

Crearea și controlul figurilor și obiectelor grafice (Instrumente GUI – Graphics User Interface Tools) Aplicații GUI create cu *App Designer*

App Designer ne permite să creăm aplicații grafice profesionale în MATLAB.

Fișierul cu aplicația GUI are extensia **.mlapp**

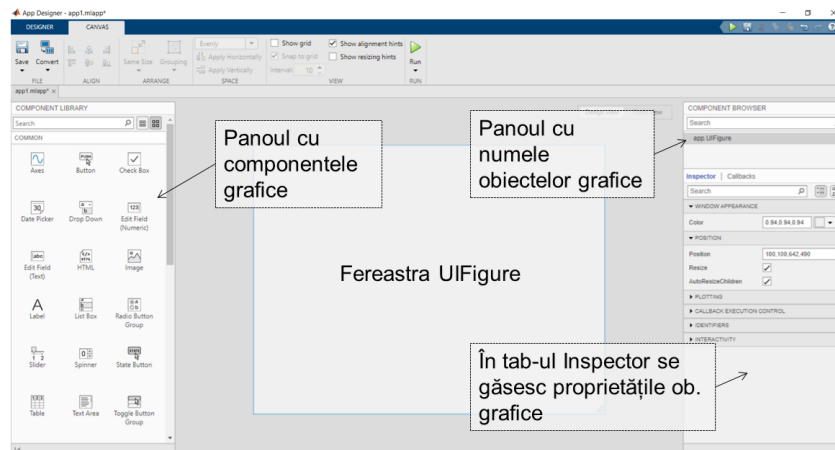
Pentru a crea interfețe grafice în Matlab putem executa: comanda **appdesigner** în fereastra de comandă, sau putem crea un fișier GUI nou de la tab-ul **Home->New->App**

În realizarea interfețelor grafice trebuie să urmăm două etape:

- Prima etapă (Design View) constă în realizarea propriu-zisă a interfeței, iar
- A doua etapă (Code View) constă în scrierea fișierului sursă (acțiuni aduse obiectelor grafice din interfață, precum și adăugarea/definirea de funcții noi).

Crearea aplicațiilor GUI în MATLAB

Proiectarea interfeței grafice cu utilizatorul se face prin utilizarea suprafeței de proiectare **Design View**.



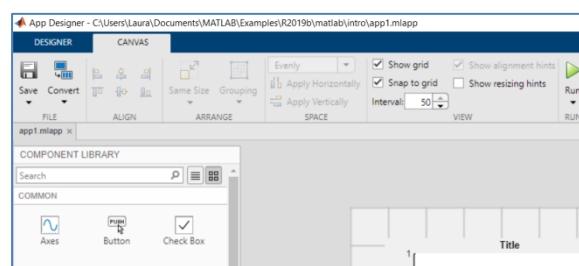
Componentele vizuale se pot adăuga prin drag-and-drop pe suprafața de proiectare și se pot utiliza opțiunile de aliniere pentru a obține un aspect precis.

App Designer generează automat codul orientat obiect care specifică aspectul și designul aplicației.

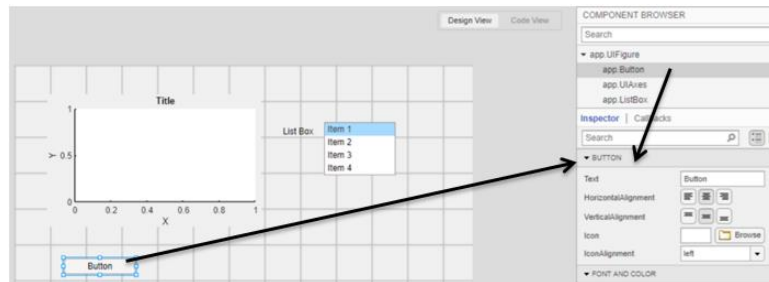
1. Aria de desenare se poate modifica astfel încât:

- să afișeze sau să ascundă grila de desenare,
- să modifice distanța între liniile grilei, etc.

Aceste opțiuni se activează de la tab-ul CANVAS din App Designer.

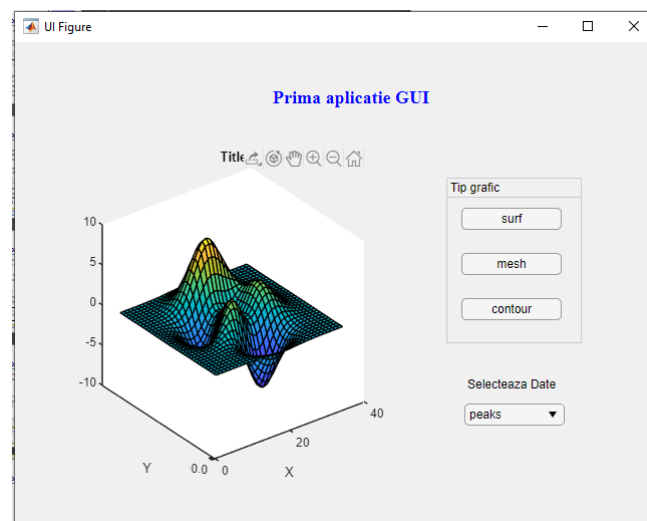


2. Pentru a adăuga un obiect grafic în fereastra din interfață, se adaugă cu mouse-ul obiectul dorit, prin drag-and-drop.
3. Parametrii unui obiect din interfață, pot fi vizualizați prin selectarea respectivului obiect grafic din interfață (Design View->Component Browser->Inspector).



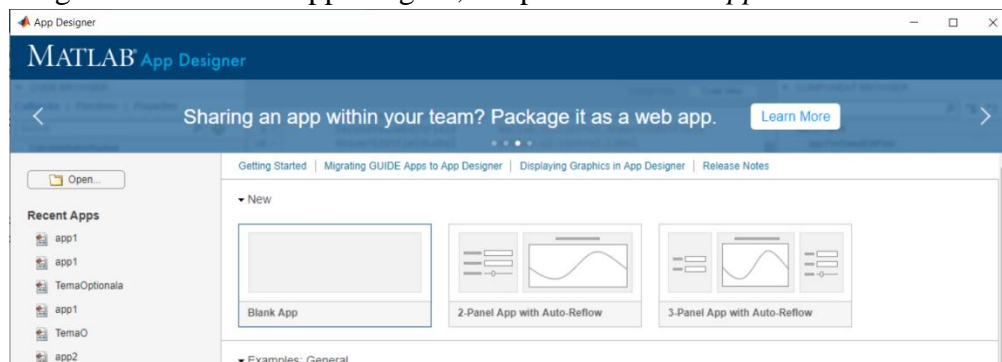
Aplicația FirstApp (v. R2019b)

Aplicația va permite afișarea unui grafic în spațiu (într-un sistem de axe) de tipul graficului selectat (*surf*, *mesh*, *contour*) la activarea butoanelor din fereastră. Într-o listă (de tip Drop Down) se vor găsi datele ce vor fi reprezentate pe axe.

