Lecture 21: Complex domains

Logistics: - HW7 7/9 please submit

- HW8 → 2D operators

Last time: - Completed 2D discrete ops

- $\underline{\underline{Dx}}^2 = \begin{bmatrix} \underline{\underline{I_y}} & \underline{\underline{I_y}} & & & \\ & -\underline{\underline{I_y}} & \underline{\underline{I_y}} & & \\ & & -\underline{\underline{I_y}} & \underline{\underline{I_y}} & \\ & & & -\underline{\underline{I_y}} & \underline{\underline{I_y}} \end{bmatrix} = \underline{\underline{Dx}} \otimes \underline{\underline{I_y}}$

pattern — copies

$\begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{bmatrix}$

2D Divergence: $\underline{\underline{Dx}} = \mathrm{kron}(\underline{\underline{Dx}}, \underline{\underline{I_y}})$

$\underline{\underline{Dy}} = \mathrm{kron}(\underline{\underline{I_x}}, \underline{\underline{Dy}})$

$\underline{\underline{D}} = [\underline{\underline{Dx}}, \underline{\underline{Dy}}]$

- Discrete gradient: $\boxed{\underline{\underline{G}} = -\underline{\underline{D}}^T}$

$\nabla\cdot \sim \underline{\underline{D}}$ set $\underline{\underline{G}} = 0$ on bnd

- looked at 1D → 2D example

⇒ almost nothing changes in main file

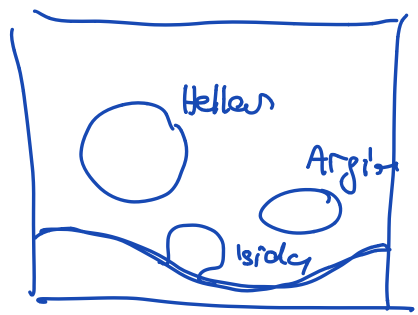
all hiden in operators

- Immediately extend non-linear solver to 2D

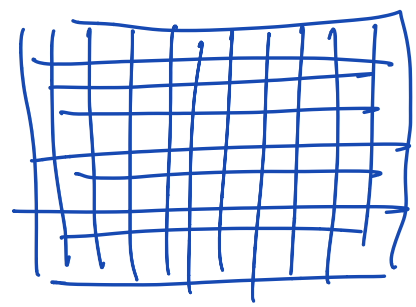
Complex domains

Motivation:

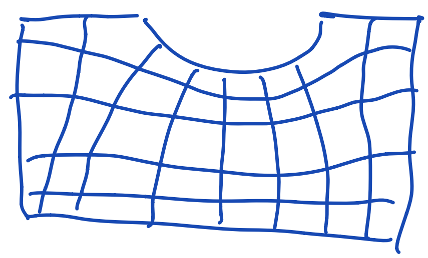
- Dichotomy boundary is not straight
- Subtract craters out of domain

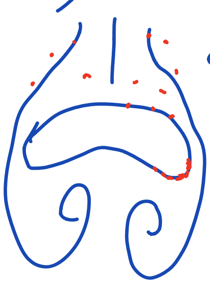


- Discrete ops for a regular cartesian grid



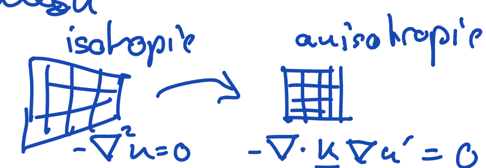
A) Curvilinear bnd fitted mesh





← Marc's bubble adventures

PRO
- represent geom on a relatively coarse mesh
- looks good

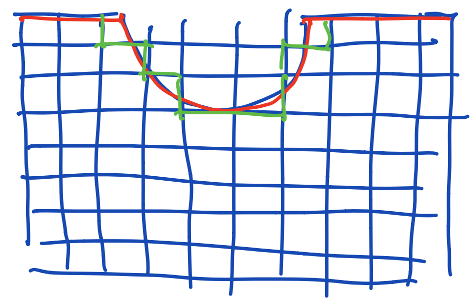


- introduce all infra structure for tensor properties

CONs: • significant complication

- many numerical pitfall
- "limited to rel. simple geometries"

B) Embedded boundary



PRO: • simple to implement
• arbitrarily complex (pore scale)

CON: - Need a fine mesh
- does not look as impressive

Note: Often people try to do this by setting K either very high or very low.

