# DICACIM programme A.Y. 2019–20
# Numerical Methods for Partial Differential Equations

## Lab 2
## Numerical solution of 2D elliptic problems with Finite Elements and MATLAB PDEtoolbox

Paola Gervasio

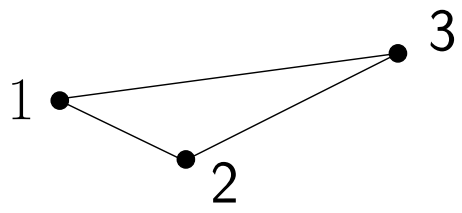
DICATAM, Università degli Studi di Brescia (Italy)

uniBS, May, 2020

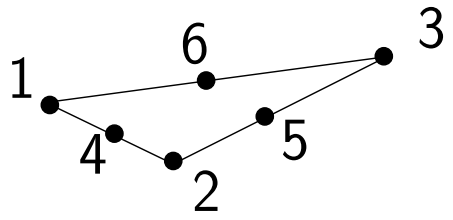
Il PDEToolbox di MATLAB implementa:

- triangoli in 2D e tetraedri in 3D,
- FEM-$\mathbb{P}_1$ (lineari) e FEM-$\mathbb{P}_2$ (quadratici).

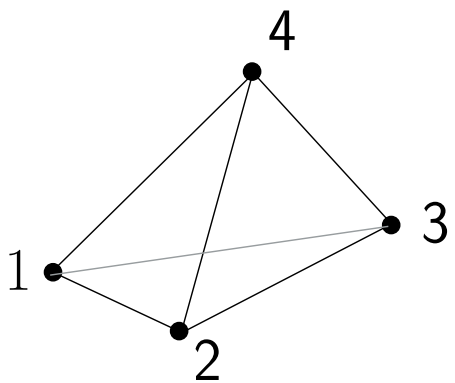
Ordinamento dei **gradi di libertà:**



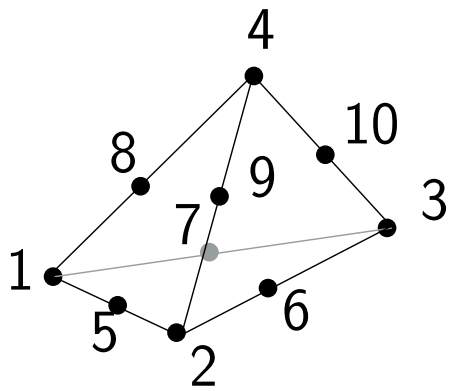

2D $\mathbb{P}_1$ (linear)




2D $\mathbb{P}_2$ (quadratic)




3D $\mathbb{P}_1$ (linear)




3D $\mathbb{P}_2$ (quadratic)

# Workflow

1. definire la **geometria** ($\Omega$),
2. selezionare il **problema** (il tipo di PDE, i coefficienti, termine sorgente),
3. definire le **condizioni al bordo**,
4. costruire la **mesh** ($\mathcal{T}_h$),
5. **risolvere** il sistema di equazioni (calcolare $u_h$),
6. rappresentare **graficamente** la soluzione,
7. analisi degli **errori** (se si ha una **soluzione test**).

Possiamo procedere in due modi diversi:

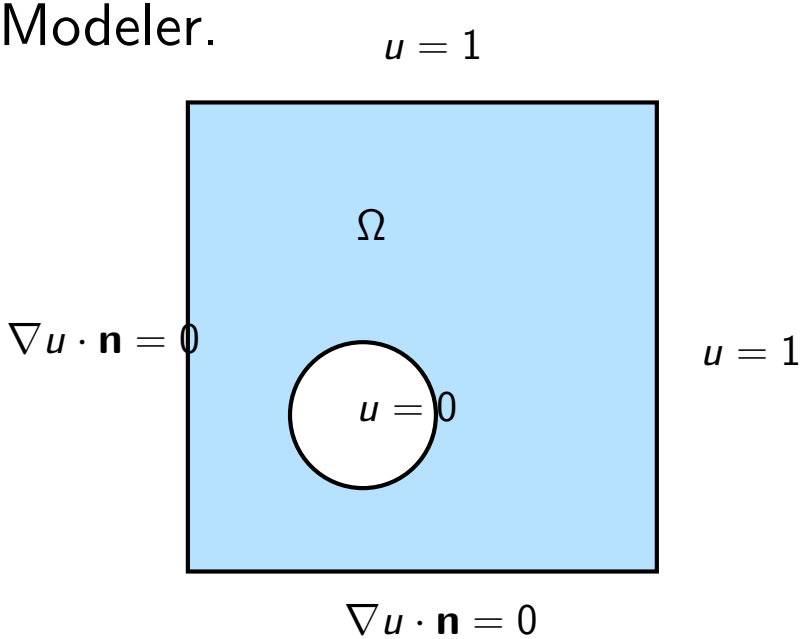
1. con **pdeModeler** (interfaccia grafica) solo $\mathbb{P}_1$ in 2D ,
2. con un **PDE model object** e dando istruzioni in script/function: $\mathbb{P}_1$ e $\mathbb{P}_2$, sia in 2D che 3D).

