Toolbox za rešavanje parcijalnih diferencijalnih jednačina u MATLABu

Toolbox za rešavanje parcijalnih diferencijalnih jednačina u MATLABu može se koristiti za rešavanje parcijalnih diferencijalnih jednačina (PDJ) i u 2D i u 3D (oblik tela koji za koji ćemo rešavati problem ćemo u daljem tekstu nazivati geometrijom i obeležavati sa Ω). Rešenja problema dobijena su korišćenjem metode konačnih elemenata.
Toolbox omogućava da se napravi dvodimenzionalni oblik ili trodimenzionalno telo, da se formulišu granični uslovi i da se definiše sama PDJ jednačina. Formulisan problem može da se reši korišćenjem scrip fajla ili preko same PDE aplikacije, a dobijenno rešenje može da se vizuealizuje i prikaže matrično.

Toolbox za rešavanje PDJ može se podjednako koristiti za standardne probleme, poput problema provođenja toplote, problem difuzije, elektrostatike, magnetizma, dinamike, protoka ali i za proizvoljno konstruisane sisteme PDJ.

- Da bi se primenio Toolbox, potrebno je da PDJ bude eliptičkog, paraboličkog ili hiperboličkog tipa.
- Dozvoljeno je korišćenje granični uslovi koji su Dirihleovog, Nojmanovog ili mešovitog tipa.
- Geometrije nad kojima se rešava problem mogu biti dvodimenzionalne i trodimenzionalne (pri čemu su 3D geometrije učitane iz STL baze).
- Mreža se formira automatski korišćenjem tetraedara i trouglova.
- Rešenje se vizuelizuje po više različitih kriterijuma.

Predviđeno je da su jednačine date u sledećem formatu:

- Jednačine eliptičkog tipa $-\Delta(c\Delta u) + au = f$
- Jednačine paraboličkog tipa $d\,\frac{\partial u}{\partial t} - \Delta(c\Delta u) + au = f$
- Jednačine hiperboličkog tipa $d\frac{\partial^2 u}{\partial t^2} - \Delta(c\Delta u) + au = f$
- Problem sopstvenih vrednosti $\Delta(c\Delta u) + au = \lambda du$

Kod PDJ eliptičkog tipa

- geometrija Ω može biti konstruisana i u 2D i u 3D,
- c, a, f i u predstavljaju skalare, sa tima da c može biti zadato i matrično.

Kod PDJ hiperboličkog i paraboličkog tipa,

- geometrija Ω može biti konstruisana i u 2D i u 3D,
- c, a, f i d mogu da zavise od vremena, od rešenja u, i od gradijenta Δu.

Ukoliko neki od koeficijenata c, a, f i d zavise od u ili njegovog gradijenta, koristi se solver za rešavanje nelinearnih jednačina (pdenonlin). Na primer, za nelinearne eliptičke PDJ

$$-\Delta(c(u)\Delta u) + a(u)u = f(u),$$

gde su c, a i f funkcije od u i Δu treba da se koristi pdenonlin.

Kako se rešavaju problemi PDJ?

1. Prvo se definiše geometrija Ω.
   Geometrija Ω može da se konstruiše korišćenjem ugrađenih oblika (krug, kvadrat, elipsa, poligon), njihovom kombinacijom ili korišćenjem gotovih CSG modela.
2. Zatim se definišu granični uslovi. Moguće je da se definišu različiti granični uslovi na različitim segmentima geometrije.
3. Definišu se koeficijenti PDJ, tj. Dodele se vrednosti koeficijentima c, a, f i d.
4. Kreira se mreža
5. Reši se PDJ preko ugrađenog solvera za rešavanje PDJ.
6. Nacrta se rešenje ili se vrati matrica rešenja.

U slučaju da niste zadovoljni kvalitetotm rešenja, dozvoljeno je „profinjivanje“ mreže i ponovno rešavanje. Takođe, može da se primeni i adaptivna mreža naredbom adaptmesh (koja pokušava da pronađe oblik koji najviše odgovara problemu).

PRIMER:

Rešiti Poasonovu jednačinu $-\Delta^2 \mathrm{u} = 1$ koja na jediničnom disku zadovoljava Dirihleove granične uslove, tj. rešiti sledeći problem

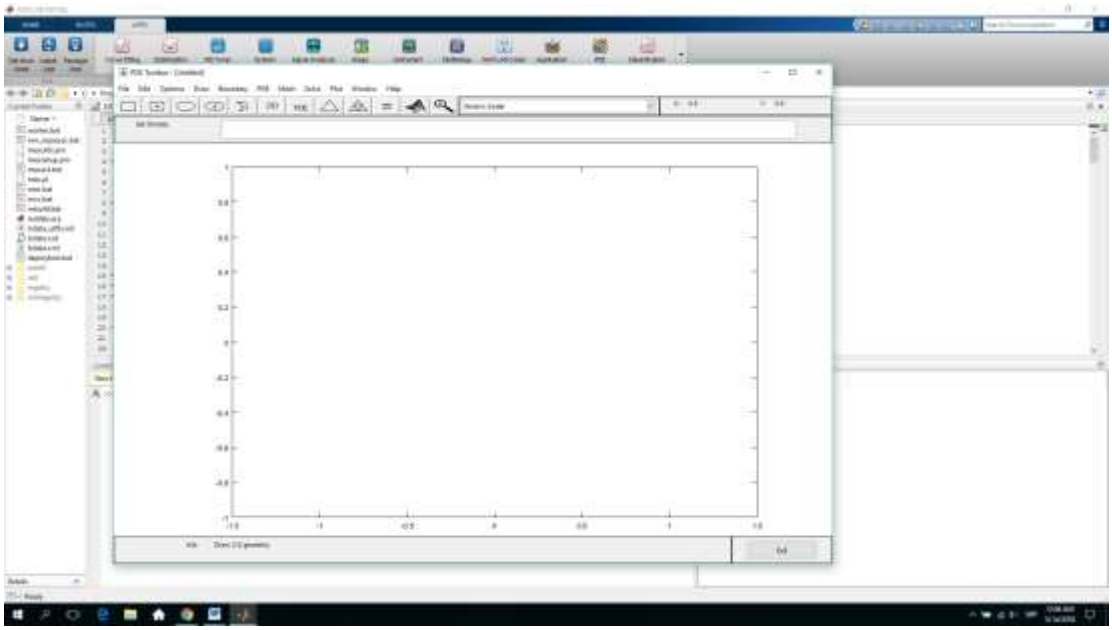
$$\Delta^2 u = 1, \qquad (x, y) \in \Omega$$
$$u(x) = 0, \qquad (x, y) \in \partial\Omega$$

Rešenje zadatog problema je

$$u(x, y) = \frac{1 - x^2 - y^2}{4}$$

Postupak u MATLABu:

Pokrenuti PDE aplikaciju kucanjem naredbe pdetool u komandnom prozoru (ili klikom na Apps tab i izborom polja PDE pod “Math, statistics and optimization”), videti Sliku niže.



