ISSM Workshop 2014

Ice Sheet System Model

Mesh Generation/Adaptation

Mathieu MORLIGHEM[1]

[1] University of California, Irvine

Outline

Mesh generation in ISSM

- Mesh generation is critical for (ice sheet) modeling

→ controls the space of solutions

- Finer mesh more precise but more computationally intensive
- ISSM has 4 main meshers:
  1. squaremesh (for ISMIP tests)
  2. roundmesh (for EISMINT tests)
  3. triangle (from J. Shewchuk)
  4. bamg (adapted from F. Hecht)

Usage

`squaremesh` generates structured uniform meshes for rectangular domain
→ needed for ISMIP tests

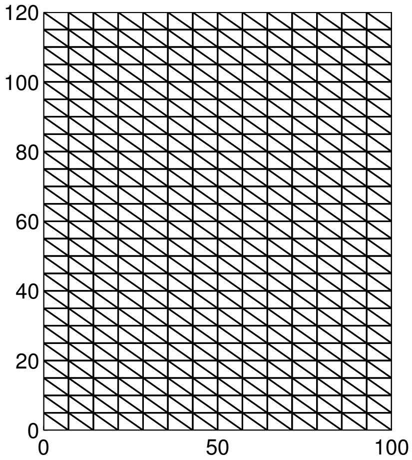
```
md=squaremesh(model,100,200,15,25);
```

Arguments:

1. model
2. x-length
3. y-length
4. number of nodes along the x axis
5. number of nodes along the y axis

Example

```
md=squaremesh(model,100,200,15,25);
```



Usage

`roundmess` generates unstructured uniform meshes for circular domain

→ needed for EISMINT tests

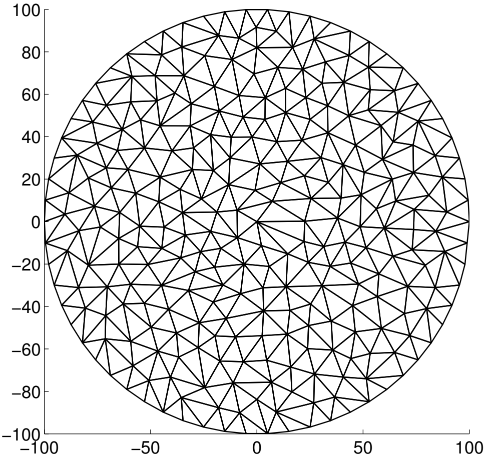
```
md=roundmesh(model,100,10);
```

Arguments:

1. model
2. radius
3. element size

Example

```
md=roundmesh(model,100,10);
```



Usage

`triangle` is a very fast algorithm for mesh generation

- + excellent for uniform mesh
- - bad at mesh refinement

```
md=triangle(model,'Square.exp',.2);
```

Arguments:

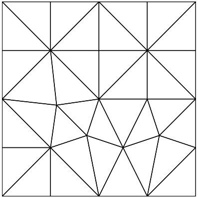
1. model
2. ARGUS file of the domain outline

```
## Name:domainoutline
## Icon:0
# Points Count  Value
5 1.
# X pos Y pos
0 0
1 0
1 1
0 1
0 0
```

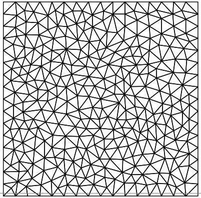
3. average element size

Example

```
md=triangle(model,'Square.exp',.2);
```



```
md=triangle(model,'Square.exp',.05);
```



History

Initial software:

- BAMG: Bidimensional Anisotropic Mesh Generator
- developed by Frédéric Hecht (INRIA/université de Jussieux)
- released in 2006 after more than 10 years of development
- now part of FreeFEM++

In ISSM:

- entirely rewritten
- usual ISSM interface

Advantages:

+ anisotropic mesh adaptation capability
- not good for uniform meshes

Usage

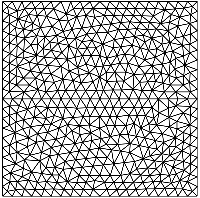
Arguments:

1. model
2. pair of options (see help)

To create a uniform mesh:

1. 'domain' followed by the domain name
2. 'hmax' followed by the triangle size

```
md=bamg(model,'domain','Square.exp','hmax',.05);
```



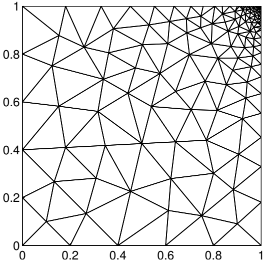
- Not as randomly distributed as `triangle`