# Problema 1

Sia $P = (0.4, 0.3)$, e $C$ il cerchio di centro $P$ e raggio $r = 0.2$.
Approssimare la soluzione $u$ del problema di Poisson

$$\begin{cases} -\Delta u = 1 & \text{in } \Omega = (0,1)^2 \setminus C \\ u = 0 & \text{su } \partial C \\ u = 1 & \text{se } x = 1, \ o \ y = 1 \\ \nabla u \cdot \mathbf{n} = 0 & \text{se } x = 0, \ o \ y = 0 \end{cases}$$

con FEM-$\mathbb{P}_1$ usando pdeModeler.



# PDEModeler (interfaccia grafica) solo in 2D

Lanciare il comando

```
>> pdeModeler
```

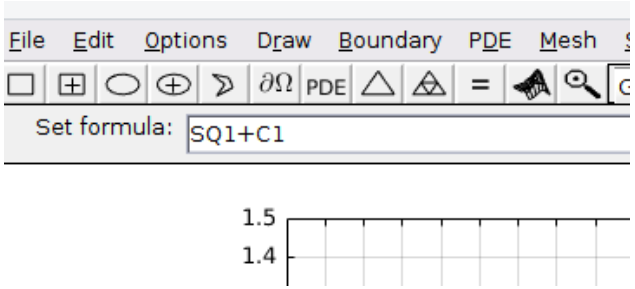
Inizializzazione del piano di lavoro

Dal menù (orizzontale alto):

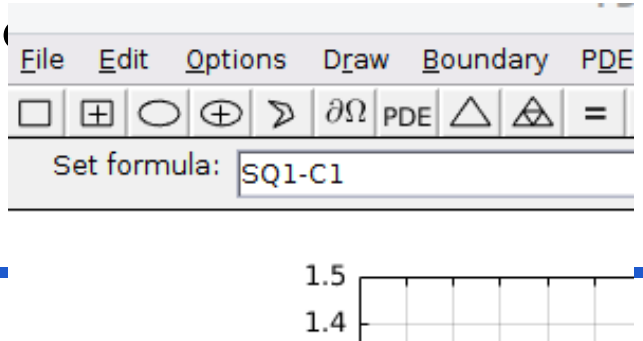
- Options > Axes Limits, per definire l'area 2D in cui disegnare il dominio,
- Options > Grid Spacing, per definire la spaziatura della griglia per il disegno,
- Options > Snap, per facilitare il disegno (i pixel cliccati sono attratti verso i punti della griglia disegnata)
- Options > Grid, per disegnare la griglia sulla finestra

# PDE Modeler – definizione della geometria

- La geomeria può essere definita come unione/intersezione/differenza di rettangoli, ellissi, poligoni, cerchi,
- cliccare sull'icona per disegnare col mouse la forma scelta,
- in automatico $\Omega$ sarà definita come l'unione di tutte le figure elementari disegnate (l'unione deve essere una regione connessa del piano) e in corrispondenza di *Set formula* si hanno "+"