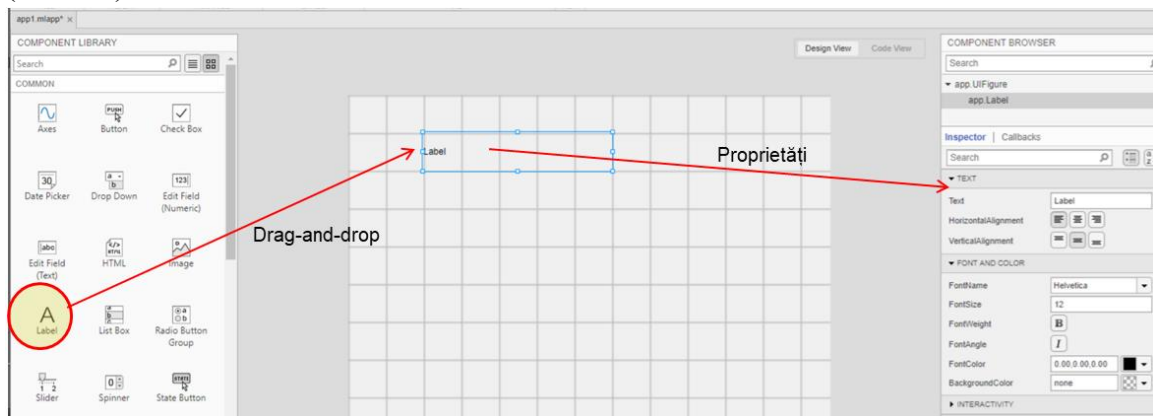


ETAPA 1) Crearea interfeței grafice în *Design View*.

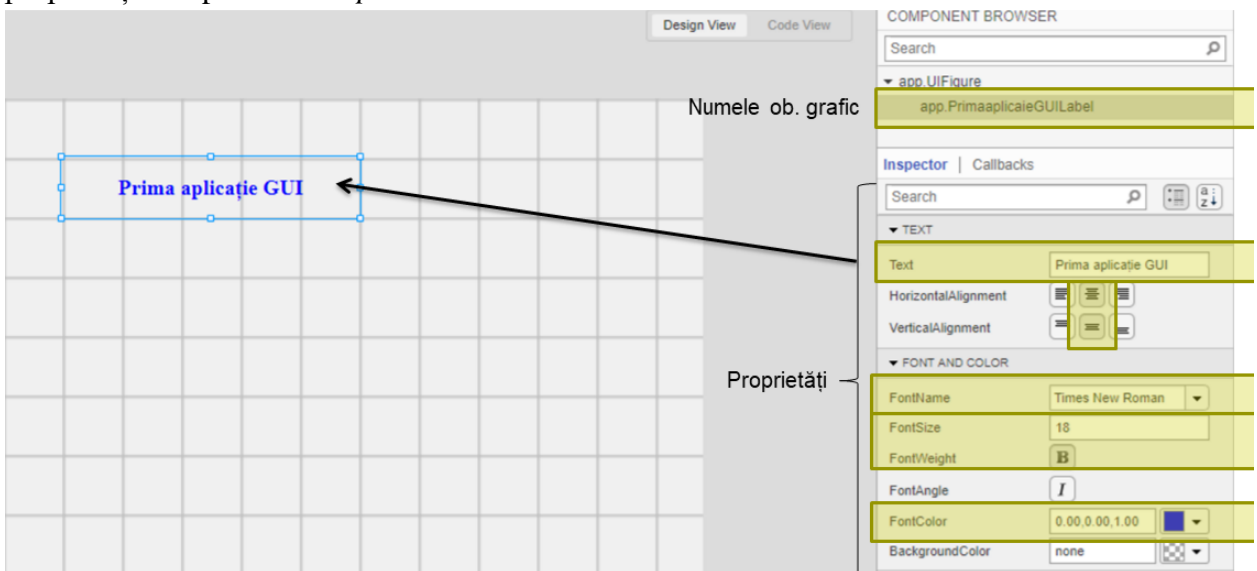
- a) Scriem la linia de comanda `appdesigner`
- b) Alegem din fereastra AppDesigner, template-ul *Blank App*



- c) Adăugăm un titlu aplicației noastre. Avem nevoie de un text static, un obiect Label (etichetă)



- d) Modificăm proprietățile componente grafice. După selectarea componente UI, modificăm proprietățile în panoul *Component Browser*.

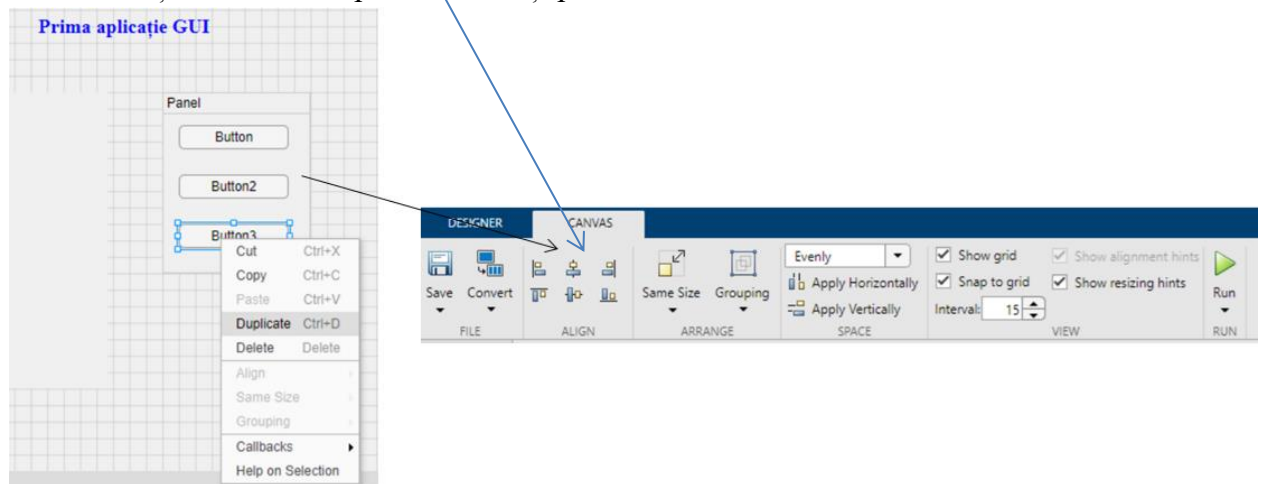


- e) Pentru aplicația pe care dorim să o construim adăugăm un obiect grafic Axes:

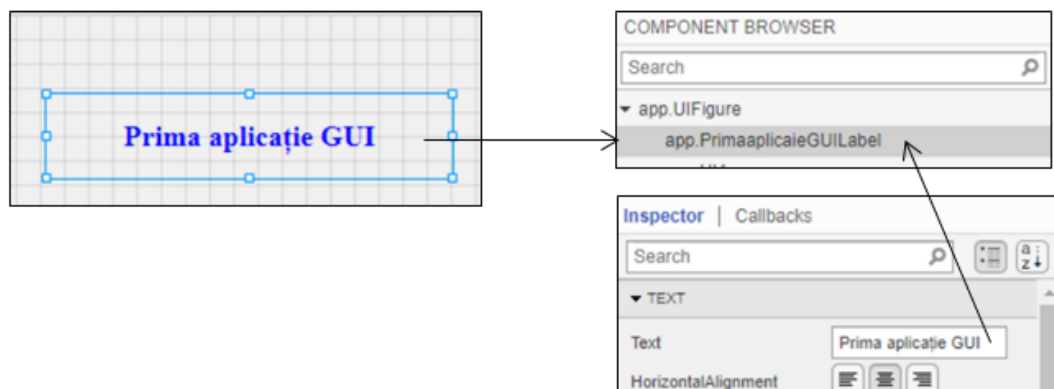


- f) Apoi adăugăm un obiect grafic Panel (secțiunea *Containers*) la care adăugăm 3 butoane – Button (secțiunea *Common*).
- g) După adăugarea unui buton, pentru multiplicare putem face copy-paste sau alegând duplicate din meniul de context.

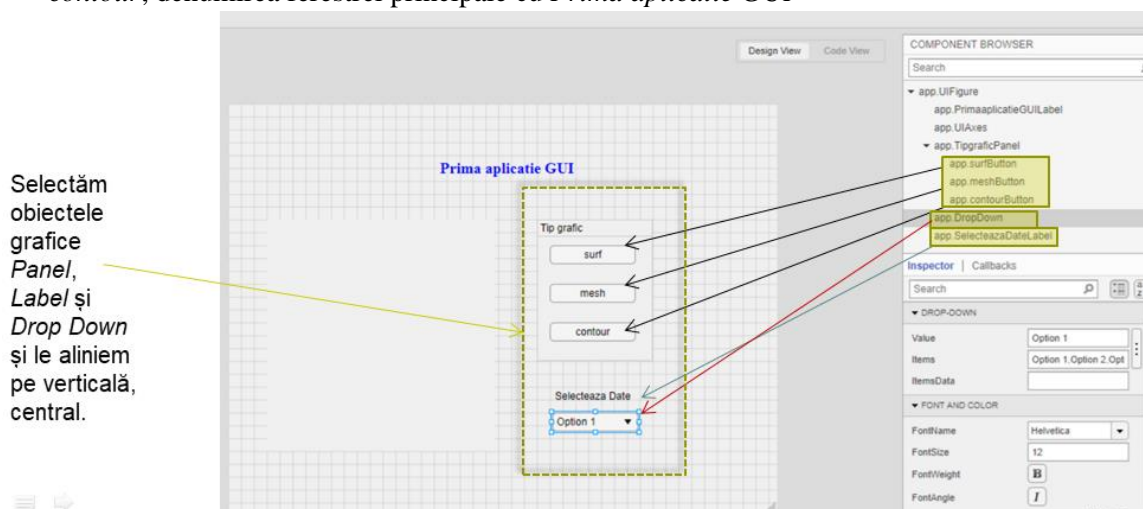
- h) Dacă selectăm butoanele, putem de la tab-ul *Canvas*, categoria *Align*, să grupăm elementele și să le aliniem pe orizontală și pe verticală.



- i) În cazul unei **obiect grafic cu proprietatea de tipul Button, Label, Edit Field, CheckBox, RadioButton**, este același **text de la proprietatea Label, Text, Title** terminat cu tipul obiectului grafic.

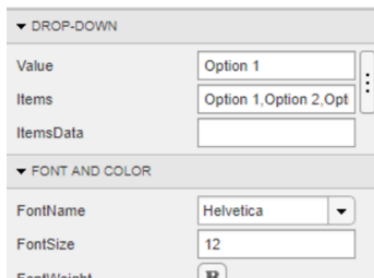
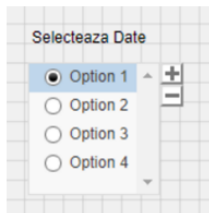


- j) Vom adăuga în continuare un obiect grafic *drop-down* și un nou text static, un *Label*.
Modificăm de la proprietăți **denumirea obiectelor**: Panel cu *tip grafic*; butoanele cu *surf, mesh, contour*; denumirea ferestrei principale cu *Prima aplicatie GUI*

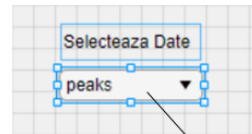


- k) Vom modifica opțiunile de la obiectul grafic *Drop Down*. Opțiunile (elementele listei) se scriu pe linii diferite.

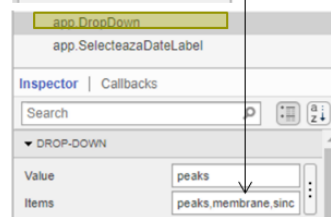
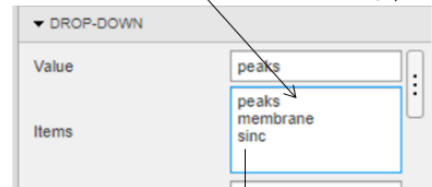
Valori inițiale