Usage

To create a non-uniform mesh:

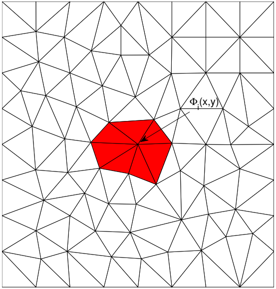
1. `'domain'` followed by the domain name
2. `'hvertices'` followed by the element size for each vertex of the domain outline

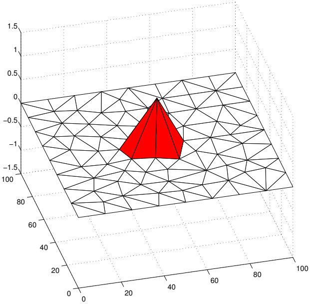
```
hvertices=[0.2;0.2;0.005;0.2];
md=bamg(model,'domain','Square.exp','hVertices',hvertices);
```



Finite element method

$$v = \sum_{i=1}^{N} v_i \, \Phi_i \left(x, y\right) \tag{1}$$

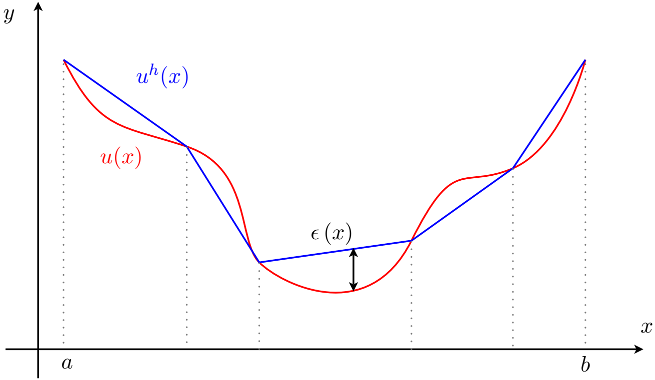




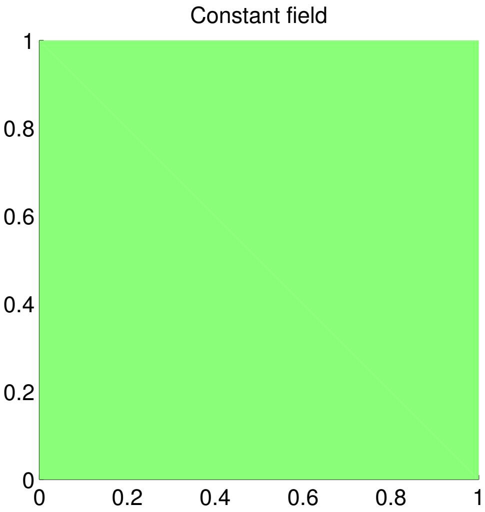
Interpolation error

- We generally use piecewise linear elements ($P1$)

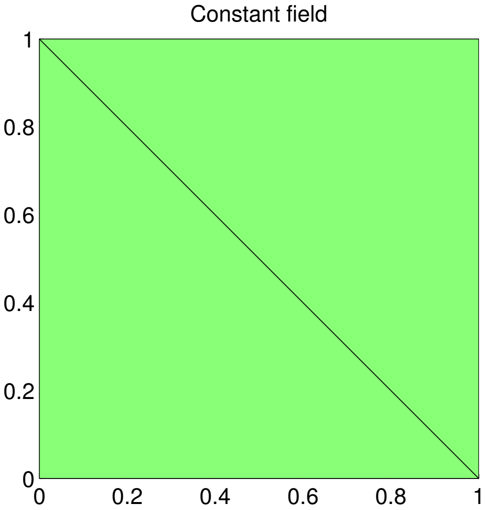
→ How to minimize interpolation error and the number of elements at the same time?



