Numerical Simulation Methods in Geophysics, Exercise 7: 2D problems with pyGIMLi

1. MGPY+MGIN

thomas.guenther@geophysik.tu-freiberg.de

TU BERGAKADEMIE FREIBERG

TUBAF

Die Ressourcenuniversität.
Seit 1765.

Recap Poisson & heat PDE

- ☑ solved the Poisson equation for arbitrary x and a
- ☑ accuracy (compare analytical) depends on discretization
- ☑ time stepping for instationary (parabolic) problem
- ☑ FE integrates curvature (stiffness) and solution (mass)
- Report 1 as task in OPAL, deadline January 31, 2026
- questions or problems? additional exercise time needed?

Tasks for today

- dive into pyGIMLi
- solve 2D Poisson problem with singular source

$$-\boldsymbol{\nabla} \cdot \sigma \boldsymbol{\nabla} u = I\delta(\mathbf{r} - \mathbf{r}_s)$$

Analytical solution: $u = -\frac{I}{2\pi\sigma} \ln r$

Secondary field approach

- split σ solution in $u = u_0 + u_a$

$$\Rightarrow -\boldsymbol{\nabla} \cdot \sigma \boldsymbol{\nabla} u_a = \boldsymbol{\nabla} \cdot (\sigma - \sigma_0) \boldsymbol{\nabla} u_0$$

- anomalies are *secondary* sources:
- equation represented by matrix vector equation

$$\mathbf{A}^{\sigma} \mathbf{u}_a = -\mathbf{A}^{\delta\sigma} \mathbf{u}_0 = (\mathbf{A}^{\sigma_0} - \mathbf{A}^{\sigma}) \mathbf{u}_0$$

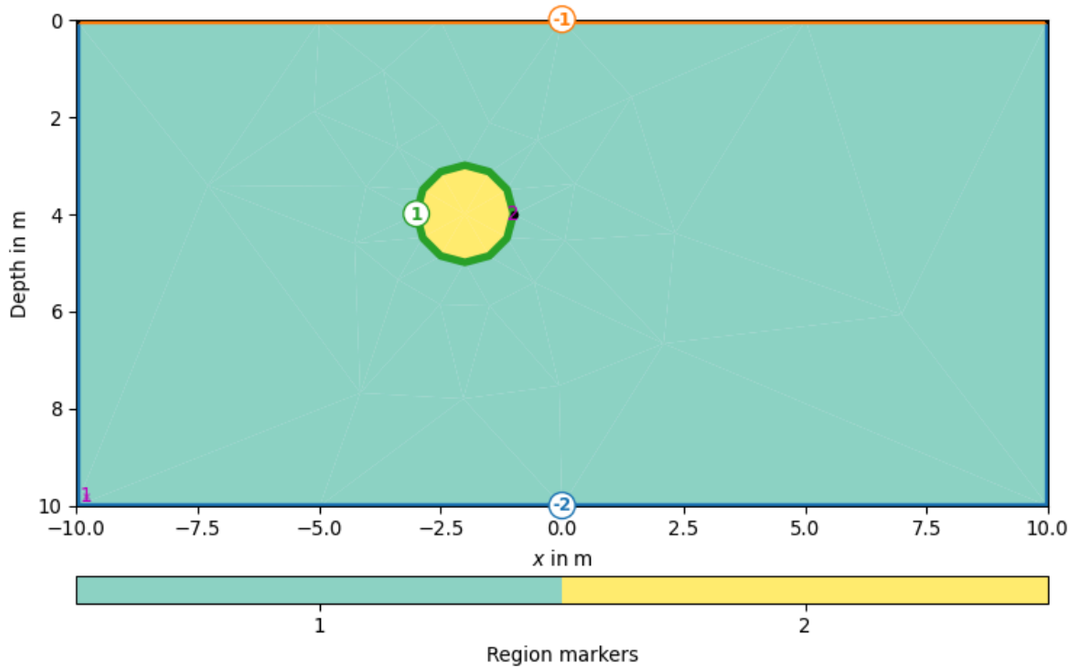
2D/3D with pyGIMLi

- `pg.createGrid(x, y)` for regular grids (also 3D)
- create geometry (PLC - piecewise linear complex) with `meshtools`
  - lots of functions for different shapes etc.
- meshing by Triangle algorithm (J. Shewchuk) or TetGen

