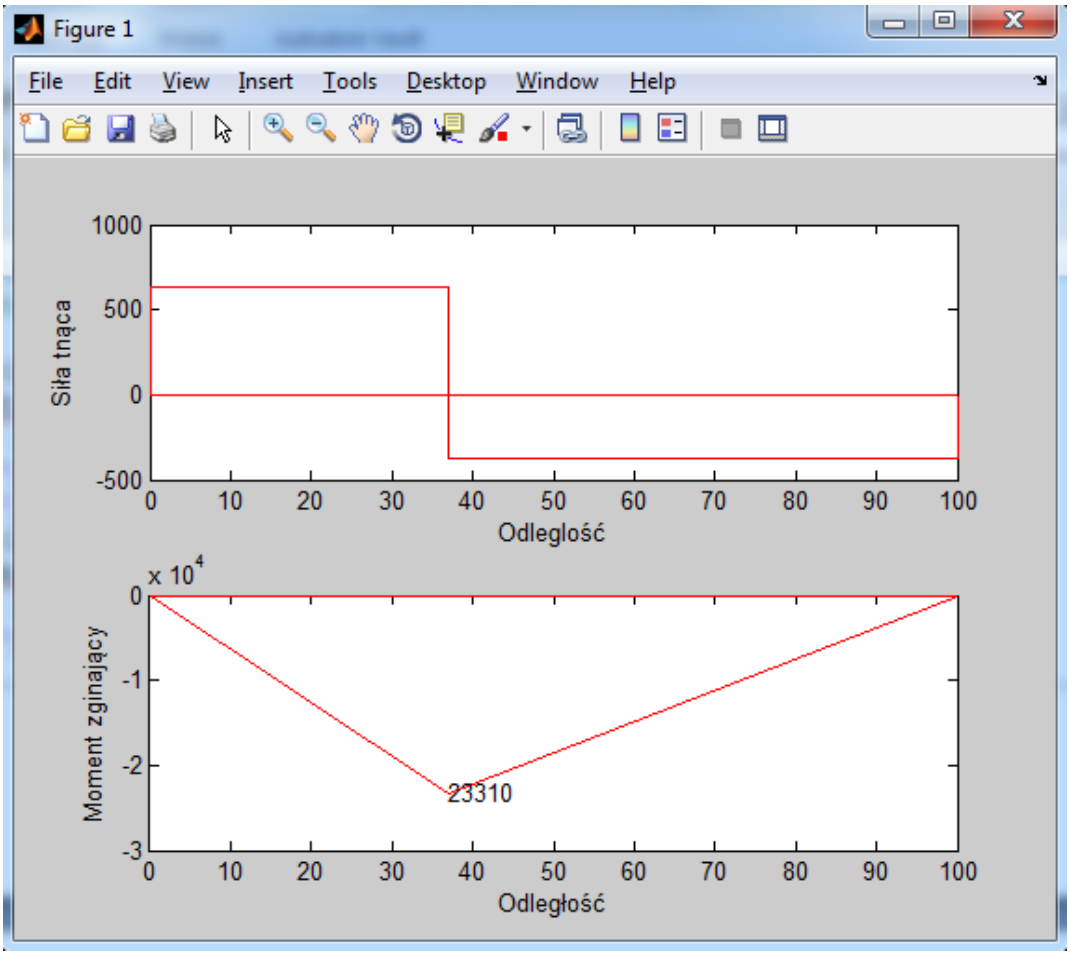
W oknie edytora piszemy program jak poniżej.

```
%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*.