Costant field

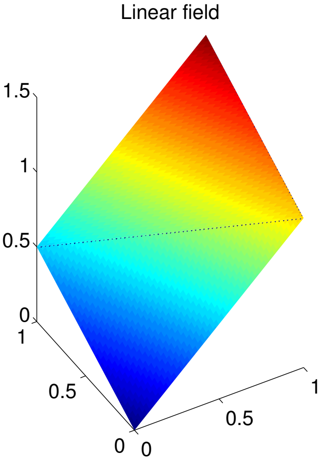


Costant field

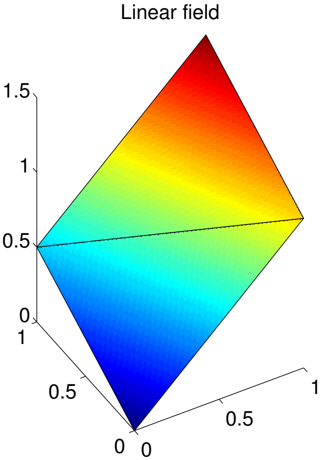


- Coarse elements OK for constant field

Linear field

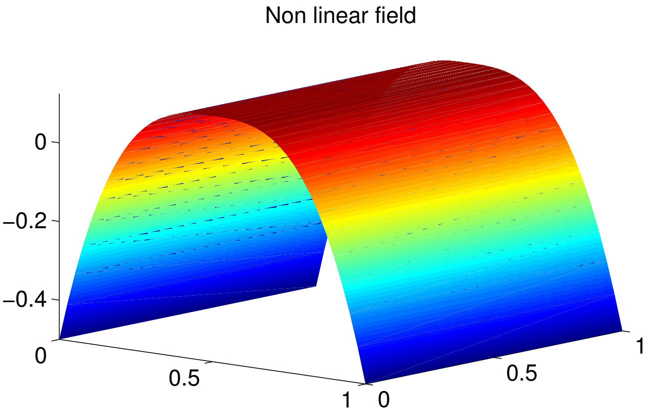


Linear field

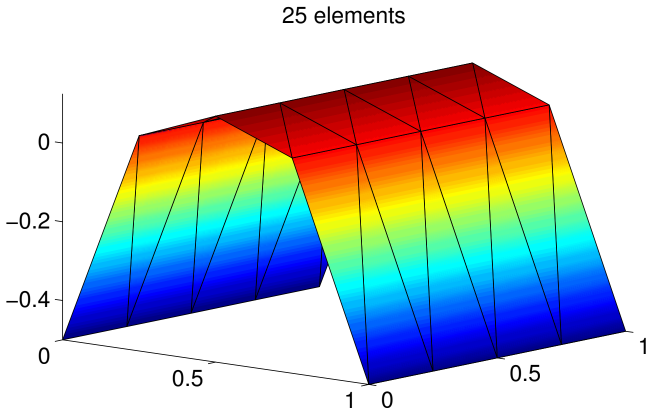


- Coarse elements OK for linear field

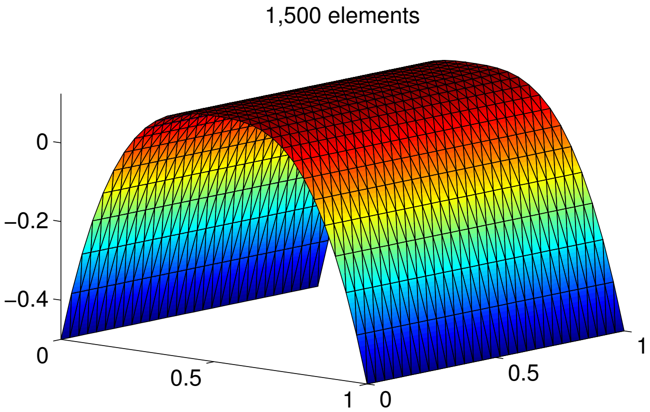
Non-Linear field



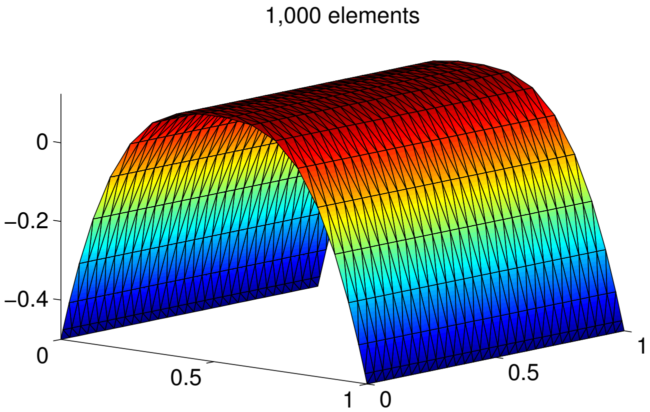
Non-Linear field



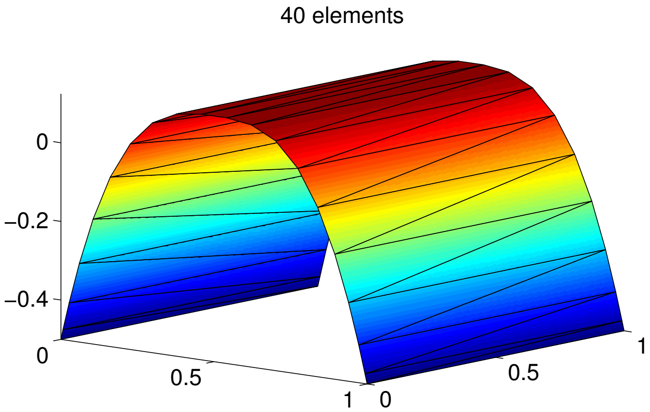
Non-Linear field



Non-Linear field



Non-Linear field



Anisotropic mesh refinement