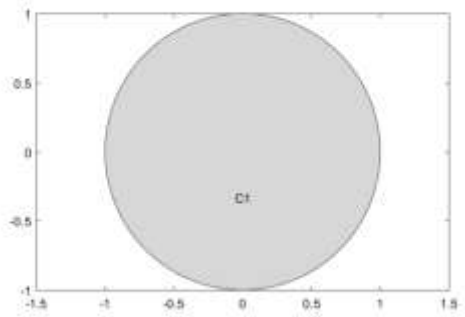
Za crtanje mreže dovoljno je da se selektuje opcija Grid iz Options menija.
Opcija Snap to grid omogućava da se mreža poravna sa geometrijom koju koristimo.
Koraci mreže mogu da se promene opcijom Grid spasing, a koordinatne ose opcijom Axes limits.

Geometrija se formira korišćenjem već gotovih oblika, kruga, pravougaonika, elipse i poligona.
Na primer: da bi se formirao krug dovoljno je da se u komandnom prozoru otkuca naredba pdecirc(xc,yc,radius), gde su **xc** i **yc** kordinate centra kruga a **radius** poluprečnik. Krug sa centrom u koordinatnom početku i jedinočnog poluprečnika se dobija naredbom

```
>> pdecirc(0,0,1)
```

Inače, naredbom pdecirc se automatski pokreće i aplikacija za rešavanje PDJ tako da je njeno prethodno pokretanje naredbom pdetool bilo nepotrebno.
Ukoliko želite da sačuvate rad pod nazivom primer1, možete da formirate fajl primer1.m i da, umesto u komandnom prozoru otkucate liniju pdecirc(0,0,1) a fajl pokrenete pozivom

```
>> primer1
```



Koordinate osa možemo da promenimo opcijom Options – Axes Limits i postavljanjem osega za X i Y na, npr.
X – axis range: [-2 2]
Y – axis range: [-2,2]
i klikom na ok (kod mene taj krug počne da liči na elipsu, ako vam to ne smeta, ako vam smeta, kliknete na opciju Axis equal I sve će biti ok).

*Ukoliko je potrebno da se formira PDE model sa elipsom, u komandnom prozoru kucati pdeellip(xc,yc, a,b,phi) , pri cemu su **xc** i **yc** coordinate kruga, **a** i **b p** poluose a **phi** ugao rotacije, npr:

```
>> pdeellip(0,0,1,0.3,pi)
```

*Za fomiranje kvadrata, dovoljno je da se u komandnom prozoru otkuca pderect(xy), gde su **x** i **y** koordinate donjeg levog i gornjeg desnog ugla. npr.

```
>> pderect([-1.5,0,-1,0])
```

Formiranje kruga, elipse i kvadrata moguće je i direktno preko aplikacije tako što se:

1. Izabere oblik (krug, elipsa, kvadrat ili poligon) korišćenjem draw menija na toolbox dugmićima.
2. Klikne i prevuče na pde aplikaciju.
   Dvoklikom na objekat otvara se polje sa informacijama o tom objektu u kome se vrednosti svih parametara mogu promeniti. Jednim klikom na polje i na opciju DELETE nacrtan obik može da se obriše.
3. Oblici mogu da se kombinuju korišćenjem polja: Set formula (biće reči o ovoj opciji kasnije).

Nakon što smo nacrtali geometriju sa kojom želimo da radimo, dodelimo vrednosti koeficijentima:
PDJ eliptičkog tipa je oblika $-\nabla(c\nabla u) + au = f$
Dakle, da bi PDJ odgovarala našem problemu, uzećemo da su $c = 1, a = 0$ i $f = 1$.

Vrednosti koeficijenata PDJ se mogu dodati direktno preko PDE aplikacije kllikom na PDE – PDE Specification i izborom opcije Eliptic (biće nam dozvoljeno da popunimo vrednosti koeficijenata c,a i f koje su trenutno postavljene na 1, 0 i 10) .

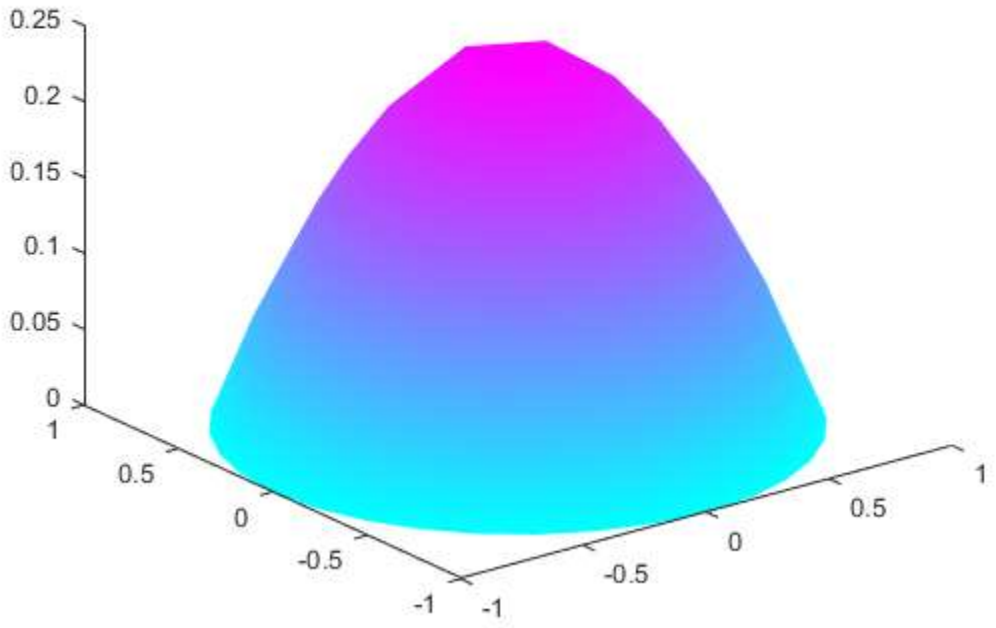
Ili dodeljivanjem vrednosti skalarima c=1, a=0 i f=1 u fajlu primer1.m

Ukoliko se odlučimo da pišemo script fajl, potrebno je da učitamo geometriju, da je prikažemo, dodelimo vrednosti koeficijentima PDJ i dodelimo vrednosti graničnim uslovima