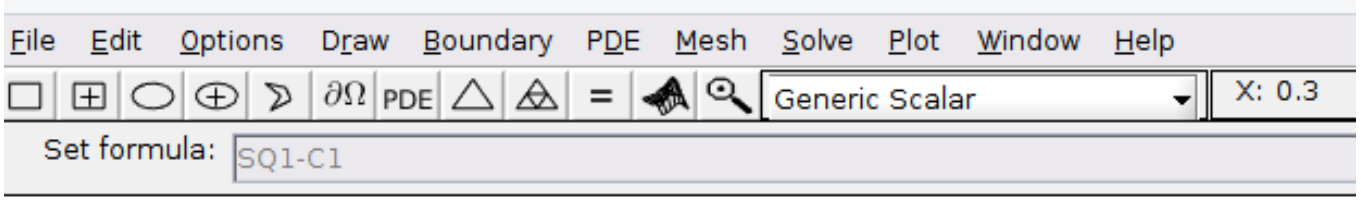



- se si vuole che $\Omega$ sia la differenza delle due figure: modificare la formula, sostituendo "+"



# PDE Modeler - scelta del problema

Dal menù a tendina posizionato a destra della lente selezionare il problema:




Con Generic Scalar si può risolvere il problema ellittico del second'ordine

$$-\nabla \cdot (c\nabla u) + au = f$$

dove $c$, $a$, $f$ possono essere costanti o funzioni di $(x, y)$.

Dal menù (orizzontale alto) selezionare PDE > PDE Specification, per definire $c$, $a$ e $f$.
Controllare che sia selezionato il tipo PDE **elliptic**.

# PDE Modeler – boundary conditions

Dal menù (orizzontale alto):
Boundary > Boundary Mode, per definire le boundary conditions.
Vengono visualizzati gli edge con il senso di percorrenza.
Ogni segmento è un edge.
La circonferenza (e l'ellisse in genere) sono suddivisi in 4 archi (corrispondenti ai 4 quadranti).

1. **Shift+click** su un edge per selezionarlo,
2. Boundary > Specify Boundary Conditions..
3. scegliere Neumann (in realtà è una condizione Robin) o Dirichlet
4. introdurre i coefficienti per
   - Dirichlet: $u = g_D$, settare $h = 1$ e $r = g_D$
   - Neumann: $c\frac{\partial u}{\partial \mathbf{n}} = c\nabla u \cdot \mathbf{n} = g_N$, settare $q = 0$ e $g = g_N$.
5. se su un edge non viene fatta alcuna scelta, allora viene imposto Dirichlet omogeneo,
6. gli edge Neumann sono colorati di blue, quelli Dirichlet di rosso.

# PDE Modeler – generazione della mesh

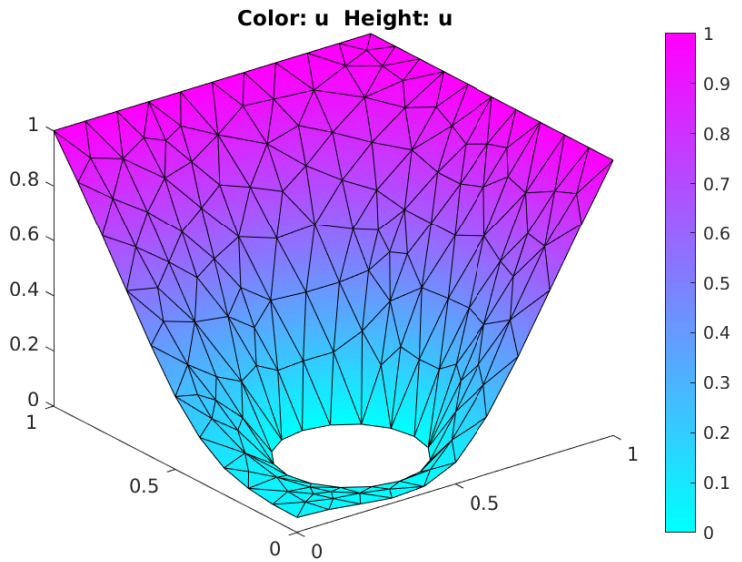
Dal menù (orizzontale alto):

1. Mesh > Mesh Mode, per costruire la mesh,
2. Mesh > Parameter...
3. definire "Maximum edge size" ($h$)
4. Mesh > Initialize Mesh
5. Mesh > Refine Mesh raffina la mesh dividendo ogni triangolo in 4 triangoli.

# PDE Modeler – risoluzione e plot

Dal menù (orizzontale alto):

1. Solve > Solve PDE per risolvere il problema,
2. Plot > Plot Solution per disegnare la soluzione in 2D
3. Plot > Parameters... se si vuole un disegno 3D, se si vuole disegnare la mesh, il gradiente, le contourlines.