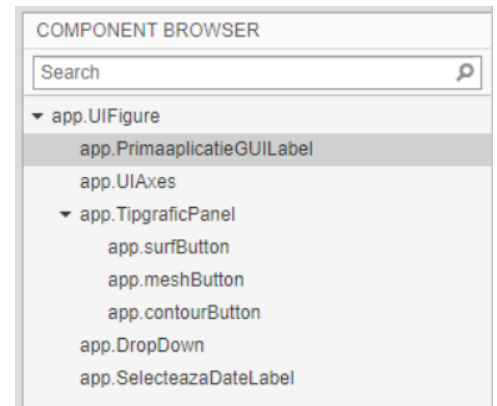
Valori modificate



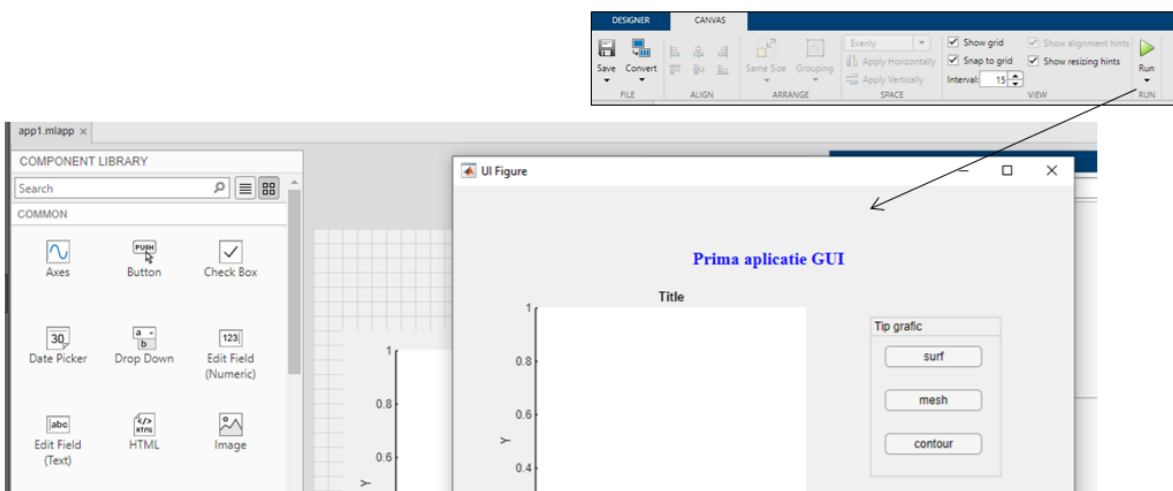
Editare opțiuni



- l) Panoul **Component Browser** conține obiectele interfeței grafice.
- m) Panoul prezintă arborele obiectelor interfeței grafice. Acest arbore are ca rădăcină un obiect de tip **UIFigure**.



- n) După terminarea construirii interfeței grafice, rulăm aplicația de la butonul **Run**
- o) Se generează și se salvează automat fișierul **.mlapp**

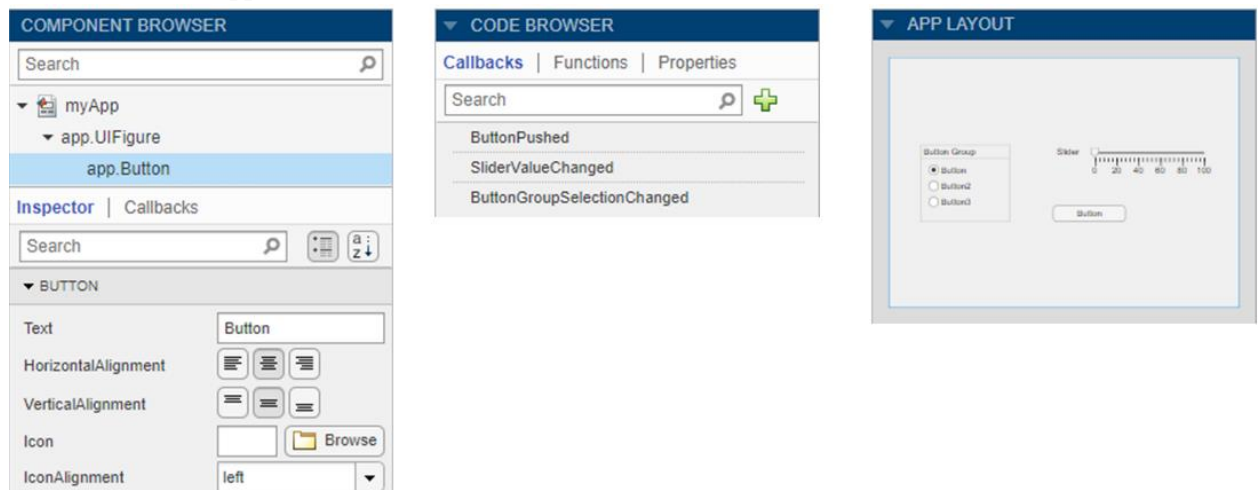


ETAPA II) Modificăm fișierul sursă din *Code View*

În *Code View* definim comportamentul aplicației.

App Designer verifică automat dacă există probleme de codare utilizând *Code Analyzer*. Putem vedea mesajele de avertizare și de eroare despre codul aplicației noastre, pe măsură ce îl scriem și putem modifica aplicația pe baza mesajelor.

Code View are trei panouri pentru a ne ajuta să gestionăm diferite aspecte ale codului: *Component Browser*, *Code Browser*, *App Layout*



a) Identificarea secțiunilor de cod modificabile.

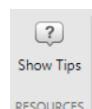
- În editor, unele secțiuni de cod sunt editabile, iar altele nu.
- **Secțiunile de cod gri nu sunt editabile.** Aceste secțiuni sunt generate și gestionate de App Designer.
- Cu toate acestea, **secțiunile albe sunt editabile** și corespund:
 - corpul funcțiilor pe care îl definim (de exemplu, funcții callback și funcții de asistență)
 - definiții de proprietăți personalizate.

```

14 % Component initialization
15
16 properties (Access = private)
17     X = 5 % Average value
18 end
19
20
21 % Callbacks that handle component events
22 methods (Access = private)
23
24     % Button pushed function: Button
25     function ButtonPushed(app, event)
26         disp('Hello World');
27     end
  
```

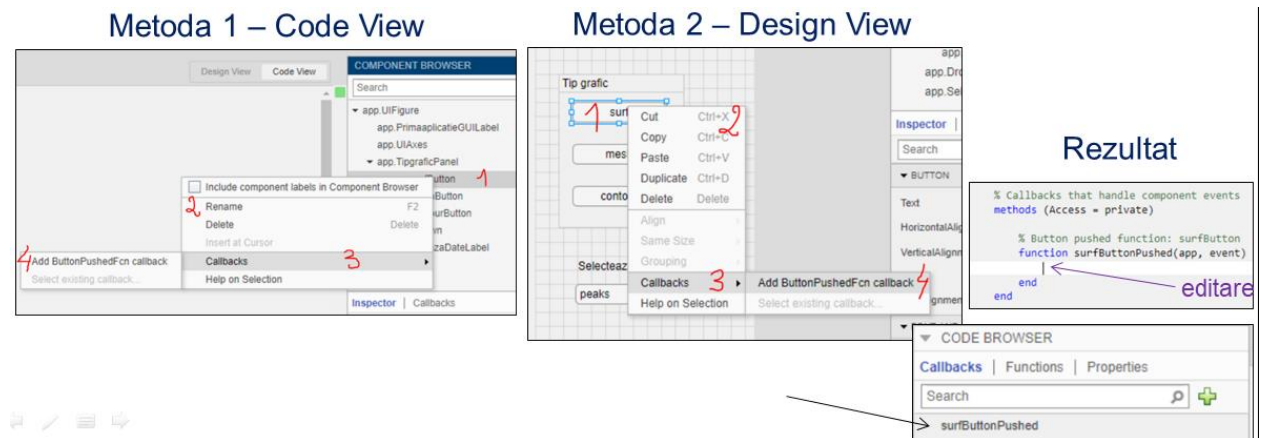
b) App Designer definește aplicația ca o clasă MATLAB.

- Nu trebuie să înțelegem clasele sau programarea orientată pe obiecte pentru a crea o aplicație grafică deoarece App Designer gestionează aceste aspecte ale codului.
- Cu toate acestea, programarea în App Designer necesită un flux de lucru diferit de cel care funcționează strict cu funcțiile.
- Putem consulta oricând un rezumat al acestui flux de lucru făcând clic pe **Show Tips** din secțiunea *Resources*, de la tab-ul **Editor**.