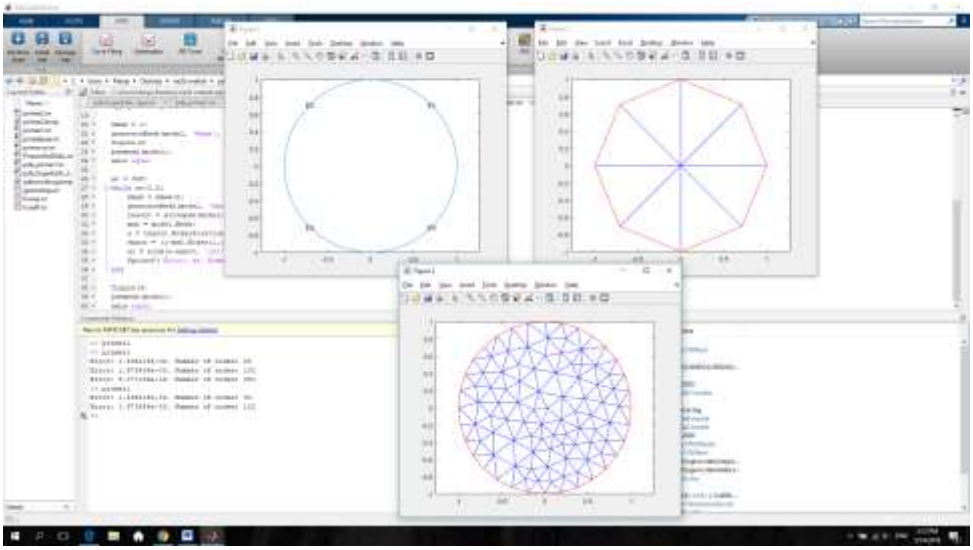
primer1.m

```
g = @circleg;
c = 1;
a = 0;
f = 1;
figure(1)                               % crta figuru
pdegplot(g, 'EdgeLabels', 'on');        % dodeljuje ime graničnim segmentima
axis equal                              % podešava da granice budu jednake

numberOfPDE = 1;                        % broj PDJ koji želimo da rešavamo
model = createpde(numberOfPDE);         % kreiramo problem pod imenom model

geometryFromEdges(model, g);            % učitamo geometriju

specifyCoefficients(model, 'm', 0, 'd', 0, 'c', c, 'a', a, 'f', f);
                                        % dodelimo vrednosti koeficijentima.
                                        % kod script  fajla MATLAB ne zna da mi
                                        % rešavamo PDJ eliptičkog tipa pa nam omogućava
                                        % da unesemo vrednosti svih koeficijenata
applyBoundaryCondition(model, 'dirichlet', 'Edge', (1:4), 'u', 0);
                                        % dodelimo Dirihleove granične uslove na sve 4
                                        % grane. Ukoliko ste crtali primetićete da se
                                        % krug sastoji iz 4 luka. Uslov u(x,y)=0 se
                                        % odnosi na sva 4 luka.
hmax = 1;
generateMesh(model, 'Hmax', hmax);      % generišemo mrežu kod koje ni jedna stranica
                                        % trougla nije duža od 1.

figure(2)                               % na drugoj slici (ovo je samo zbog
                                        % preglednosti, inače sve može da se crta na
                                        % jednoj figuri.

pdemesh(model);                         % dodelimo mrežu našem problemu
axis equal
                                        % rešavamo problem korišćenjem metode konačnih
                                        % elemenata tako što u svakoj iteraciji pravimo
                                        % finiju mrežu sve dok razlika između našeg,
                                        % numeričkog rešenja i tačnog rešenja ne bude
                                        % manja od 0.01
er = Inf;
while er>0.001
      hmax = hmax/2;
      generateMesh(model, 'Hmax', hmax);
      result = solvepde(model);             % resave problem
      msh = model.Mesh;
      u = result.NodalSolution;             % rešenje problema pamtimo kao u
                                            % rešenje ćemo dobiti u temenima trougla
                                            % računamo tačno rešenje u istim tačkama
                                            % msh.Nodes(1,:) predstavlja niz x
                                            % koordinata
                                            % msh.Nodes(2,:) predstavlja niz y
                                            % koordinata
      tacno = (1-msh.Nodes(1,:).^2-msh.Nodes(2,:).^2)'/4;
```

```
        er = norm(u-tacno, 'inf');
        fprintf('Greska: %e. Broj cvorova: %d\n', er, size(msh.Nodes, 2));
    end
    figure(3)
    pdemesh(model);
    axis equal

    figure(4)
    pdeplot(model, 'XYData',u,'ZData',u,'ColorBar','off');
```



Ako se pokrene fajl primer1.m dobiće se sledeći prozori:



Ako bi rešavali preko aplikacije, MATLAB će sam rešiti problem optimalno. U okviru while petlje mi smo zapravo proveravali koliko finu mrežu treba da napravimo kako bi rešiil problem sa zadatom tačnošću (neće biti na kolokvijumu).

pdegplot(g, 'EdgeLabels', 'on'); - dobija se klikom na Bounday Mode – Show Edge Labels
specifyCoefficients(model, 'm', 0, 'd', 0, 'c', c, 'a', a, 'f', f); - PDE - PDE Specification a zatim se izabere opcija Elliptic.
applyBoundaryCondition(model, 'dirichlet', 'Edge', (1:4), 'u', 0); - Boundary – Specify Boundary Condition, a zatim se izabere opcija Dirichet i unesu vrednosti koeficijenata, tj. postavi se h na 1 a r na 0.

Mreža se generiše naredbom Mesh – Initialize Mesh.
Meža se profini naredbom Mesh – Refine Mesh ili Jiggle Mesh (Jiggle Mesh omogućava da se formira mreža koja više odgovara problemu).

Problem se rešava klikom na polje „=“ ili klikom na opciju SOLVE – SOLVE PDE.

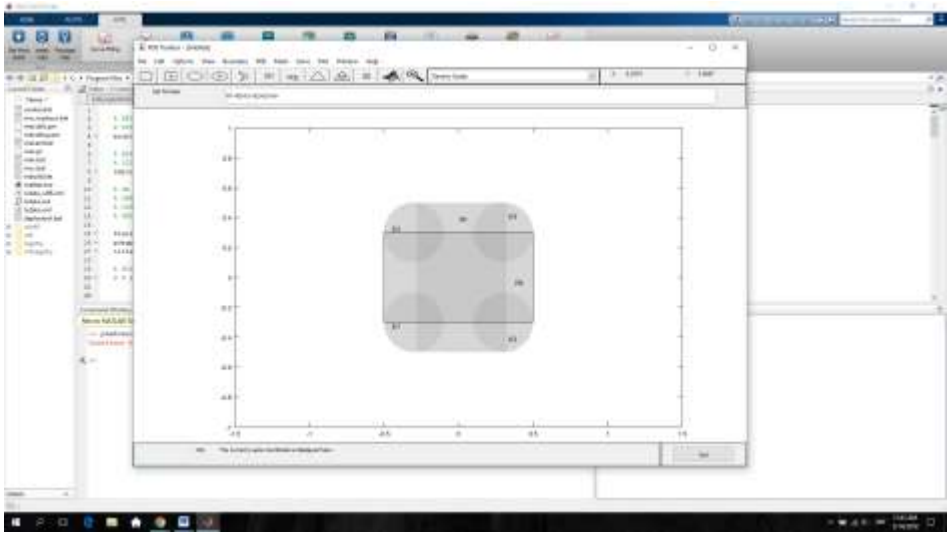
Crtanje problema (pdeplot) omogućava crtanje problema u 3D formatu opcija je Plot – Plot Parametres gde se označi opcija Height (3-D plot).
U istom prozoru može da se promeni način prikaza (da se odčekira boja i da se čekira deo za konture ili strelice), može da se promeni način crtanja (Plot style). Colormap može biti jiggle umesto cool i sl... Probajte različite kombinacije sami.

