DICACIM programme A.Y. 2024–25
Numerical Methods for Partial Differential Equations

MATLAB pdeModeler

Paola Gervasio

DICATAM, Università degli Studi di Brescia (Italy)

uniBS, April-May, 2025

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

PDE Modeler – Remarks

- it is useful to make the first tests and if the geometry is simple, it is unpractical if you need to do many tests and the geometry is complex,
- geometry, mesh, boundary conditions, data, solution are not visible in the MATLAB workspace. If you want to save/modify them you need to export them to the MATLAB Workspace:
  - Boundary > Export ... to export geometry (g) and boundary conditions (b),
  - PDE > Export ... to export coefficients (c, a, f, d),
  - Mesh > Export Mesh to export the mesh (p, e , t): p = nodes coordinates, e = edges matrix, t= connectivity matrix (the last row contains the number of the subdomain), https://it.mathworks.com/help/pde/ug/mesh-data-pet-triples.html
  - Solve > Export Solution to export the solution (u).

You can define alternative names for the variables, inside the popup windows that open during the saving step.

PDEModel object

Open a new script:

```
clear % clear the workspace

% create a PDEModel object with name ''model''
model = createpde;
% model is a structure with 8 properties (or attribute
%          PDESystemSize: 1
%       IsTimeDependent: 0
%              Geometry: []
%  EquationCoefficients: []
%    BoundaryConditions: []
%     InitialConditions: []
%                  Mesh: []
%         SolverOptions: [1x1 pde.PDESolverOptions]
```

UNIVERSITÀ DEGLI STUDI DI BRESCIA

PDEModel object: Geometry

```
% define the  square (0,1)x(0,1)
% 3 = Rectangle, 4 = number of edges,
% all the x-coordinates and all the y-coordinates
Q1 = [3,4, 0,1,1,0, 0,0,1,1]';  % column vector
% 1 = Circle, center (.4,.3), radius .2
C1 = [1,0.4,0.3,0.2]';
% Pad C1 with zeros to enable concatenation with Q1
C1 = [C1; zeros(length(Q1)-length(C1),1)];
% gd = geometry description matrix
gd=[Q1,C1];
% ns = name-space matrix
ns = (char('Q1','C1'))';
% sf = set formula
sf = 'Q1-C1';
% Create the geometry
g = decsg(gd,sf,ns);
% import the geometry inside the model
geometryFromEdges(model,g);
```

PDEModel object: PDE and coefficients

The general equation reads:

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) + au = f$$

```
specifyCoefficients(model,...
  'm',0,'d',0,'c',1,'a',0,'f',1);
%  model = object
%  name of the coefficient followed by:
%  value (if costant)
%  array (if the coefficient is variable and
%      you know it point by point)
%  function handle (if the coefficient is variabile
%      and you know the formula)
```

PDEModel object: Boundary conditions

For the elliptic problem:

$$\begin{aligned} \textit{Dirichlet}: &\quad hu = r && \text{on } \partial\Omega_D \\ \textit{Neumann}: &\quad \mathbf{n}\cdot(c\nabla u) + qu = g && \text{on } \partial\Omega_N \end{aligned}$$

Draw the geometry and set the number of edges to impose boundary conditions

```
pdegplot(model,'EdgeLabels','on')
```

set the boundary conditions:

```
% u=1 on the edges 1 and 2
applyBoundaryCondition(model,...
  'dirichlet','Edge',1:2,'h',1,'r',1);
% u=0 on the edges 5, 6, 7, 8
applyBoundaryCondition(model,...
  'dirichlet','Edge',5:8,'h',1,'r',0);
% homogeneous Neumann on the edges 3 and 4
applyBoundaryCondition(model,...
  'neumann','Edge',3:4,'q',0,'g',0);
```

PDEModel object: Mesh generation

To generate the $\mathbb{P}_1$ mesh with $h = 0.1$:

```
generateMesh(model,'Hmax',0.1,...
    'GeometricOrder','linear');
```

To generate the $\mathbb{P}_2$ mesh with $h = 0.1$:

```
generateMesh(model,'Hmax',0.1,...
    'GeometricOrder','quadratic');
```

To show the mesh (once you have generated it)

```
pdeplot(model)
```

To export the mesh with the format `p,e,t`

```
[p,e,t]=meshToPet(model.Mesh);
```