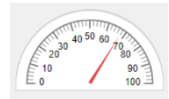
c) Crearea și gestionarea funcțiilor Callbacks.

Pentru a face o componentă să răspundă la interacțiunile utilizatorilor, se adaugă o funcție callback. Pentru aceasta, facem clic dreapta pe componentă în **Component Browser** și selectăm **Callbacks - > Add (callback property) callback**.



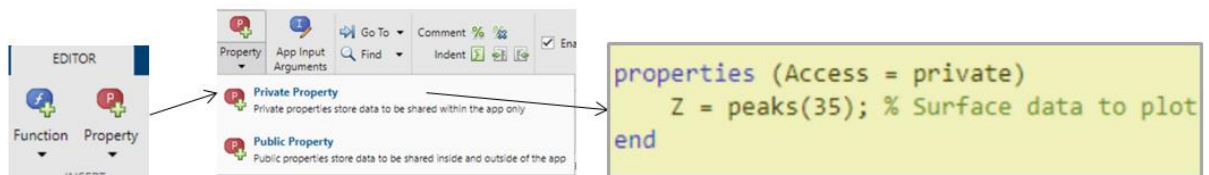
d) Utilizarea argumentelor de intrare de la funcția Callback

- Toate funcțiile callback din *App Designer* au următoarele argumente de intrare în antetul funcției:
 - **app** - obiectul aplicației. Utilizăm acest obiect pentru a accesa componentele UI din aplicație, precum și alte variabile stocate ca proprietăți.
 - **event** - un obiect care conține informații specifice despre interacțiunea utilizatorului cu componenta UI.
- Argumentul **app** prevede obiectul *app* pentru callback. Putem accesa orice componentă (și toate proprietățile specifice componentelor) în cadrul oricărui callback utilizând următoarea sintaxă:
 - `app.Component.Property`
- De exemplu, comanda `app.PressureGauge.Value = 50;` setează proprietatea *Value* a unui *gauge* (*manometru*) la 50. În acest caz, numele obiectului grafic este *PressureGauge*.



e) Scopul atributelor de proprietăți

- Specificarea atributelor în definiția clasei ne permite să personalizăm comportamentul proprietăților în scopuri specifice.
- Se pot controla caracteristicile precum accesul, stocarea datelor și vizibilitatea proprietăților prin setarea atributelor. Subclasele nu moștenesc attribute de membru superclasă.
- Modificăm codul astfel încât la deschiderea aplicației să avem setate valorile de start. Pentru început vom adăuga o proprietate pentru a crea o variabilă, pentru a stoca și pt. a partaja date între funcțiile *callbacks* și alte funcții 'utilizator'.



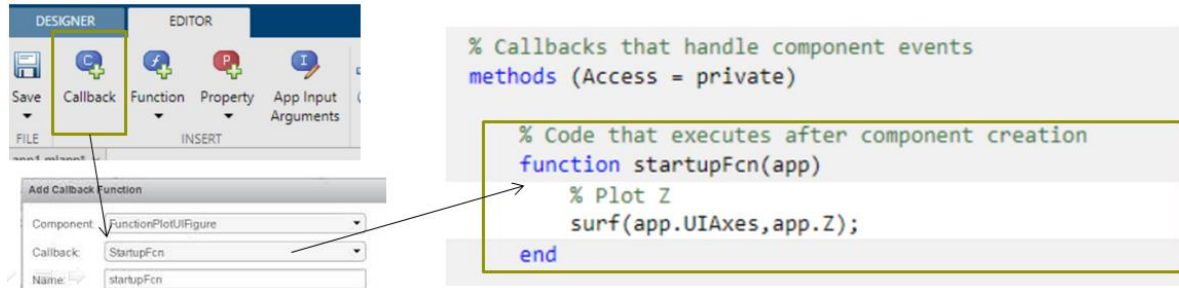
Atributul Access (numai la scriere, nu poate interoga proprietatea *meta.property*. Pentru interogări se utilizează *GetAccess* și *SetAccess*.)

- **public** – acces nerestricționat

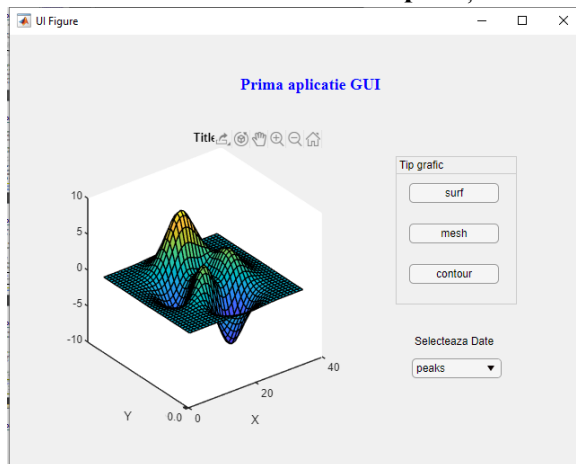
- **protected** – acces dintr-o clasă sau dintr-o subclasă
- **private** – acces doar de către membrii clasei (nu de subclase)

f) Funcția de start

Pentru a construi aplicația GUI cu valori predefinite (acțiuni ce se vor executa la deschiderea/lansarea aplicației) va trebui să creăm o funcție callback denumită *startupFcn(app)*. La deschidere se desenează suprafața cu datele din variabila Z.



g) Rezultatul la lansarea/rularea aplicației:



h) În continuare completăm funcțiile callback ale obiectelor de tip Button: *surf*, *mesh*, *contour*.

Pas 1) Creăm funcția Callback

Pas 2) Scriem codul ce va fi executat la apăsarea butoanelor.

```
% Code that executes after component creation
function startupFcn(app)
    % plot Z
    surf(app.UIAxes,app.Z);
end

% Button pushed function: surfButton
function surfButtonPushed(app, event)
    app.Z = SaveDrawData(app);
    surf(app.UIAxes,app.Z);
end

% Button pushed function: meshButton
function meshButtonPushed(app, event)
    app.Z = SaveDrawData(app);
    mesh(app.UIAxes,app.Z);
end

% Button pushed function: contourButton
function contourButtonPushed(app, event)
    app.Z = SaveDrawData(app);
    contour(app.UIAxes,app.Z);
end
```


- i) La selectarea butoanelor *surf*, *mesh*, *contour* trebuie mai întâi să se identifice elementul selectat din lista DropDown și apoi să se facă desenarea.
- j) Pentru a reduce codul executat la apăsarea butoanelor (să nu scriem la fiecare buton aceeași secvență de căutare a elementului selectat din lista DropDown), vom opta pentru construirea unei funcții ‘utilizator’ numită *SaveDrawData(app)*. Funcția va returna în argumentul de ieșire datele ce vor fi desenate în componenta UIAxes.