Soluzione con $h = 0.1$.

# PDE Modeler – Osservazioni

- è comodo per fare le prime prove e se la geometria è semplice, scomodo se si devono fare molte prove e/o se la geometria è complessa,
- geometria, mesh, boundary conditions, dati, soluzione non sono visibili nel workspace di MATLAB, se li si vuole esportare nel MATLAB Workspace bisogna esportarli:
  - Boundary > Export ... per esportare geometria (g) e boundary conditions (b),
  - PDE > Export ... per esportare i coefficienti ($c$, $a$, $f$, $d$),
  - Mesh > Export Mesh per esportare la mesh (p, e , t): p = matrice delle coordinate dei nodi, e = matrice degli edge, t= matrice di connettività (ultima riga contiene il numero del sottodominio), https://it.mathworks.com/help/pde/ug/mesh-data-pet-triples.html
  - Solve > Export Solution per esportare la soluzione ($u$).

Nei popup che si aprono si possono definire i nomi delle variabili in cui salvare i dati (alternativi a quelli che propone MATLAB).

# PDEModel object

In un nuovo script:

```
clear % pulisce workspace

model = createpde; % crea l'oggetto di nome ''model''
% della classe PDEModel
% model e' una struttura  i cui campi conterranno:
% geometria, mesh, coefficienti, soluzione, ...
```

# PDEModel object: Geometria

```
% definisco il  quadrato (0,1)x(0,1)
% Rectangle is code 3, 4 sides,
% followed by x-coordinates and then y-coordinates
Q1 = [3,4, 0,1,1,0, 0,0,1,1]';  % vett. colonna
% Circle is code 1, 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=[R1,C1];
% ns = name-space matrix
ns = (char('Q1','C1'))';
% sf = set formula
sf = 'Q1-C1';
% Crea la geometria
g = decsg(gd,sf,ns);
% importa la geometria nel modello
geometryFromEdges(model,g);
```

# PDEModel object: PDE e coefficienti

L'equazione di riferimento è

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

```
specifyCoefficients(model,...
  'm',0,'d',0,'c',1,'a',0,'f',1);
%  model = oggetto
%  nome del coefficiente tra apici seguito da:
%  valore (se costante)
%  vettore (se e' noto nei punti della mesh)
%  function handle (se variabile)
```

# PDEModel object: PDE e coefficienti (2)

Ad esempio, se $f(x, y) = exp(x + y)$, allora

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

dove

```
function [f]=fcoeff(location,state)
% location e state  sono nomi obbligatori
f=exp(location.x+location.y);
```

`location` è una struttura con 4 campi: `.x`, `.y`, `.z` e `.subdomain`
(posso definire $\Omega$ come unione di sottodomini e specificare dati diversi per ogni sottodominio)

`state` è una struttura con 5 campi: `.u`, `.ux`, `.uy`, `.uz` e `.time`

# PDEModel object: PDE e coefficienti (3)

Altro esempio: $f$ dipende anche dalla funzione incognita $u$:

```
function [f]=fcoeff(location,state)
% location e state  sono nomi obbligatori
f=exp(location.x+location.y)+state.u;
```

# PDEModel object: Boundary conditions

Per il problema ellittico:

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

Disegno la geometria e numero gli edge per imporre le bc

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

Definisco le bc:

```
% u=1 sugli edge 1 e 2
applyBoundaryCondition(model,...
  'dirichlet','Edge',1:2,'h',1,'r',1);
% u=0 sugli edge 5, 6, 7, 8
applyBoundaryCondition(model,...
  'dirichlet','Edge',5:8,'h',1,'r',0);
% Neumann omogoeneo su edge 3 e 4
applyBoundaryCondition(model,...
  'neumann','Edge',3:4,'q',0,'g',0);
```

# PDEModel object: Generazione della mesh

Per generare la mesh per $\mathbb{P}_1$ con $h = 0.1$:

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

Per generare la mesh per $\mathbb{P}_2$ con $h = 0.1$:

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

Per visualizzare la mesh (una volta che è stata generata)

```
pdeplot(model)
```

Per esportare la mesh in formato `p,e,t`

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

# PDEModel object: Risoluzione e plot

Risoluzione:

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

Disegno:

```
figure(1); clf
% estraggo la u da results
u = results.NodalSolution;
% disegno u in 3d con la mesh
pdeplot(model,'XYData',u,'ZData',u,'Mesh','on')

figure(2); clf
% disegno u,  le contour, il gradiente
[gradx,grady] = evaluateGradient(results);
 pdeplot(model,'XYData',u,'Contour','on',...
      'FlowData',[gradx,grady])
```

# Geometrie base da dare in pasto a decsg

Generare vettori colonna con le seguenti informazioni:

**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 |

La poligonale deve essere chiusa e non intrecciata

# Geometrie base da dare in pasto a decsg (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

La geometria finale si ottiene scrivendo una stringa che descrive operazioni di unione e intersezione:
$+$ per l'unione,
$*$ per l'intersezione,
$-$ per la differenza,
() per raggruppare.

$+$ e $*$ hanno la stessa precedenza, $-$ ha precedenza più alta.

**Esempio:**

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

Comando **pdegplot**.
Si possono definire anche regioni di piano interne a curve parametriche.

```
>> help pdegplot
```

# Problema 2

Risolvere con FEM-$\mathbb{P}_1$ e FEM-$\mathbb{P}_2$ il problema

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

dove $\Omega$ è il poligono di vertici (0,0), (1,0), (1.5,1), (0.7, 1.5), (-0.5,0.8), e $\mu = 0.1$, $\sigma = 1$, $f = 0.8e^{x+y}$, $g_D = e^{x+y}$.
Sapendo che $u9x, y) = e^{x+y}$, verificare che l'errore decresce come predetto dalla teoria.
Utilizzare la function

`FEM_2d/fem_2d_errors.m`

per calcolare gli errori.

# Problema 3

Risolvere con FEM-$\mathbb{P}_1$ l'equazione

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

essendo

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

la conducibilità idraulica di un mezzo poroso che occupa la regione $\Omega$ e $u$ la quota piezometrica del fluido che filtra nel mezzo poroso.

# Problema 3: Geometria

Per definire il coefficiente $K$ bisogna utilizzare i sottodomini, chiamati '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);
% disegno  la geometria con Edge e Face
pdegplot(model,'EdgeLabels','on','FaceLabels','on')
```

# Problema 3: Definizione dei dati

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

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

# Matrici e vettori del sistema lineare

L'istruzione

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

memorizza in `FEM` le matrici:

```
Kc:    % matrice A0, (i,j) non-dirichlet
Fc:    % t. noto f0, (i) non-dirichlet
B:     % per rimappare u0 in u
ud:    % ud = lifting discreto Rgd^h
M:     % massa, solo per p. tempo-dip
```

Si ha che $K_c\mathbf{u}^0 = F_c$ e $\mathbf{u} = B\mathbf{u}^0 + \mathbf{u}_d$.
Algebricamente: `u=B*(Kc\Fc)+ud`

Per visualizzare il pattern di `Kc`:

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

# Problema 5

Calcolare la soluzione FEM-P1 e FEM-P2 di

$$\begin{cases} -\mu\Delta u + \sigma u = 1 & \text{in } \Omega = (0,1)^2 \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

con $\mu = 10^{-4}$ e $\sigma = 1$, verificando che anche per $d = 2$ la soluzione numerica mostra oscillazioni se $h$ non è sufficientemente piccolo.

# Problema 6

$$\begin{cases} -\mu\Delta u + \sigma u = 0 & \text{in } \Omega = (0,100)\times(0,20)\times(0,50) \\ u = 1 & \text{sulle facce con } x \text{ o } z \text{ costanti} \\ \mu\nabla u \cdot \mathbf{n} = 1 & \text{sulle facce con } y \text{ costante} \end{cases}$$

$\mu = 10$, $\sigma = 1$.

# Problema 6

```
model = createpde;
% importa la geometria da file STL
importGeometry(model,'Block.stl');
pdegplot(model,'FaceLabels','on')
% set the coefficients of the pde
specifyCoefficients(model,'m',0,'d',0,...
        'c',10,'a',1,'f',0);
% boundary conditions
applyBoundaryCondition(model,...
  'dirichlet','Face',1:4,'u',1);
applyBoundaryCondition(model,...
  'neumann','Face',5:6,'q',0,'g',1);
generateMesh(model); % generate the mesh
results = solvepde(model); % solve
u = results.NodalSolution;
pdeplot3D(model,'ColorMapData',u) % plot
```

# Riferimenti

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