Primer2:

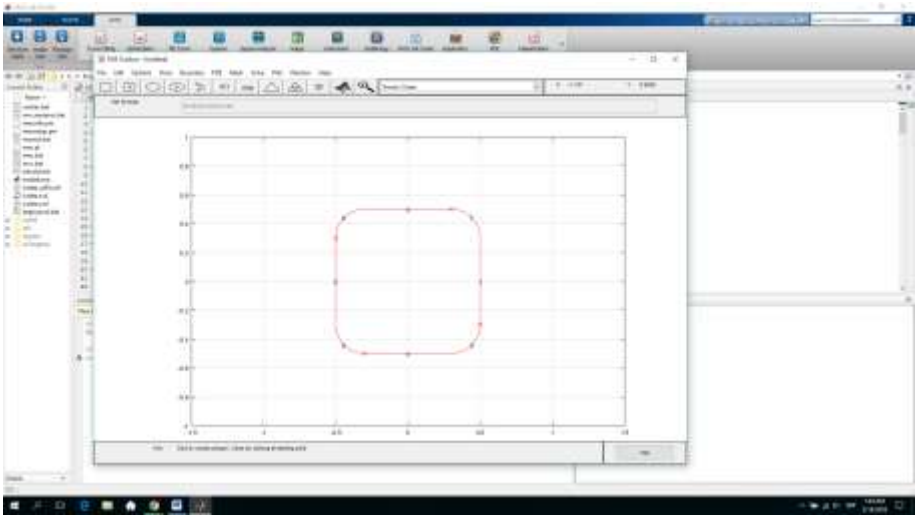
Umesto kruga, možemo nacrtati kvadrat sa oblim uglovima tako što prvo nacrtamo dva pravougaonika (neka su koordinate donjeg levog ugla prvog pravougaonika (-0.3,-0.5) i neka je dužine 0.6 i visine 1, a koordinate donjeg levog ugla drugog pravougaonika (-0.5, -0.3) i neka je je on dužine 1 I visine 0.6, a zatim, nacrtati 4. kruga sa centrima u preseku stranica ova dva kvadrata, poluprečnika 0.2.

Kvadrati će dobiti oznaku R1 i R2 dok će krugovi biti označeni sa E1, E2, E3 i E4 (dvoklikom na oblik otvoriće se prozor sa podacima o nacrtanom obliku, za krug će biti poznate koordinate centra i dužine poluosa, za pravougaonike dužina stranica). Oblici mogu da se selektuju istovremeno tako što se drži dugme SHIFT i desnim klikom obeleže oblici koje želimo istovremeno da selektujemo. Takodje, može da se koristi opcija SELECT ALL iz padajućeg menija kod polja EDIT.

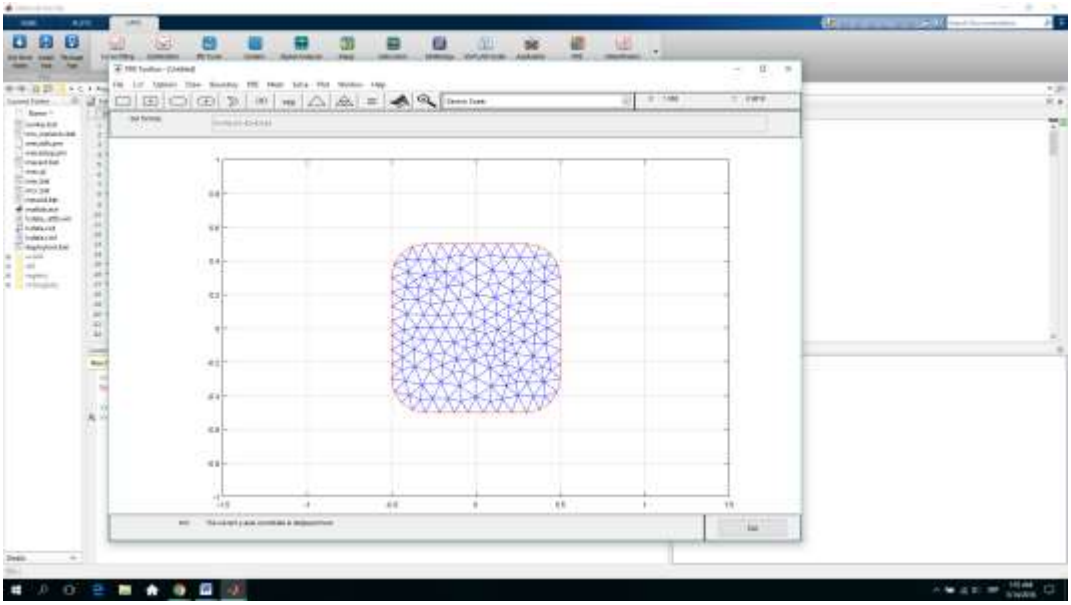
Ako u polju Set Formula napišemo R1+R2+E1+E2+E3+E4 dobićemo oblik koji predstavlja uniju nacrtanih oblika. Znak „-„ predstavlja razliku oblika. Dobijeni oblik možemo da zapamtimo naredbom SAVE.



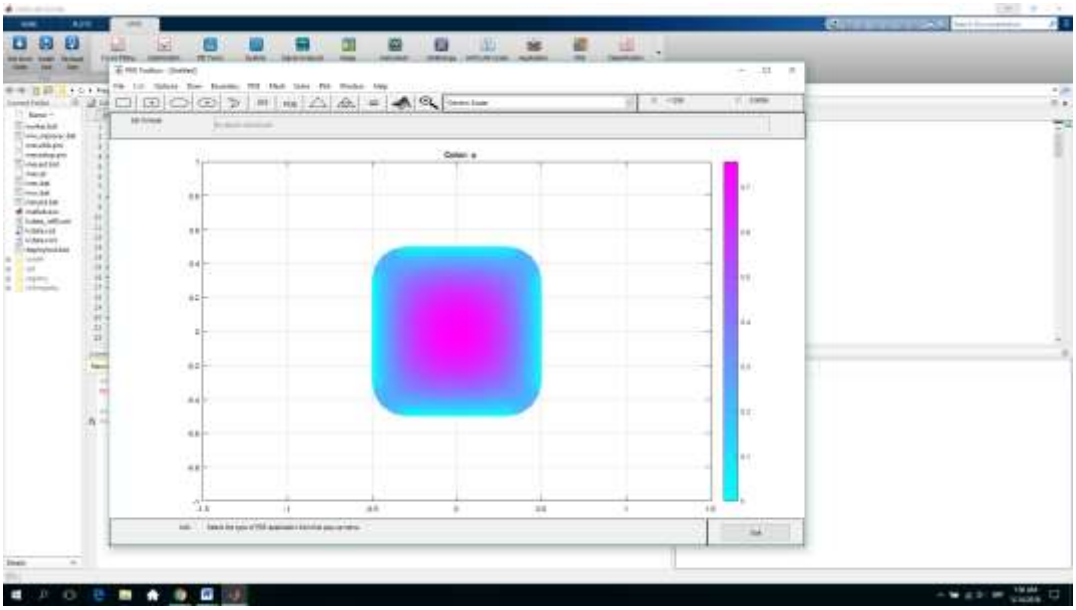
Prelaskom na Boundary mode videćemo koje su granice našeg oblika.
Levim klikom na granicu i klikom na polje „Remove subdomain border“ (padajući meni polja Boudary), granica će biti obrisana. Kllikom na polje „Remove All Subomain Border“ (Boundary meni) biće obrisane sve unutrašnje granice (videti sliku niže).