Generate a geometry

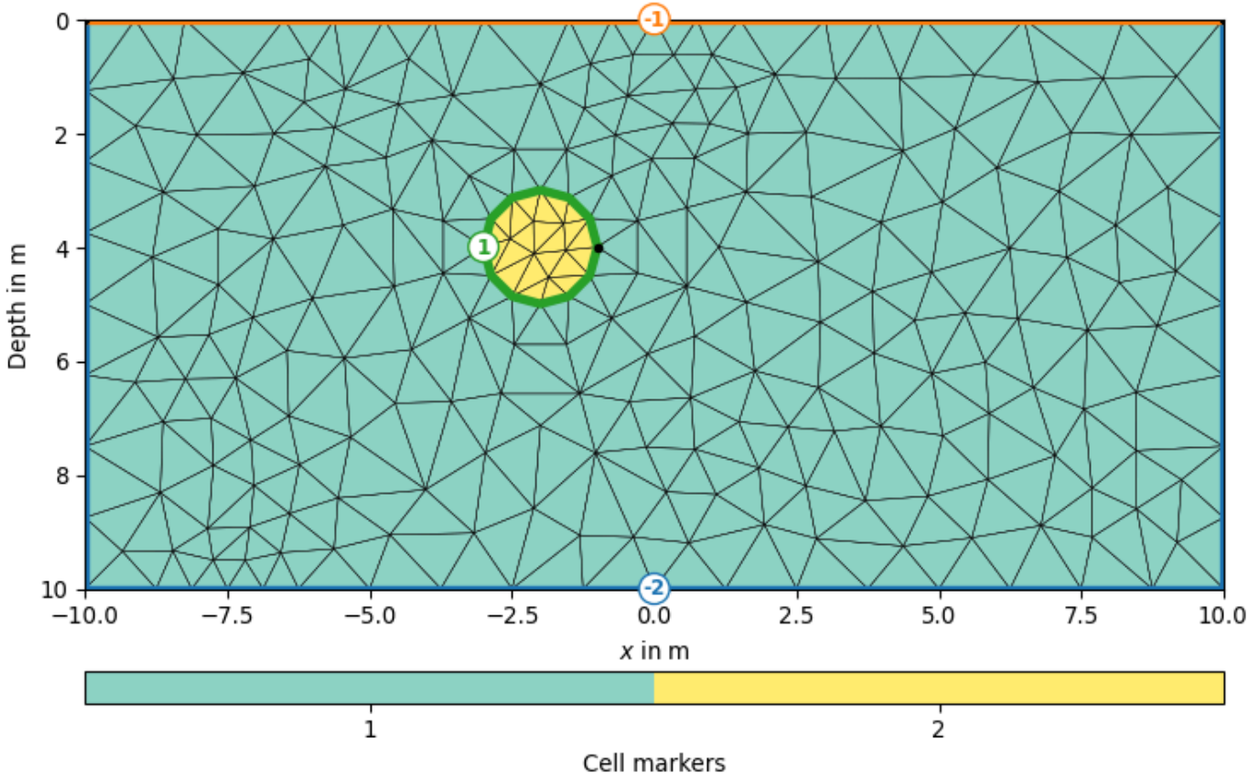
```
1  import pygimli as pg
2  import pygimli.meshtools as mt
3  world = mt.createWorld(start=[-10, -10], end=[10, 0])
4  world += mt.createCircle(pos=[-2, -4], radius=1, marker=2)
5  print(world)
6  pg.show(world, boundaryMarkers=True);
```

```
Mesh: Nodes: 16 Cells: 0 Boundaries: 16
```