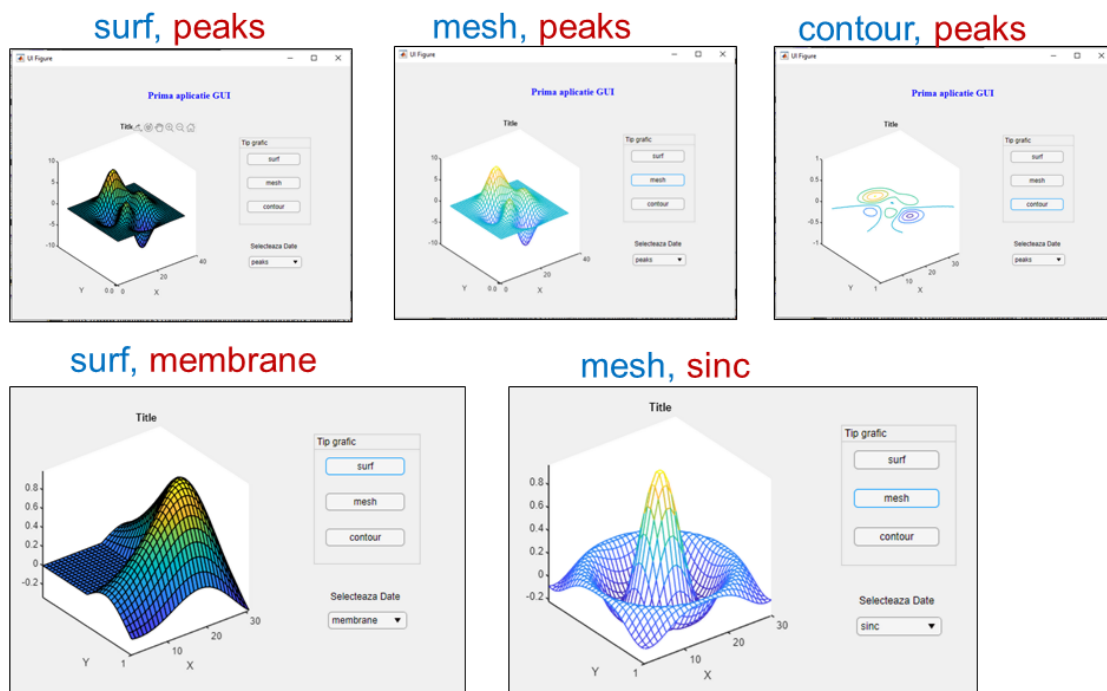


methods (Access = private)

```
function valData = SaveDrawData(app)
    value = app.DropDown.Value;
    valData=[];

    switch value
        case 'peaks'
            valData = peaks(35);
        case 'membrane'
            valData = membrane;
        case 'sinc'
            [x,y]=meshgrid(linspace(-8,8,30));
            r=sqrt(x.^2+y.^2)+eps;
            sinc=sin(r)./r;
            valData = sinc;
    end
end
```

k) Rezultatul final la lansarea/rularea aplicației:



Bibliografie:

- https://www.mathworks.com/help/matlab/creating_guis/app-designer-code-generation.html
- https://www.mathworks.com/help/matlab/creating_guis/write-callbacks-for-gui-in-app-designer.html
- <https://www.mathworks.com/help/matlab/ref/matlab.ui.control.uicontrol-properties.html>
- https://www.mathworks.com/help/matlab/creating_guis/share-data-across-callbacks-in-app-designer.html
- https://www.mathworks.com/help/matlab/creating_guis/creating-multiwindow-apps-in-app-designer.html

Calcul simbolic

Un obiect simbolic este o structură de date ce conține un șir de caractere reprezentând simbolul. *Symbolic Math Toolbox* utilizează obiecte simbolice pentru reprezentarea variabilelor, expresiilor și matricelor simbolice.

Exemplul 1) Pentru a crea o variabilă simbolică se folosește funcția `sym()` sau comanda `syms`:

```
>> syms a b % Metoda 1) syms
```

```
>> c = a+b
```

```
c = a + b
```

```
>> clear a b; % Metoda 2) sym()
```

```
>> a = sym('a'); b = sym('b');
```

```
>> c = a+b
```

```
c = a + b
```

Name	Class	Value
a	sym	1x1 sym
b	sym	1x1 sym
c	sym	1x1 sym

Exemplul 2) Pentru a crea o expresie simbolică reprezentând ecuația de gradul 2 se folosește funcția `sym()`:

```
>> syms a b c x
```

```
>> f1 = sym(a*x^2+b*x+c) % fără apostrofuri
```

```
f1 = a*x^2+b*x+c % toate variabilele sunt simbolice
```

Name	Class	Value
a	sym	1x1 sym
b	sym	1x1 sym
c	sym	1x1 sym
f1	sym	1x1 sym
x	sym	1x1 sym

Exemplul 3) Pentru a crea o expresie simbolică reprezentând ecuația de gradul 2 se poate scrie expresia ca șir de caractere și apoi se convertește într-o expresie simbolică utilizând funcția `str2sym()`:

```
>> clear, f1 = str2sym('a*x^2+b*x+c') % cu apostrofuri
```

```
f1 = a*x^2+b*x+c % doar variabila f1 este simbolică
```

Name	Class	Value
f1	sym	1x1 sym

Exemplul 4) Pentru substituirea unei variabile se utilizează funcția `subs()`

```
>> clear, syms a b c x
```

```
>> f=sym( a*x^2+b*x+c ); expr=f.^2
```

```
expr = (a*x^2 + b*x + c)^2
```

```
>> f1 = subs(f,a,2)
```

```
f1 = 2*x^2 + b*x + c
```

```
>> f2 = subs(f,[a b c],[1 -1 1])
```

```
f2 = x^2 - x + 1
```

Name	Class	Value
a	sym	1x1 sym
b	sym	1x1 sym
c	sym	1x1 sym
expr	sym	1x1 sym
f	sym	1x1 sym
f1	sym	1x1 sym
f2	sym	1x1 sym
x	sym	1x1 sym

Exemplul 5) Determinarea variabilelor simbolice din cadrul expresiilor se face cu funcția `symvar()`

```
>> clear, syms a b c x
>> f=sym( a*x^2+b*x+c )
>> symvar(f)
ans = [ a, b, c, x]
```

Exemplul 6) Substituirea variabilei x cu 2 din funcția $f(x) = a*x^2+b*x+c$ se face cu funcția `subs()`

```
>> clear, syms a b c x
>> f=sym( a*x^2+b*x+c );
>> rez=subs(f,2) % inlocuieste pe x cu 2
rez = 4*a + 2*b + c
```

Exemplul 7) Substituirea variabilei x cu 2 respectiv cu 3 din funcția $f(x) = a*x^2+b*x+c$ se face cu funcția `subs()`

```
>> clear, syms a b c x
>> f=sym( a*x^2+b*x+c );
>> rez1=subs(f,[2 3]) % inlocuieste pe x cu 2
rez1 = [ 4*a + 2*b + c , 9*a + 3*b + c]
```