Klikom na „Show Edge Labels“ biće prikazani nazivi krivih koje čine naš oblik.
Opcija „Specify Boundary Condition“ omogućava da se zadaju granični uslovi koji su jednaki za sve granice.
Otvoriće se polje koje će nam omogućiti dva različita tipa graničnih uslova, Nojmanove i Dirihleove uslove.
U oba slučaja prikazaće formulu i omogućiti da se unesu odgovarajući koeficijenti.
Recimo da želimo da postavimo Dirihleove uslove. Ako ni jedne uslove ne zadamo podrazumevaće se da Dirihleovi uslovi $u = 0$.
Međutim, ukoliko ne želite da dodelite iste granične uslove svim stranama, dovoljno je da se dva puta klikne na granu kojoj želite da dodelite uslove i popuni prozor Boundary Condition. Neka su na granama 5,6,7 i 8 Nojmanovi uslovi (q =5, q=1) a na granama 1,2,3 i 4 Dirihleovi (h =1, r=5). Primetićemo da je boja promenjena. Granice koje imaju Dirihleove uslve su označene crvenom bojom, granice sa Nojmanovom su plave dok su granice sa mešovitim uslovima zelene.
Tip PDJ se određuje iz polja PDE – PDE SPECIFICATION. Ovde ćemo izabrati da je naš problem eliptičkog tipa. Uzećemo da su $c = 1, a = 0, f = 10$.
Konačno, klikom na trouglić (ili izborom opcije Mesh – INITIAL MESH) inicijalizovaćemo mrežu. Mreža se može profiniti opcijom REFINE MESH iz MESH padajućeg menija. Povratak na prethodnu mrežu se omogućava opcijom UNDO MESH CHANGE.
Mi ćemo izabrati opciju JIGGLE MESH koja poboljšava kvalitet trijangulacije.



Da bismo rešili PDJ, dovoljno je kliknuti na polje „=“ ili na opciju SOLVE – SOLVE PDE



Rešenje problema se može dobiti i vektorski. Prilikom crtanja rešenja, po pravilu se koriste boje. Stil prikazivanja rešenja se može promeniti promenom Parametara u meniju PLOT.

Primer3:

Rešiti sledeću PDJ hiperboličkog tipa na pravougaoniku sa rupom u sredini

$$5\,\frac{\partial u}{\partial t} - \nabla\big((1 + x^2 + y^2)\nabla u\big) = f(t)$$

gde je

$$f = 10 * \begin{cases} 10t, & 0 \le t \le 0.1 \\ 1, & 0.1 \le t \le 0.9 \\ 10 - 10t, & 0.9 \le t \le 1 \end{cases}$$

za $(x, y) \in \Omega$

$u(x, y) = t(x - y)$ na stranama pravougaonika [1,4] (Dirihleovi uslovi)

$q = 1, q = x^2 + y^2$ na krugu (Nojmanovi uslovi)

Početni uslovi su

$u(x, y) = 0$, kada je $t = 0$

Dakle,

$$d = 5$$
$$a = 0$$
$$c = 1 + x^2 + y^2$$

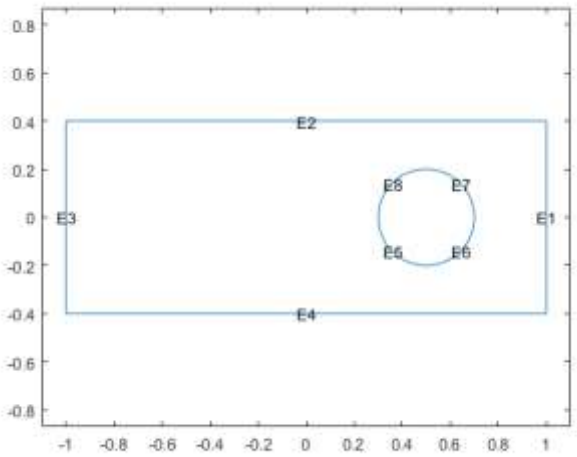
Neka je oblast Ω oblika pravougaonika sa krugom u sredini (rupom)

Da bi geometriju sačuvali pišemo sledeće:

```
model = createpde(1);
% Formiramo pravougaonik sa koordinatama:
R1 = [3,4,-1,1,1,-1,-.4,-.4,.4,.4]';
% Formiramo krug sa centrom u tacki (.5,0), poluprecnika .2
C1 = [1,.5,0,.2]';
% Promenljivoj C1 dodeljujemo nule sa desne strane kako bi je povezali u geometriju
C1 = [C1;zeros(length(R1)-length(C1),1)];
geom = [R1,C1];
% Dodeljujemo imena objektima
```

```
ns = (char('R1','C1'))';
% Podesavamo formula (Set formula)
sf = 'R1 - C1';
% Kreiramo oblik
gd = decsg(geom,sf,ns);
% Ukljucujemo oblik koji smo upravo nacrtali u nas problem
geometryFromEdges(model,gd);
% Ukljucujemo nazive granica
pdegplot(model,'EdgeLabels','on')
xlim([-1.1 1.1])
axis equal
```

% sve od navedenog smo mogli da odradimo I preko PDE aplikacije



Da bi funkciju f definisali korektno uvešćemo pomoćnu funkciju (ovaj deo kod kucamo na kraj fajla primer3.m

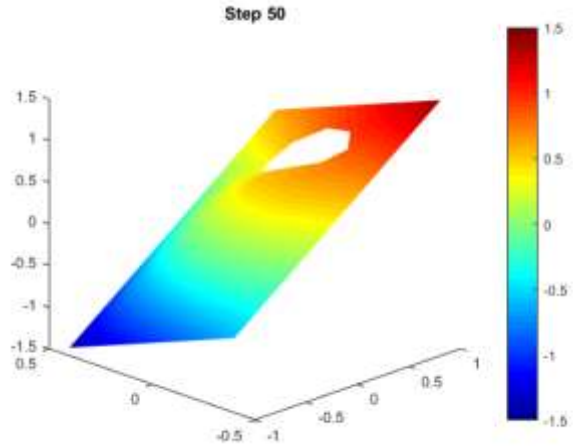
```
function f = framp(t)
if t <= 0.1
 f = 10*t;
elseif t <= 0.9
 f = 1;
else
 f = 10-10*t;
end
f = 10*f;
end
```