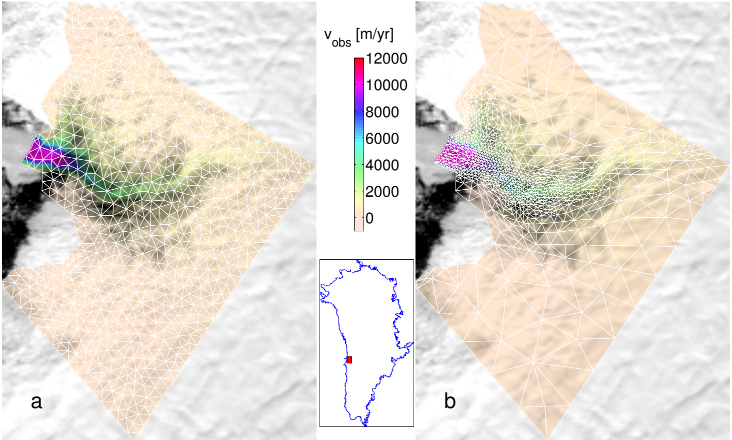
Strategy:

- Minimize the interpolation error for a given field
- Metric based on field's Hessian matrix (second derivative)

Anisotropic mesh refinement

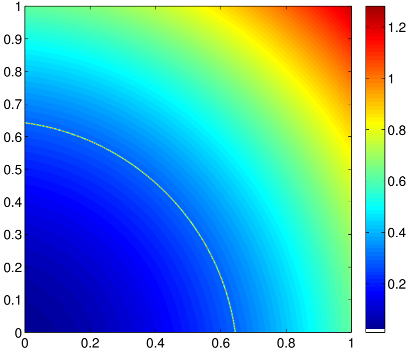
Strategy:

- Minimize the interpolation error for a given field
- Metric based on field's Hessian matrix (second derivative)



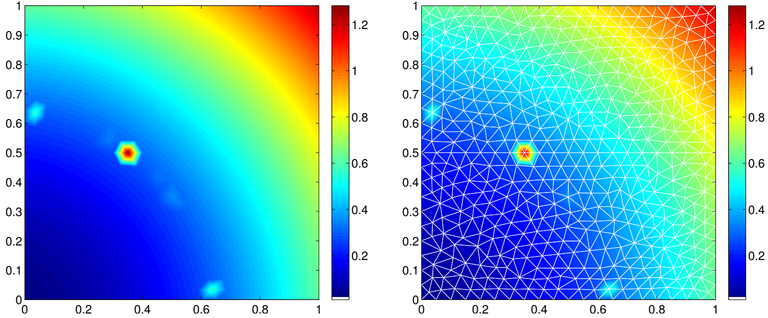
Field to capture

$$f(x,y) = \exp\left(-\left(\frac{r-0.75}{\varepsilon}\right)^2\right) + 0.5r^2$$
$$\text{with} \quad \varepsilon = 0.25 \quad \text{and} \quad r = (x+0.1)^2 + (y+0.1)^2$$