Exemplul 8) Dacă expresia funcției $f()$ conține mai mult de o variabilă, adică $f(x,y)=x^2*y+5*x*\sqrt{y}$, se poate specifica variabila care se dorește a fi substituită:

```
>> clear, syms x y
>> f=x^2*y+5*x*sqrt(y);
>> fy = subs(f,x,3)
>> fx = subs(f,y,3)

fy = 9*y+15*y^(1/2)
fx = 3*x^2 + 5*3^(1/2)*x
```

Name	Class	Value
f	sym	1x1 sym
fx	sym	1x1 sym
fy	sym	1x1 sym
x	sym	1x1 sym
y	sym	1x1 sym

Exemplul 9) Substituirea unei variabile simbolice cu o altă variabilă simbolică. Substituirea se face într-o matrice

```
>> clear, syms a b c;
>> A=[a b c; b c a; c a b]
>> syms alpha beta;
>> A(2,3)=beta; % elementul de pe lin=2, col=3
>> A = subs(A,b,alpha)
```

A =

```
[ a, b, c]
[ b, c, a]
[ c, a, b]
```

A =

```
[      a, alpha,      c]
[ alpha,      c,  beta]
[      c,      a, alpha]
```

Name	Class	Value
a	sym	1x1 sym
A	sym	3x3 sym
alpha	sym	1x1 sym
b	sym	1x1 sym
beta	sym	1x1 sym
c	sym	1x1 sym

Exemplul 10) Conversia unei matrice din numeric în simbolic

```
>> A=hilb(3) % matricea de tip Hilbert de ordin 3
```

```
A = 1.0000    0.5000    0.3333
     0.5000    0.3333    0.2500
     0.3333    0.2500    0.2000
```

```
>> A=sym(A) % matricea simbolica corespunzatoare matricei de tip Hilbert
```

```
A = [ 1, 1/2, 1/3]
     [ 1/2, 1/3, 1/4]
     [ 1/3, 1/4, 1/5]
```

Exemplul 11) Construirea numerelor reale și complexe

```
>> syms x y real % Metoda 1) syms
```

```
>> z=x+i*y
```

```
>> conj(z)
```

```
z = x + y*1i
ans = x - y*1i
```

Workspace		
Name ^	Class	Value
ans	sym	1x1 sym
x	sym	1x1 sym
y	sym	1x1 sym
z	sym	1x1 sym

```
>> x=sym('x','real'); y=sym('y','real'); % Metoda 2) sym()
>> z=x+i*y;
>> expand(z*conj(z))
```

```
ans = x^2+y^2 % Exemplu: expand((x+1)^3) ⇔ x^3+3*x^2+3*x+1
```

Exemplul 12) Pentru calculul derivatelor funcțiilor de o singură variabilă se utilizează funcția `diff()`

```
>> clear, syms x
>> f=x^3+2*x^2-x+4
>> df = diff(f) % derivata de ordinul I
```

```
f = x^3+2*x^2-x+4
df = 3*x^2+4*x-1
```

```
>> clear, syms x; f=x^3+2*x^2-x+4;
>> diff(f,2) % derivata de ordinul II
```

```
ans = 6*x + 4
```

Exemplul 13) Pentru calculul derivatelor funcțiilor de mai multe variabile se utilizează funcția `diff()`

```
>> clear, syms x y; f=x^3*y+2*x^2+y-x+4
>> df2x = diff(f,x,2), df1y = diff(f,y,1)
```

```
f = y*x^3 + 2*x^2 - x + y + 4
df2x = 6*x*y + 4
df1y = x^3+1
```

The screenshot shows the MATLAB Workspace window with the following content:

Name	Class	Value
df1y	sym	1x1 sym
df2x	sym	1x1 sym
f	sym	1x1 sym
x	sym	1x1 sym
y	sym	1x1 sym

Exemplul 14) Simplificarea unei expresii se face cu funcția `simplify()` și expandarea (nesimplificat) se face cu funcția `expand()`

```
>> syms x
>> f=(x+1)^2+(x+1)^3
>> simplify(f), expand(f)
```

```
f = (x + 1)^2 + (x + 1)^3
ans = (x + 1)^2*(x + 2)
ans = x^3 + 4*x^2 + 5*x + 2
```

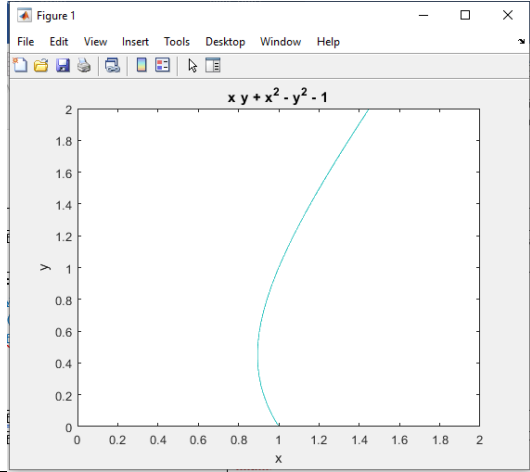
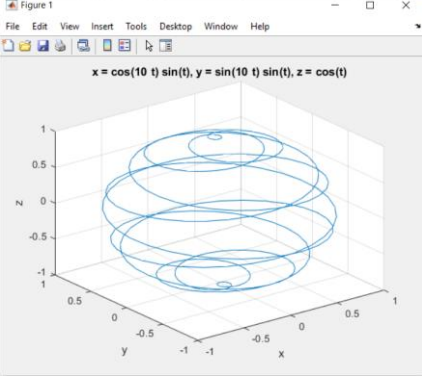
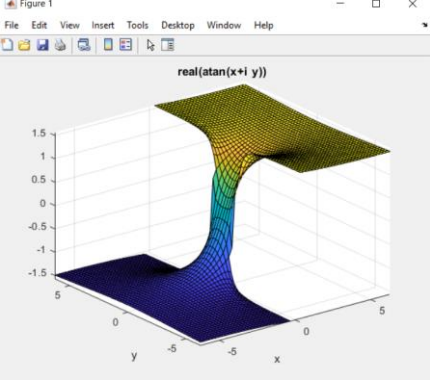
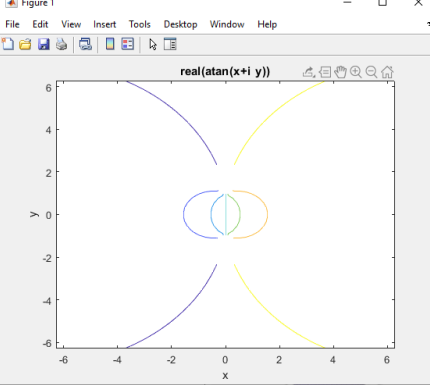
Exemplul 15) Valoarea numerică a unui obiect simbolic se obține cu funcția `double()`

```
>> rez=sqrt(sym(2))
rez = 2^(1/2)

>> double(rez)
ans = 1.4142
```