fajl primer3.m je sledećeg oblika:

```
model = createpde(1);
% Crtamo pravougaonik
R1 = [3,4,-1,1,1,-1,-.4,-.4,.4,.4]';
% Crtamo krug
C1 = [1,.5,0,.2]';
% Popunjavamo vector C1 nulama kako bi dobili iste dimenzije
C1 = [C1;zeros(length(R1)-length(C1),1)];
geom = [R1,C1];          % Lepimo pravougaonik I krug u istu geometriju
ns = (char('R1','C1'))'; % dodeljujemo im ime
% Set formula
sf = 'R1 - C1';    % Nasu geometriju cini pravougaonik sa rupom, zato oduzimamo krug
% Kreiramo oblik decsg(geometrija, formula, naziv geometrije)
gd = decsg(geom,sf,ns);
% Importujemo geometriju u problem
```

```
geometryFromEdges(model,gd);
% Želimo da vidimo kako se koja kriva zove
pdegplot(model,'EdgeLabels','on')
xlim([-1.1 1.1])
axis equal

 % formiramo pomoćne funkcije u i g
myufun = @(region,state)state.time*(region.x - region.y);
mygfun = @(region,state)(region.x.^2 + region.y.^2);
% dodeljujemo granične uslove
applyBoundaryCondition(model,'edge',1:4,'u',myufun,'Vectorized','on');
applyBoundaryCondition(model,'edge',5:8,'q',1,'g',mygfun,'Vectorized','on');
 % dodeljujemo početne uslove
u0 = 0;
 % generišemo mrežu
generateMesh(model);
% formiramo niz vremena t
vreme = linspace(0,1,50);

d = 5;
a = 0;
f = 'framp(t)';
c = '1 + x.^2 + y.^2';

% rešavamo problem
u = parabolic(u0,vreme,model,c,a,f,d);

% crtamo problem za različite vremenske intervale
for tt = 1:size(u,2) % različiti vremenski intervali
 pdeplot(model,'xydata',u(:,tt),'zdata',u(:,tt),'colormap','jet')
 axis([-1 1 -1/2 1/2 -1.5 1.5 -1.5 1.5]) % fiksirane ose
 title(['Vreme ' num2str(tt)])
 view(-45,22)   % menjamo ugao iz kog posmatramo telo
 drawnow          % animiramo rešenje sa pauzom 0.1
 pause(.1)
end


function f = framp(t)

if t<=0.1
    f = 10*t;
elseif t<=0.9
    f = 1;
else
    f = 10-10*t;
end
f = 10*f;
end
```

Nakon 50 iteracija:



Funkcija c je mogla da se zada i na sledeći način:

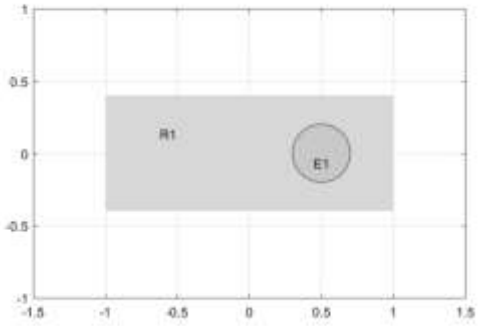
```
function c = cfunc(p,t,u,time)
% Temena trougla
it1 = t(1,:);
it2 = t(2,:);
it3 = t(3,:);
% Pronađemo cenatr trougla
xpts = (p(1,it1) + p(1,it2) + p(1,it3))/3;
ypts = (p(2,it1) + p(2,it2) + p(2,it3))/3;
c = 1 + xpts.^2 + ypts.^2;
end


U tom slučaju bi funkciju c pozvali na sledeći način
c = @cfunc;
u = parabolic(u0,tlist,model,c,a,f,d);
```

Kako bi problem rešili preko aplikacije?

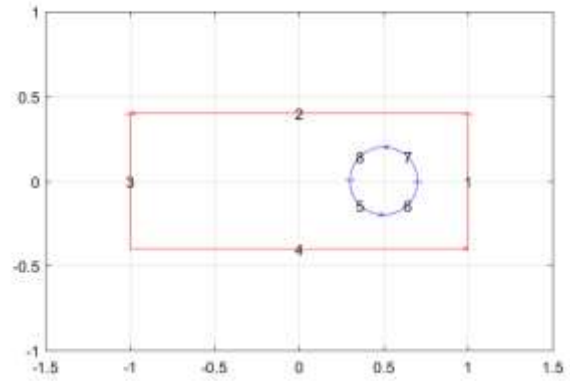
Napravimo fajl framp.m i pozicioniramo se na folder u kome se taj fajl nalazi.
Pokrenemo pdetool
Kliknempo na opciju PDE – PDE Specification, selektujemo Parabolic i popunimo polja vezana za koeficijente

c = 1+x.^2+y.^2 % Bez ikakvih navodnika ili znakova jednakosti
a = 0
f = framp(t)
d = 5

Kliknemo na ok.
Selektujemo opciju Options – Grid i Options – Snap kako bi nalepili mrežu na našu geometriju.
Selektujemo opciju Draw – Draw mode
Nacrtamo pravougaonik i krug.
Postavimo formulu Set formula na R1-E1 (bez = opet)

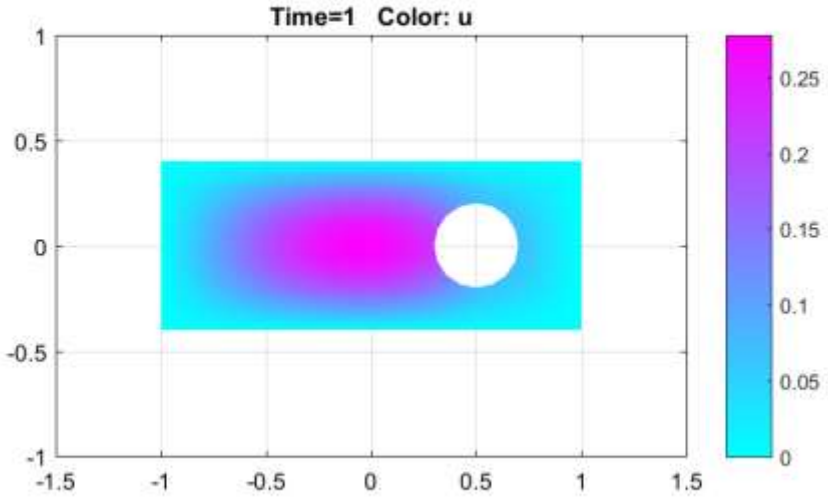


Selektujemo Boundaru – Booundary Mode
Boundary – Show Edge Labels
Levim klikom uz Shift označimo sve 4 strane pravougaonika i dvoklikom otvorimo prozor u koji ćemo uneti granične uslove. Označimo Dirihleove uslove i unesemo
h=1
r =t(x-y)
Zatim selektrujemo krug, označimo Nojmanove uslove i stavimo da je
g=x.^2+y.^2 (samo popunjavamo bez =)
q =1

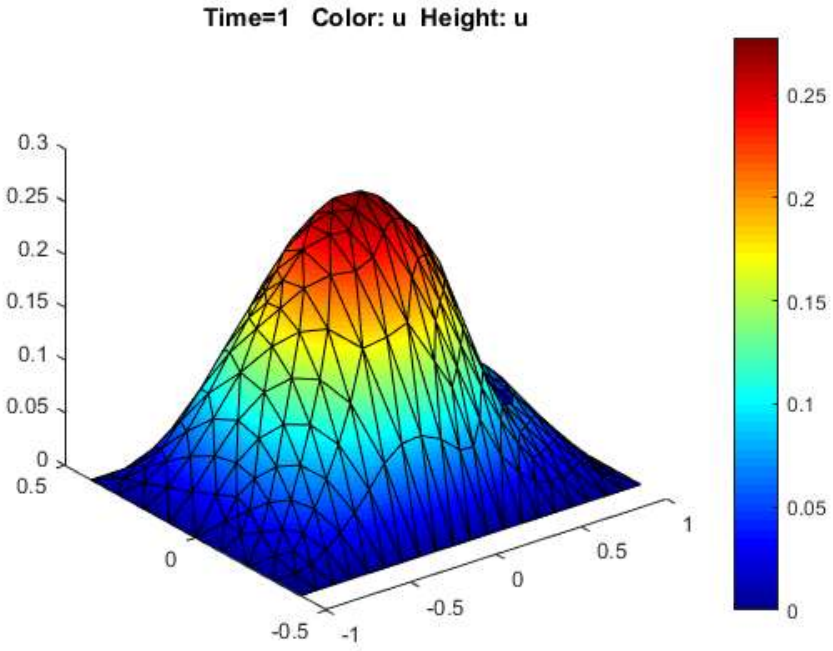


Kliknemo na trouglić, kao obi inicijalizovali mrežu, pa na tri trouglića kako bi profinili mrežu ili odemo na opciju

Mesh – Jiggle Mesh kako bi popravili kvalitet mreže
Podesimo vreme opcijom
Solve – Parametres i podesimo
Time = linspace(0,1,50)
u(to)=0
Rešimo problem naredbom = .
Dobićemo sledeću sliku



Drugačiji prikaz rešenja možemo da dobijemo ako kliknemo na Plot – Plot Parametres i čekiramo opciju Heigh (3D plot), Show mesh, izaberemo za Colormap „jet“ i kliknemo na Plot dugme.