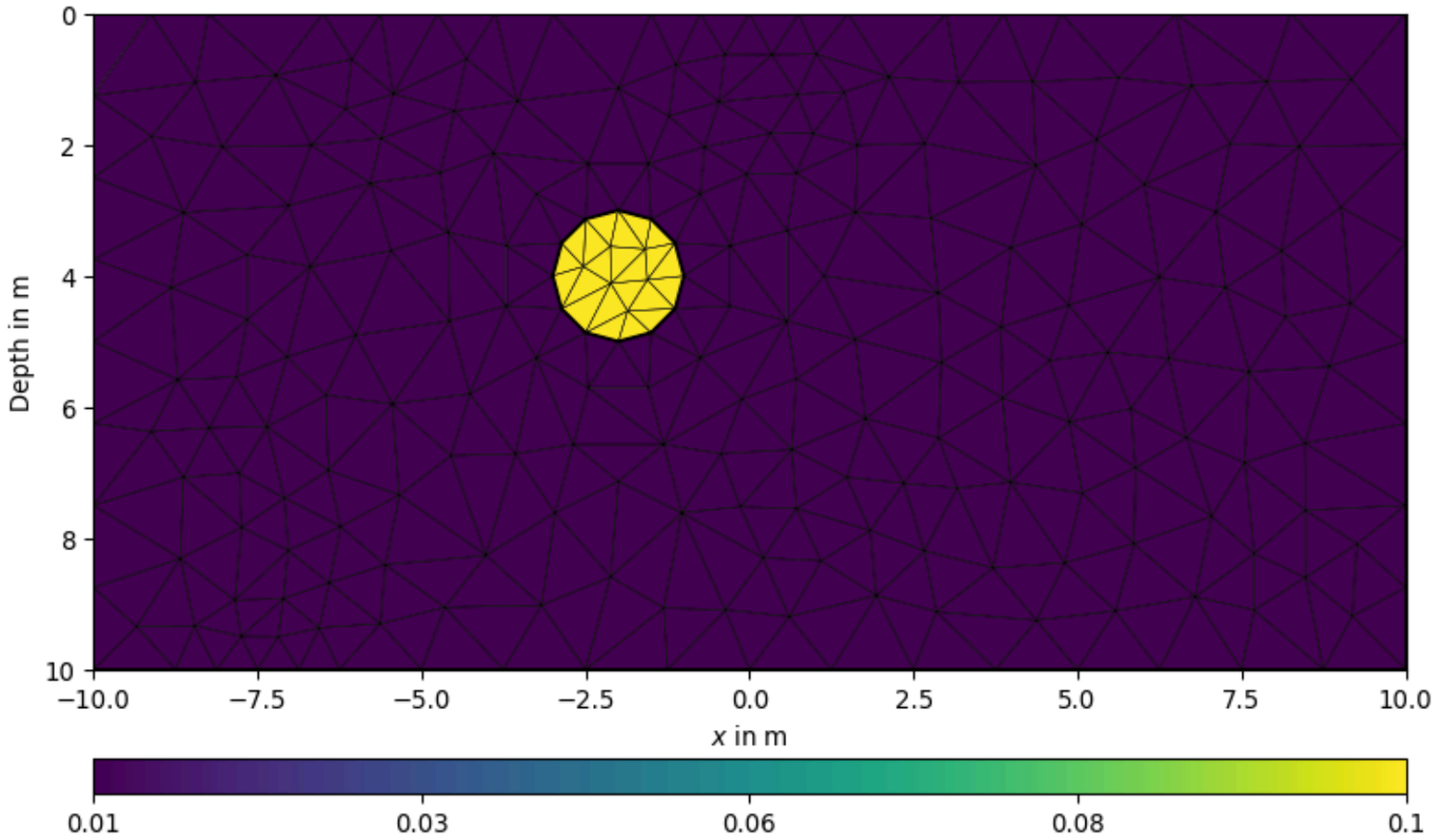
Generate a mesh

```
1  sourceA, sourceB = [-3, 0], [3, 0]
2  world.createNode(sourceA)
3  world.createNode(sourceB)
4  mesh = mt.createMesh(world, quality=34, area=1)
5  pg.show(mesh, markers=True, showMesh=True);
```



Attribute 2D conductivity

```
1  sigma0 = 1 / 100  # 100 Ohmm
2  sigma = mesh.populate("sigma", {1: sigma0, 2: sigma0*10})
3  pg.show(mesh, "sigma", showMesh=True);
```



The pyGIMLi solver module

- ready PDE solvers `solveFiniteElements`, `solveFiniteElements`
- `createStiffnessMatrix`, `createMassMatrix` assembling
- `createLoadVector`
- `assembleDirichletBC`, `assembleNeumannBC`
- `linSolve` for linear equation solvers
- `div`, `grad` operators

Tasks

- write function for analytical solution and plot
- generate a model with two electrodes at the surface
- start with homogeneous conductivity
- create stiffness matrix and load vector
- solve the matrix-vector equation
- plot solution and compare with analytical solution
- use inhomogeneous conductivity and try secondary field approach