PDEModel object: Solve and plot

Solving:

```
results = solvepde(model);
```

Plot:

```
figure(1); clf
% extract u from results
u = results.NodalSolution;
% 3d-plot of u with mesh
pdeplot(model,'XYData',u,'ZData',u,'Mesh','on')

figure(2); clf
% plot u,  the contourlines, the gradient
[gradx,grady] = evaluateGradient(results);
 pdeplot(model,'XYData',u,'Contour','on',...
      'FlowData',[gradx,grady])
```

How to define elementary geometries

Define column arrays with the following data:

Circle:

Row	Value
1	1 (indicates a circle)
2	x-coordinate of circle center
3	y-coordinate of circle center
4	Radius (strictly positive)

Polygon:

Row	Value
1	2 (indicates a polygon)
2	Number of line segments n
3 through 3+n-1	x-coordinate of edge starting points
3+n through 2*n+2	y-coordinate of edge starting points

The polygon must be closed and not intertwined

How to define elementary geometries (2)

Rectangle:

Row	Value
1	3 (indicates a rectangle)
2	4 (number of line segments)
3 through 6	x-coordinate of edge starting points
7 through 10	y-coordinate of edge starting points

Ellipse:

Row	Value
1	4 (indicates an ellipse)
2	x-coordinate of ellipse center
3	y-coordinate of ellipse center
4	First semiaxis length (strictly positive)
5	Second semiaxis length (strictly positive)
6	Angle in radians from x axis to first semiaxis

Set Formula

The final geometry is the result of operations written in a string:
$+$ union,
$*$ intersection,
$-$ difference,
() to amodify the precedence of operations.

$+$ and $*$ have the same precedence, $-$ has higher precedence.

Example:

```
sf = '(R1+C1)-C2';
```

Command **pdegplot**.
You can also define regions of the plane inside parametric curves.

```
>> help pdegplot
```

UNIVERSITÀ DEGLI STUDI DI BRESCIA

Problem 2

Solve the following problem using $\mathbb{P}_1$ and $\mathbb{P}_2$:

$$\begin{cases} -\mu\Delta u + \sigma u = f & \text{in } \Omega \\ u = g_D & \text{on } \partial\Omega \end{cases}$$

where Ω is a polygon whose vertices are (0,0), (1,0), (1.5,1), (0.7, 1.5), (-0.5,0.8). The coefficients are $\mu = 0.1$, $\sigma = 1$, $f = 0.8e^{x+y}$, $g_D = e^{x+y}$.
The exact solution is $u(x, y) = e^{x+y}$, verify that the errors vanish as stated by the theory.

Use the function

`FEM_2d/fem_2d_errors.m`

to compute the errors.

Problem 2: define the geometry

Ω is a polygon whose vertices are (0,0), (1,0), (1.5,1), (0.7, 1.5), (-0.5,0.8)

```
% 2=polygon; 5= number of edges; all x; all y
P1=[2,5,0,1,1.5,0.7,-0.5, 0 0 1 1.5 0.8]';
gd=[P1];
ns=(char('P1'))';
sf='P1';
g=decsg(gd,sf,ns);
% import the geometry into the model
geometryFromEdges(model,g)
% visualization of the geometry
pdegplot(model,'Edge','on')
```

PDEModel object: PDE and coefficients (2)

Example: if $f(x, y) = exp(x + y)$, then

```
specifyCoefficients(model,...
  'm',0,'d',0,'c',0.1,'a',1,'f',@f_problem2);
```

where

```
function [f]=f_problem2(location,state)
% location and state are unchangeable names
f=exp(location.x+location.y);
```

`location` is a structure with 4 attributes: `x, y, z` and `subdomain`
(You can define Ω as the union of subsets and specify different data on different subsets)

`state` is a structure with 5 attributes: `u, ux, uy, uz, time`

UNIVERSITÀ DEGLI STUDI DI BRESCIA

PDEModel object: PDE and coefficients (3)

Example: f depends on the unknown function u, too:

```
function [f]=fcoeff(location,state)
% location and state are unchangeable names
f=exp(location.x+location.y)+state.u;
```


UNIVERSITÀ DEGLI STUDI DI BRESCIA

Paola Gervasio - DICATAM - unibs.it

Problem 2: boundary conditions

$u = e^{x+y}$ on $\partial\Omega$.

```
applyBoundaryCondition(model,...
  'dirichlet','Edge',1:5,'h',1,'r',@gd_problem2);
```

where

```
function [f]=gd_problem2(location,state)
% g_d(x,y)=exp(x+y)
f=exp(location.x+location.y);
```