Da rezimiramo:

PDJ problem može se rešiti i na sledeći način:

Kreiramo Pde problem naredbom

```
>> model = createpde;
```

Ili

```
>> model = createpde(N);  % ukoliko imamo N jednačina u sistemu
```

Učitamo geometriju sa kojom želimo da radio

```
>>importGeometry(model, 'geometry.stl');  % ako želimo da učitamo geometriju koja je 3D
```

ili

Da bi podesili granične uslove potrebno je da znamo nazive granica:

```
>> pdegplot(model,'FaceLabels','on') % 'FaceLabels' za 3-D
>> pdegplot(model,'EdgeLabels','on') % 'EdgeLabels' za 2-D
```

Kreiramo granične uslove:

```
>>applyBoundaryCondition(model,'face',[2,3,5],'u',[0,0]);
```

% za 3D oblike kreiramo Dirihleove uslove na oblastima 2,3,5 i postavljamo vrednosti parametara na 0

```
>>applyBoundaryCondition(model,'edge',[1,4],'g',1,'q',eye(2));
```

% za 2D oblike kreiramo Nojmanove uslove na granicama 1 i 4, tako što parametrima g i q dodeljujemo vrednosti koje odgovaraju jediničnoj matrici.

Dodeljujemo vrednosti koeficijentima PDJ:

```
>>f = [1;2];
>>a = 0;
>>c = [1;3;5];
>> specifyCoefficients(model,'m',0,'d',0,'c',c,'a',a,'f',f);
```

Generišemo mrežu

```
>>generateMesh(model);
```

Pozovemo odgovarajući solver:

```
>>u = assempde(model, c,a,f);
```

- Za PDJ eliptičkog tipa kod kojih koeficijenti ne zavise od u se koristi assempde
- Inače se koristi pdenonlinear
- Za PDJ paraboličkog tipa se koristi parabolic
- Za PDJ hiperboličkog tipa se koristi hyperbolic
- Za rešavanje sospstvenih vrednosti pdeeig

Repšenje zapamtimo kao vektor

```
>> result = solvepde(model);  % za statičke probleme
>> result = solvepde(model,tlist); % za dinamičke probleme
>> result = solvepdeeig(model); % za sopstvene vrednosti
```