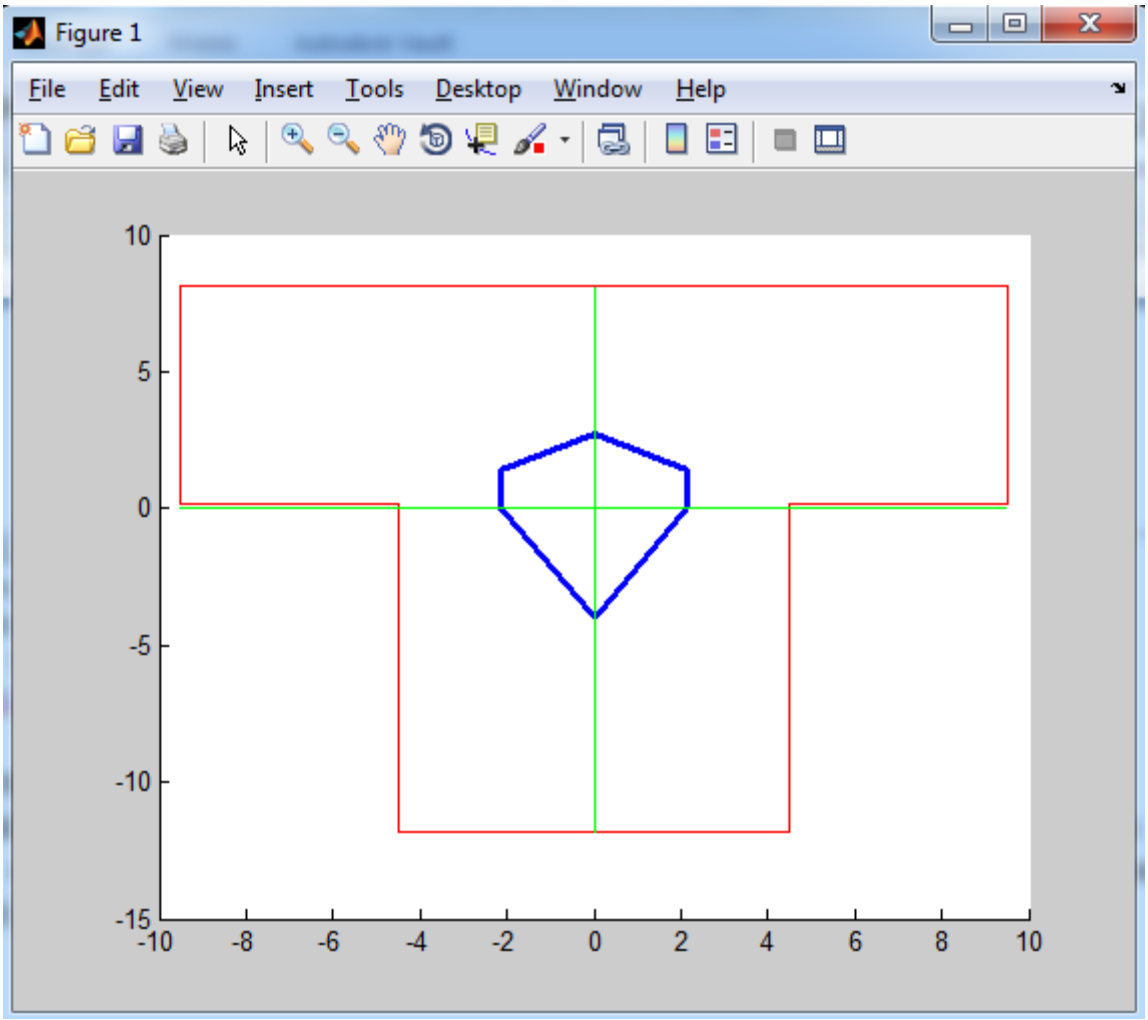
>>save teownik.m
>> edit teownik.m
W oknie edytora piszemy program jak poniżej.