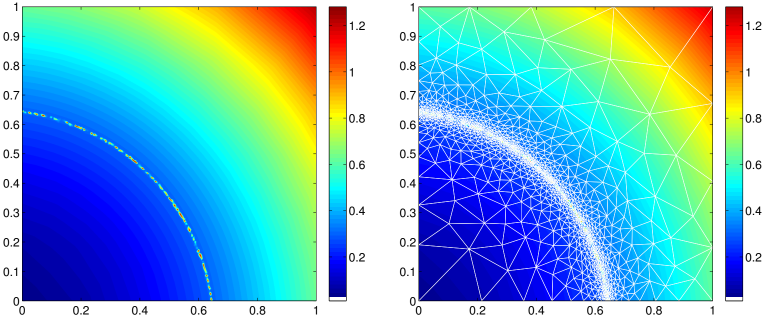
Uniform mesh

```
md=bamg(model,'domain','Square.exp','hmax',.05);
vel=shock(md.mesh.x,md.mesh.y);
plotmodel(md,'data',vel,'edgecolor','w');
```



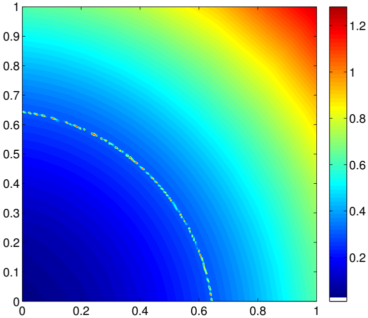
Mesh refinement

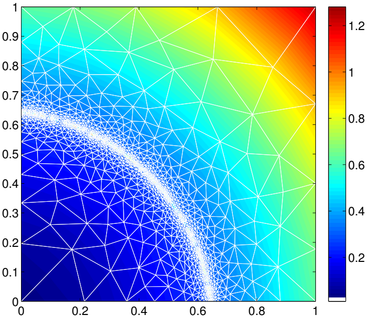
```
md=bamg(model,'domain','Square.exp','hmax',.005);
vel=shock(md.mesh.x,md.mesh.y);
md=bamg(md,'field',vel,'err',0.05,'hmin',0.005,'hmax',0.3);
vel=shock(md.mesh.x,md.mesh.y);
plotmodel(md,'data',vel,'edgecolor','w');
```



Mesh refinement

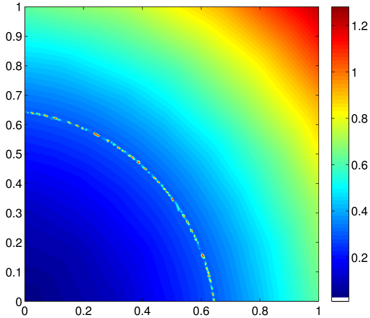
```
md=bamg(model,'domain','Square.exp','hmax',.005);
vel=shock(md.mesh.x,md.mesh.y);
md=bamg(md,'field',vel,'err',0.03,'hmin',0.005,'hmax',0.3);
vel=shock(md.mesh.x,md.mesh.y);
plotmodel(md,'data',vel,'edgecolor','w');
```

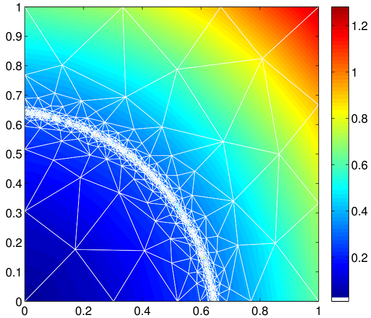




Mesh refinement

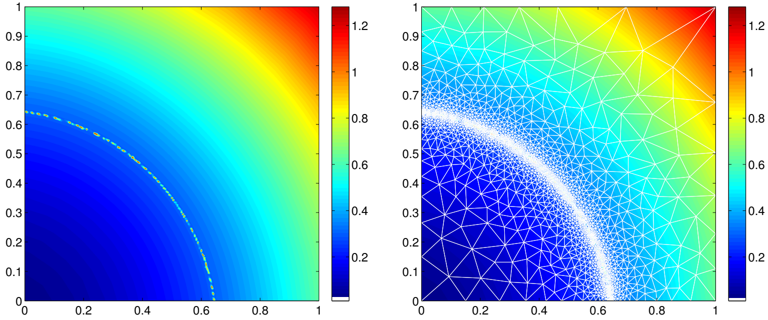
```
md=bamg(model,'domain','Square.exp','hmax',.005);
vel=shock(md.mesh.x,md.mesh.y);
md=bamg(md,'field',vel,'err',0.03,'hmin',0.005,'hmax',0.3,'gradation',3);
vel=shock(md.mesh.x,md.mesh.y);
plotmodel(md,'data',vel,'edgecolor','w');
```