Rešavanje PDJ u 3D (sa časa)

```
% PRVO KREIRAMO MODEL NAD KOJIM CEMO RADITI KAO PDJ (PARCIJALNU
% DIFERENCIJALNU JEDNACINU)
model = createpde();

% DEFINISEMO OBLIK MODELA NAD KOJIM CEMO RADITI (U OVOM SLUCAJU ZA MODEL
% UZIMAMO DRZAC SA RUPOM)
importGeometry(model, 'BracketWithHole.stl');
      % neke druge geometrije
      %     importGeometry(model,'Torus.stl');
      %      importGeometry(model, 'Block.stl');
      %     importGeometry(model, 'Plate10x10x1.stl');
      %     importGeometry(model, 'Tetrahedron.stl');
      %     importGeometry(model, 'BracketTwoHoles.stl');
      %     importGeometry(model, 'ForearmLink.stl');



% DA BI CRTALI NA RAZLICITIM GRAFICIMA POSTAVIMO FIGURU 1
% GRAFIK CRTAMO POZIVANJEM NAREDBE pdegplot
% OZNAKE FaceLabels I on NAM OMOGUCAVAJU DA SVAKA STRANICA NASE FIGURE IMA
% NAZIV

figure(1)
pdegplot(model, 'FaceLabels' ,'on');
title('Drzac sa rupom');

% POSTAVLJAMO KOEFICIJENTE PDE (KOJI MOGU BITI SKALARI, MATRICE, NIZOVI...
c = 1;    % na primer:
          % >> c = char('x..^2+y.^2', 'cos(x.*y)', 'u./(1+x.^2+y.^2');
          % formira c kao simetricnu matricu koja na u prvom redu ima
          % prva dva argumenta dok u drugom redu ima drugi i treci argument
a = 0;
         % a i d najcesce imaju dijagonalnu formu ('a' je na glavnoj
         % dijagonali i svi ostali elementi su jednaki nuli, dok se 'd'
         % zadaje kao vektor cije su vrednosti na glavnoj dijagonali dok su
         % svi ostali elementi jednaki nuli)

         % na primer: Ako imamo da je N=3
         % >> a = char('sin(x)+cos(y)', 'cosh(x.*y)',
         % 'x.*y./(1+x.^2+y.^2)');
f = 0.5;
         % Funkcija f moze biti zadata i na sledeci nacin
         % function f = fcoeff(x,y,t,sd)
            %f = (x.*y)./(1 + x.^2 + y.^2); % f za subdomen 1
            %f = f + log(1 + t); % ukljucuje vreme r = (sd == 2); % subdomen2
            %f2 = cos(x + y); % koeficijenti na subdomenu2;
            %f(r) = f2(r);      % f na subdomen 2

            % Paraboličkа funkcija ovakvog tipa koristi paraboličkи solver
            %   u1 = parabolic(u0,tlist,b,p,e,t,c,a,'fcoeff(x,y,t,sd)',d)

            % Ako se koristi PDE modelar, tada se u polju uz f samo upisuje
      "foceff)x,z,t,sd" (bez "=" ili bilo kojih drugih karaktera)

d = 1;
        % kod pde eliptickog tipa, ukoliko koeficijenti c,a,f i d sadrze
        % 'u', 'ux', 'uy' ili 'uz' umesto solvera ASSEMPDE se koristi
        % solver PDENONLIN.
```

```
% b - granicni uslovi (Boundary condition) se zadaju kao funkcija ili ime
% fajla
% na primer, mozemo da napisemo
% b = 'circleb1'
% ili
% b = @circleb1;



% GENERISEMO MREZU
generateMesh(model);

% DODELJUJEMO POCETNE I GRANICNE USLOVE
% - GRANICNI USLOV NA ZADNJOJ STRANI FIGURE SU DIRICHLE-OVI USLOVI SA VREDNOSCU 0;
% - hu =r
applyBoundaryCondition(model, 'Face', 4,'u',0);  % koristi se za 3D figure

% Neka se, na primer dodeljuje vrednost u=2 delovima figure koji su oznaceni sa e1, e2 i
e3
% tada za 3D figure pisemo
% applyBoundaryCondition(model,'dirichlet','Face',[e1,e2,e3],'u',2);
% a za 2D figure
% applyBoundaryCondition(model,'dirichlet','Edge',[e1,e2,e3],'u',2);

% ukoliko nije konkretno naznacena vrednost parametara 'r' i 'h'  podrazumeva se da
% je r=0 i da je h=1;
% na primer, uslovi su 2u=3
% 3D
% applyBoundaryCondition(model,'dirichlet','Face',[e1,e2,e3],'r',3,'h',2);
% 2D
% applyBoundaryCondition(model,'dirichlet','Edge',[e1,e2,e3],'r',3,'h',2);

% recimo da na e1 u=1, dok je na e2 i e3 u=2, tada se zapisuje
% 3D
%
applyBoundaryCondition(model,'dirichlet','Face',[e1,e2,e3],'u',[1,2,2],'EquationIndex',[1
,2,3]);
% 2D
%
applyBoundaryCondition(model,'dirichlet','Edge',[e1,e2,e3],'u',[1,2,2],'EquationIndex',[1
,2,3]);

% u prethodnom slucaju 'u' i 'EquationIndex' moraju biti istih duzina
% ukoliko se ne zapise 'EquationIndex' podrazumeva se da je 'u' duzine N
% ako se ne zada applyBoundaryCondition podrazumeva se da su granicni
% uslovi jednaki nuli.



% - SVE OSTALE STRANE IMAJU ZADANE GRANICNE USLOVE.
% - POCETNE VREDNOSTI SU u(0)=0 I du/dt(0)=x/2

u0 = 0;    % --- pocetni uslov.
           % --- moze da se zada kao skalar
           % >> u0 = 5;
           % kao vektor konstantnih vrednosti(npr. imamo sistem od 3 jednacine)
           % >> u0 = char('3', '-3', '0');
           % ili kao izraz  u(x,y) = x*y*cos(x)/(1+x^2+y^2);
           % >> u0 = 'x.*y.*cos(x)./(1+x.^2+y.^2)';
```

```
          % odnosno ako je u pitanju sistem
          % >> uo = char('x.^2+5*cos(x.*y)', 'tanh(x.*y)./(1+z.^2)');
          % cvorovi su ili 'p' iz strukture [p,e,t] ili iz model.Mesh.Nodes.

%  -- DA BI POSTAVILI POCETNE USLOVE ZA IZVOD, SVIM CVOROVIMA KOJI SE
%  NALAZE U PRVOM REDU MATRICE DODELICEMO VREDNOSTI X/2;
%  --- POCETNI USLOVI ZA IZVOD SE MOGU ZADATI SLICNO KAO I ZA U

cvorovi = model.Mesh.Nodes(1,:);
ut0 = cvorovi(:)/2;

% KREIRAMO ODGOVARAJUCE MATRICE KONACNIH ELEMENATA
% Kc - matrica krutosti (definise se kao retka ili puna matrica, predstavlja rezultat
assempde)
% Fc - vektor
% B - Dirichlet-ovo nullspace
% ud - Dirichlet vector

[Kc, Fc, B, ud] = assempde(model, c,a,f);
[~,M,~] = assema(model, 0,d,f);

% ---------------------------------
% SMANJIVANJE OSCILACIJA
UmanjivanjeOscilacija = 0.1*M;  % INTEZITET OSCILACIJA SE SMANJUJE 10% OD ITERACIJE DO
ITERACIJE

% RESENJE PO VREMENU JE OBICNO VEKTOR REALNIH VREDNOSTI
tlist = linspace(0,5,50);

u = hyperbolic(u0, ut0, tlist,Kc, Fc, B, ud, M, 'DampingMatrix', UmanjivanjeOscilacija);

% CRTAMO MAKSIMALNO VREME U SVAKOM MOMENTU.
% OSCILACIJE SE SMANJUJU KAKO VREME PROLAZI
figure(3)
plot(max(u));
xlabel('Vreme');
ylabel('Maksimalna vrednost');
title('Maximalno resenje');

% ---------------------------------


% RESAVAMO PDJ ZA VREME OD 0 DO 2;

tlist = linspace(0,5,50);
u = hyperbolic(u0, ut0, tlist,Kc, Fc, B, ud, M);


% PRIKAZUJEMO RESENJE U PRVIH PAR VREMENSKIH INTERVALA
% SKALIRAMO SVE CRTEZE KAKO BI IMALI ISTI RASPON BOJA NAREDBOM caxis

umax = max(max(u));
umin = min(min(u));


figure(2)
subplot(2,3,1); % postavljamo lokaciju crteza
pdeplot3D(model, 'ColorMapData', u(:,5));
caxis([umin umax])
title('Vreme 1/2');
```

```
subplot(2,3,2); % postavljamo lokaciju crteza
pdeplot3D(model, 'ColorMapData', u(:,10));
caxis([umin umax])
title('Vreme 1');


subplot(2,3,3); % postavljamo lokaciju crteza
pdeplot3D(model, 'ColorMapData', u(:,15));
caxis([umin umax])
title('Vreme 3/2');


subplot(2,3,4); % postavljamo lokaciju crteza
pdeplot3D(model, 'ColorMapData', u(:,20));
caxis([umin umax])
title('Vreme 2');


subplot(2,3,5); % postavljamo lokaciju crteza
pdeplot3D(model, 'ColorMapData', u(:,25));
caxis([umin umax])
title('Vreme 5/2');

subplot(2,3,6); % postavljamo lokaciju crteza
pdeplot3D(model, 'ColorMapData', u(:,30));
caxis([umin umax])
title('Vreme 3');
```

Primer 4
Resavanje sledeći problem

$$\frac{\partial^2 u}{\partial t^2} = \Delta u$$

na geometriji squareg
Sa Dirihleovim graničnim uslovima $u = 0$ kada je $x = \pm 1$ i Nojmanovim uslvima $\nabla u\, n = 0$ kada je $y = \pm 1$.
Početnim uslovima

$$u_0 = \operatorname{atan}\left(\cos\left(\frac{pi}{2}x\right)\right)$$
$$u_{to} = 3\sin(pi * x) .* \exp(\cos(pi * y))$$

```
model = createpde;
geometryFromEdges(model, @squareg);
pdegplot(model, 'EdgeLabels', 'on');
ylim([-1.2 1.2]);
axis equal


applyBoundaryCondition(model, 'dirichlet', 'Edge', [2,4],'u',0);
applyBoundaryCondition(model, 'neumann', 'Edge', [1,3], 'g',0);

% pocetne vrednosti
u0 = 'atan(cos(pi/2*x))';
ut0 = '3*sin(pi*x).*exp(cos(pi*y))';

% resenje po vremenu
tlist = linspace(0,5,31);

% koeficijenti
c = 1;
a = 0;
f = 0;
d = 1;


generateMesh(model, 'GeometricOrder', 'linear', 'Hmax', 0.1);
u1 = hyperbolic(u0, ut0, tlist, model, c,a,f,d);


figure(2)
pdeplot(model,'XYData', u1(:,1));


figure(3)
pdeplot(model,'XYData', u1(:,end));
```