```
%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:

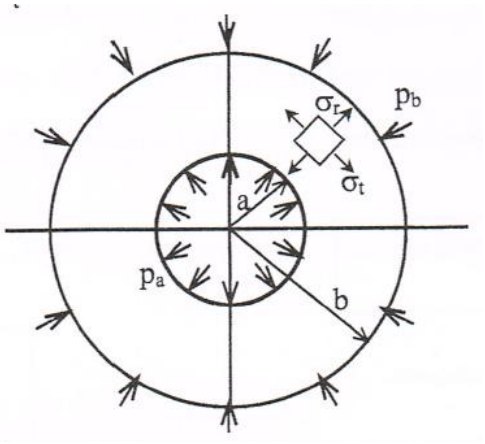


Wyniki obliczeń numerycznych zostaną zapisane w pliku wyniki.mat

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:

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

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



Rys. 4. Zagadnienie Lame'go

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

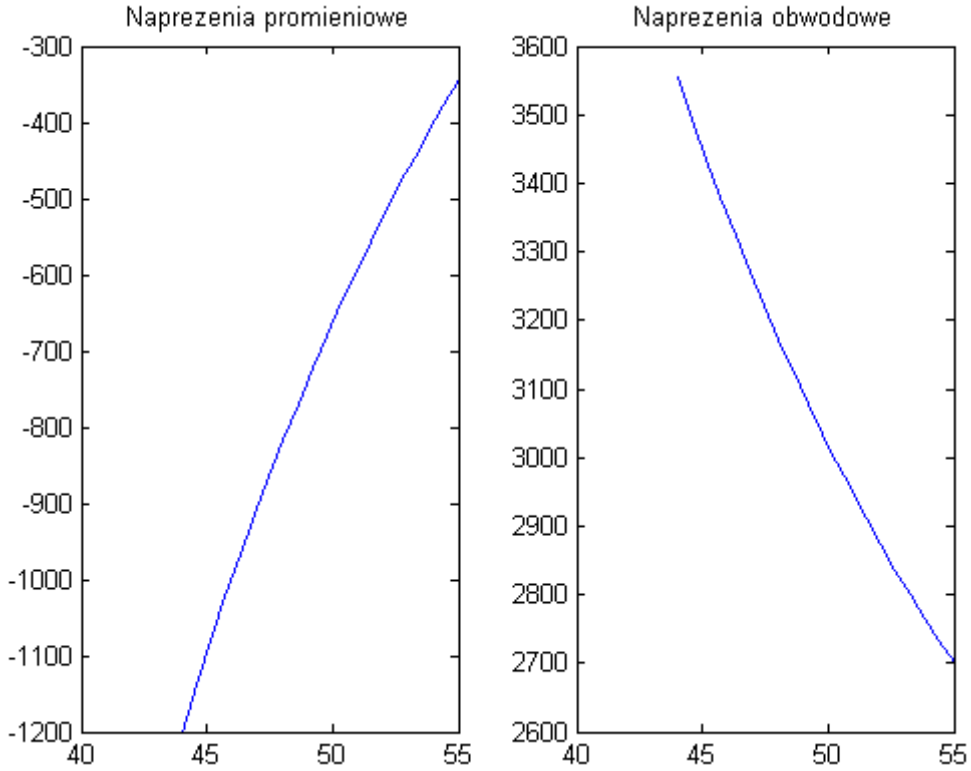
```
clc;
clear;
%wczytywanie danych
a=input('Podaj promień wewnętrzny a= ');
b=input('Podaj promień zewnętrzny b= ');
pa=input('Podaj ciśnienie wewnętrzne pa= ');
pb=input('Podaj ciśnienie zewnętrzne pb= ');
if a>=b
   disp('Blad!! powinno byc a<b!!')
else
   r=linspace(a,b,20);
   sr=(a^2*pa-b^2*pb)/(b^2-a^2)-a^2*b^2*(pa-pb)/(b^2-a^2)./r.^2;
   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



Rys. 5. Wynik działania zagadnienia Lame'go

Biblioteka SYMBOLIC

Podstawowe funkcje symboliczne	
Nazwa	Symbol
Definicja zmiennej symbolicznej postaci ‘a’	sym(‘a’)
Definicja zmiennych symbolicznych o nazwach a, b, c i postaciach ‘a’ ‘b’ ‘c’	syms a b c
Uproszczenie postaci wyrażenia zmiennej symbolicznej a	sipmplify(a)
Zamiana zmiennej symbolicznej a na tekst	char(a)
Podstawienie wartości w za symbol ‘t’ w zmiennej symbolicznej f	subs(f, ‘t’,w)
Obliczenie wyrażenia funkcji odwrotnej do x(t) czyli t(x)	finverse(x,t)
Obliczenie wyrażenia symbolicznego funkcji złożonej f(g(x))	compose(f,g)
Rozwiązywanie równań algebraicznych	solve
Rozwiązywanie równań różniczkowych	dsolve
Obliczanie wyrażenia symbolicznego pochodnej funkcji f	diff(f)
Obliczanie wyrażenia symbolicznego funkcji pierwotnej f	int(f)
Rysowanie wykresu 2D funkcji wyrażenia symbolicznego f w przedziale [a b]	Ezplot(f, [a b])