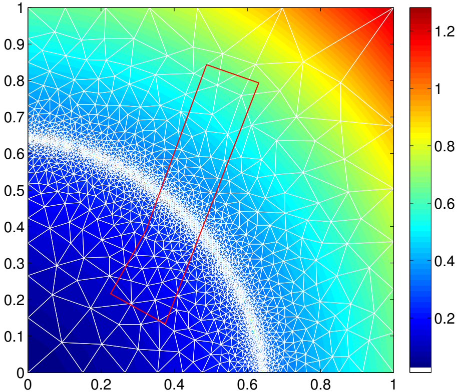
Mesh refinement

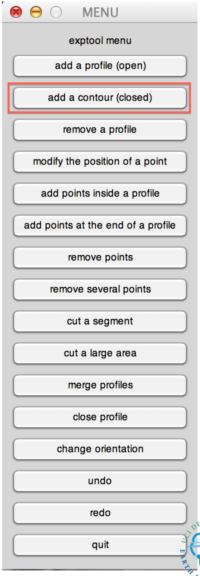
```
md=bamg(model,'domain','Square.exp','hmax',.005);
vel=shock(md.mesh.x,md.mesh.y);
md=bamg(md,'field',vel,'err',0.03,'hmin',0.005,'hmax',0.3,'gradation',1.3,'anisomax',1);
vel=shock(md.mesh.x,md.mesh.y);
plotmodel(md,'data',vel,'edgecolor','w');
```



Mesh refinement

```
plotmodel(md,'data',vel,'edgecolor','w');
exptool('refinement.exp')
```



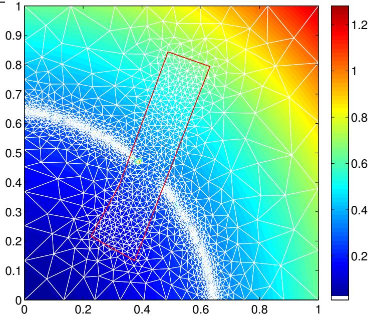


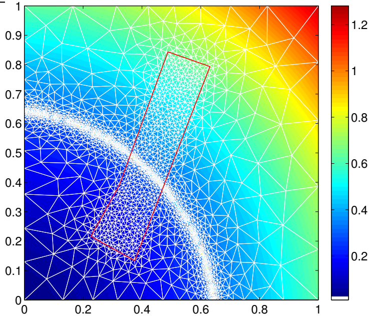
Mesh refinement

```
md=bamg(model,'domain','Square.exp','hmax',.005);

h  = NaN*ones(md.mesh.numberofvertices,1);
in = ContourToNodes(md.mesh.x,md.mesh.y,'refinement.exp',1);
h(find(in))=0.02;
plotmodel(md,'data',in);

vel=shock(md.mesh.x,md.mesh.y);
```

```
md=bamg(md,'field',vel,'err',0.03,'hmin',0.005,'hmax',0.3,'hVertices',h);
```

```
md=bamg(md,'field',vel,'err',0.03,'hmin',0.005,'hmax',0.3,'hmaxVertices',h);
```



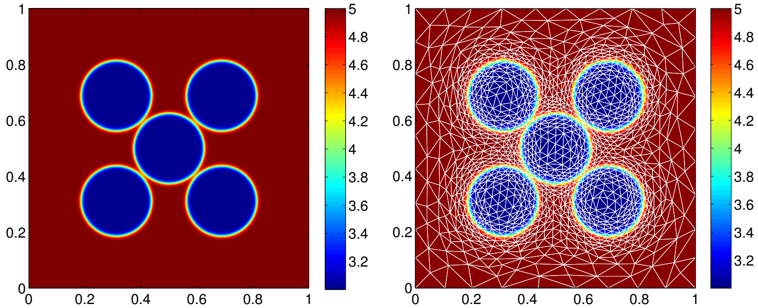


Example 2

Similar example with `circles.m`

$$
\begin{aligned}
f(x,y) = & \tanh\left(30\left(u^2+v^2-\varepsilon\right)\right)\\
& + \tanh\left(30\left((u-0.75)^2+(v-0.75)^2-\varepsilon\right)\right) + \tanh\left(30\left((u-0.75)^2+(v-0.75)^2+\varepsilon\right)\right)\\
& + \tanh\left(30\left((u-0.75)^2-(v-0.75)^2+\varepsilon\right)\right) + \tanh\left(30\left((u-0.75)^2-(v-0.75)^2-\varepsilon\right)\right)
\end{aligned}
$$

with $\varepsilon = 0.25$ and $u = 4x-2, \quad v = 4y-2$




Thanks!