Problem 2: mesh, solve and plot

Choose either $\mathbb{P}_1$ or $\mathbb{P}_2$ and define the mesh

```
p=....; h=0.01;
if p==1
  generateMesh(model,'Hmax',h,...
               'GeometricOrder','linear');
elseif p==2
  generateMesh(model,'Hmax',h,...
               'GeometricOrder','quadratic');
end
% plot the mesh
figure(1); clf
pdeplot(model);
% solve
results=solvepde(model);
% plot the solution
figure(2); clf
u=results.NodalSolution;
pdeplot(model,'XYData',u,'Zdata',u,'Mesh','off')
```

Problem 2: compute the errors

```
uex=@(x,y)exp(x+y); % exact solution
uexx=@(x,y)exp(x+y); % du/dx
uexy=@(x,y)exp(x+y); % du/dy

...

[errors]=fem_2d_errors(u,model,uex,uexx,uexy);
H=[H;h];
Err_l2=[Err_l2;errors.l2];
Err_h1=[Err_h1;errors.h1];
```

Problem 3

Solve the following equation with FEM-$\mathbb{P}_1$

$$\begin{cases} -\nabla \cdot (K \nabla u) = 0 & \text{in } \Omega = (0,1)^2 \\ u = 1 & \text{on } (0,1) \times \{0\} \\ u = 0 & \text{on } (0,1) \times \{1\} \\ (K \nabla u) \cdot \mathbf{n} = 0 & \text{on } (\{0\} \cup \{1\}) \times (0,1) \end{cases}$$

where

$$K(x,y) = \begin{cases} 11 & \text{if } (x-0.5)^2 + (y-0.5)^2 < (0.2)^2 \\ 1 & \text{otherwise} \end{cases}$$

is the hydraulic conductivity of a porous medium that occupies the domain Ω, while u is the hydraulic head of the fluid filtering inside the porous medium.

Problem 3: Geometry

To define the function K we have to use 2 subdomains, each of them is called 'Face'.

```
% geometry
Q1 = [3,4,0,1,1,0,0,0,1,1]';
C1 = [1,0.5,0.5,0.2]';
C1 = [C1; zeros(length(Q1)-length(C1),1)];
gd=[Q1,C1];
% Names for the two geometric objects
ns = (char('Q1','C1'))';
% Set formula
sf = 'Q1+C1';
% Create geometry
g = decsg(gd,sf,ns);
% import the geometry
geometryFromEdges(model,g);
% draw the geometru with Edges and Faces
pdegplot(model,'EdgeLabels','on','FaceLabels','o
```

Problem 3: Set the data

```
% set the coefficients of the pde: on Face 2
specifyCoefficients(model,'m',0,'d',0,...
    'c',11,'a',0,'f',0,'Face',2);
% on Face 1
specifyCoefficients(model,'m',0,'d',0,...
    'c',1,'a',0,'f',0,'Face',1);

% apply boundary conditions: u=1 on Edge 4
applyBoundaryCondition(model,...
   'dirichlet','Edge',4,'u',1);
% u=0 on Edge 2
applyBoundaryCondition(model,...
   'dirichlet','Edge',2,'u',0);
% du/dn=0 on Edges 1 and 3
applyBoundaryCondition(model,...
   'neumann','Edge',[1,3],'q',0,'g',0);
```

UNIVERSITÀ DEGLI STUDI DI BRESCIA

Matrices and arrays of the linear system

We can generate a new structure that contains the blocks of the linear system:

```
FEM = assembleFEMatrices(model,'nullspace');
```

FEM as the following attributes:

```
Kc:    % matrix A0, (i,j) non-dirichlet rows and colu
Fc:    % rhs f0, (i) non-dirichlet rows
B:     % to remap u0 in u
ud:    % ud = discrete lifting  Rgd^h
M:     % mass matrix, only for time-dependent problem
```

It holds: $K_c \mathbf{u}^0 = F_c$ and $\mathbf{u} = B\mathbf{u}^0 + \mathbf{u}_d$.
Algebric operation: `u=B*(Kc\Fc)+ud`

To view the pattern of the matrix Kc:

```
figure; spy(FEM.Kc)
```

UNIVERSITÀ DEGLI STUDI DI BRESCIA

References

MATLAB PDEToolbox: https://it.mathworks.com/help/pde/index.html?s_tid=CRUX_lftnav
PDF Documentation of PDEToolbox