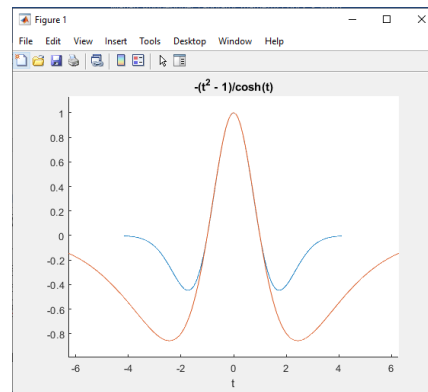
Reprezentări grafice ale obiectelor simbolice

Sintaxe	Descriere
<code>ezplot(f)</code>	reprezintă grafic funcția $f(x)$ peste domeniul inițial $x \in (-2\pi, 2\pi)$
<code>ezplot(f1, [a,b])</code>	reprezintă grafic funcția $f1(x)$ peste dom. $x \in (a,b)$
<code>ezplot(f2)</code>	reprezintă grafic funcția $f2(x,y)$ peste domeniul inițial $x \in (-2\pi, 2\pi)$
<code>ezplot(f2, [a,b])</code>	reprezintă grafic funcția $f2(x,y)$ peste dom. $x,y \in (a,b)$
<code>ezplot(f2, [ax,bx,ay,by])</code>	reprezintă grafic funcția $f2(x,y)$ peste domeniul $x \in (ax,bx), y \in (ay,by)$

<p>Exemplu:</p> <pre>syms x y fct=sym(x*y + x^2 - y^2 - 1); ezplot(fct,[0,2])</pre>	
<p>ezplot3(x,y,z,[ax,bx,ay,by,az,bz])</p>	<p>reprezintă grafic funcția $f_3(x,y,z)$ în spațiu peste domeniul $x \in (ax,bx)$, $y \in (ay,by)$, $z \in (az,bz)$</p>
<p>Exemplu:</p> <pre>syms t x y z x = sin(t).*cos(10*t); y = sin(t).*sin(10*t); z = cos(t); ezplot3(x,y,z)</pre>	
<p>ezsurf(f4) ezsurf(f4,[a,b,c,d],NR)</p>	<p>reprezintă suprafața $f_4(x,y)$ în spațiu peste domeniul $x \in (a,b)$, $y \in (c,d)$</p>
<p>Exemplu:</p> <pre>syms x y ezsurf('real(atan(x+i*y))')</pre>	
<p>ezcontour(f2)</p>	<p>reprezintă liniile de contur ale unui grafic</p>
<p>Exemplu:</p> <pre>syms x y ezcontour('real(atan(x+i*y))')</pre>	

Exemplul 16) Reprezentarea a două funcții simbolice în același sistem de axe

```
clear, syms t
f1=sym((1-t^2)*exp(-1/2*t^2))
f2=sym((1-t^2)*sech(t))
hold on;
ezplot(f1);
ezplot(f2);
hold off;
```

**Aplicații de laborator:**

1. Să se reprezinte grafic curba dată prin *ecuația simbolică*:

$$\begin{cases} x = 1 + |t| \\ y = |1 - t^2|, \end{cases} \quad \text{pentru } t \in [-1, 1]$$

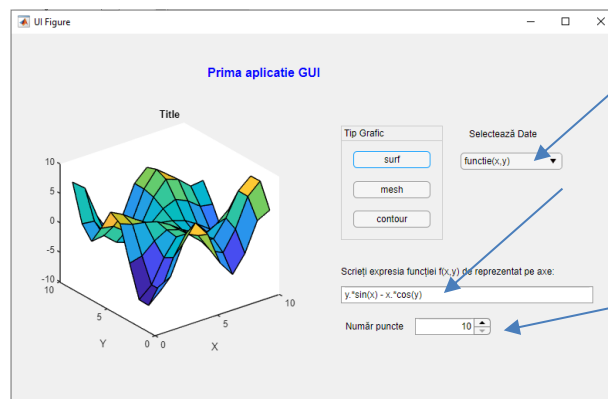
2. Scrieți un program în Matlab care să reprezinte grafic funcția $f(x, y) = \frac{1}{2\pi} e^{-\frac{1}{2}(x^2+y^2)}$,

pe domeniul $D = [-1, 1] \times [-1, 1]$, în două subgrafice alăturate, folosind funcțiile *ezsurf* și *ezcontour*. În fiecare domeniu avem 30 de puncte.

3. Pornind de la aplicația GUI *FirstApp*, prezentată la începutul laboratorului (pagina 2), se cere să se completeze interfața aplicației cu:

- un obiect grafic *Edit Field (Text)* prin care să se permită citirea de către utilizator a funcției ce se va reprezenta pe grafic (în sistemul de axe).
- un obiect grafic *Spinner* cu limitele 10, 50 și pas 1.

Pentru a prelua expresia funcției de la utilizator, la lista DropDown se va adăuga o nouă opțiune, numită *functie(x,y)*. La activarea opțiunii se va prelua expresia funcției scrisă de utilizator, **se va converti la o funcție simbolică** și se va reprezenta pe grafic. Pentru desenare se vor folosi cele 3 butoane existente.



Exemple de testat:

$$f1(x, y) = (x^3 y - x y^3) / (x^2 + y^2)$$

$$f2(x, y) = y \sin(x) - x \cos(y)$$

$$f3(x, y) = x \exp(-x^2 - y^2)$$

$$f4(x, y) = (x^2 - y^2) / (x^2 + y^2)$$

$$f5(x, y) = (y(x^4 + 4x^2 y^2 - y^4)) / (x^2 + y^2)^2$$

$$f6(x, y) = \sin(x) + \sin(y) - (x^2 + y^2) / 20$$

Indicație: La selectarea opțiunii '*funcție(x,y)*' din lista *DropDown* se execută următorii pași:

Pas 1) se citește valoarea din *Spinner* și se salvează în variabila *puncte*;

Pas 2) se inițializează variabilele *x* și *y* în domeniul dorit pentru numărul de elemente alese din *Spinner*

`x=linspace(-5,5,puncte);`

`y=linspace(-5,5,puncte);`

Pas 3) se extind vectorii *x*, *y* la două matrice utilizând funcția *meshgrid()*

`[x,y]=meshgrid(x,y);`

Pas 4) se convertesc variabilele *x* și *y* în variabile simbolice

`x = sym(x); y = sym(y);`

Pas 5) se citește expresia funcției și se transformă în expresie simbolică

`f = str2sym(lower(app.FunctieEditField.Value));`

Pas 6) se substituie variabilele *x* și *y* din funcție cu matricele obținute la pasul 4)

`f=subs(f);`

Pas 7) se transformă rezultatul funcției din matrice simbolică în matrice numerică utilizând funcția *double()*

`valData = double(f); % sau, app.Z = double(f);`