$\Rightarrow$ BC is not enforced properly

$\Rightarrow$ very ill posed matrix

$\Rightarrow$ not reducing problem size.

$\Longrightarrow$ Live script demo_compex_domains.mlx

Step 1: Find cells in crater

$$\underline{d} = \sqrt{(\underset{X(:)}{x} - x_0)^2 + (\underset{Y(:)}{\cancel{x}\, y} - y_0)^2} \quad \leftarrow \text{meshgrid}$$

$$\text{dof\_in} = \text{Grid.dof}(\underline{d} \le R_c)$$

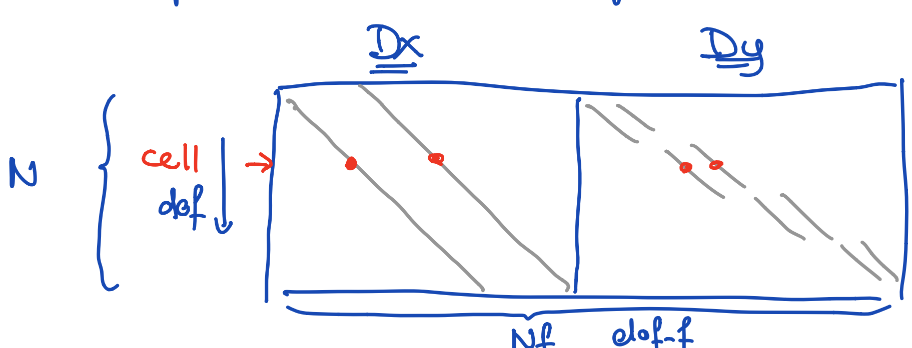
$$\text{dof\_out} = \text{Grid.dof}(\underline{d} > R_c)$$

Step 2: Find faces on bnd of crater

Gived dof of a cell what are dof_f's of the associated faces?

$\Rightarrow$ this info is in $\underline{\underline{D}}$

Each row of $\underline{\underline{D}}$ computes divergence of a cell from the fluxes across its faces

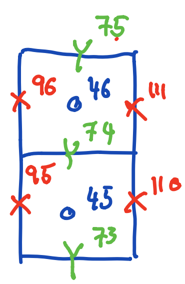


Each row has only 4 non-zero entries corresponding to the four faces of the cell.

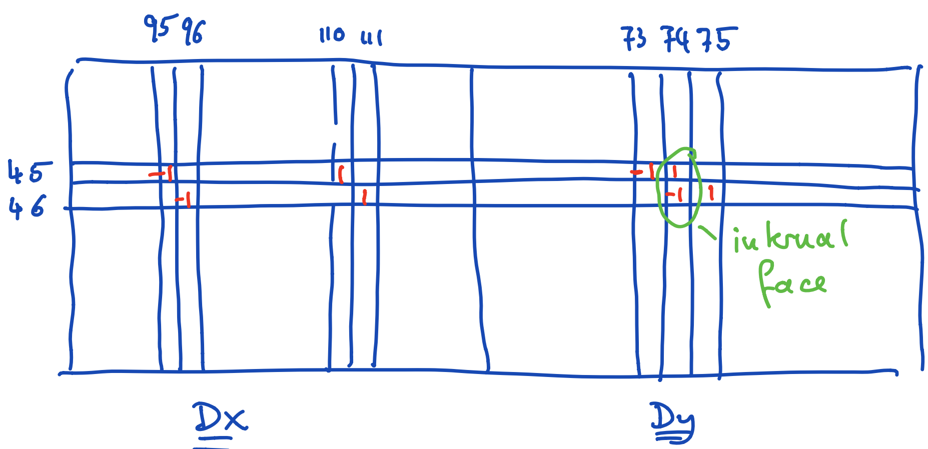
$\Rightarrow$ column indices of 4 non-zero entries are the dof-f of the faces of the cell

But we only want the exterior faces that form the bud of crater.

How can we tell if a face is external to a group of cells?



74 is internal face



If two cells share face the column corresponding to the face has two entries of same magnitude but opposite sign.

Determine external faces:

1) Select all rows of $\underline{\underline{D}}$ corresponding to cells in the crater

$\underline{\underline{Din}} = \underline{\underline{D}}(\underline{dof\_in}, :);$

2) Sum the columns

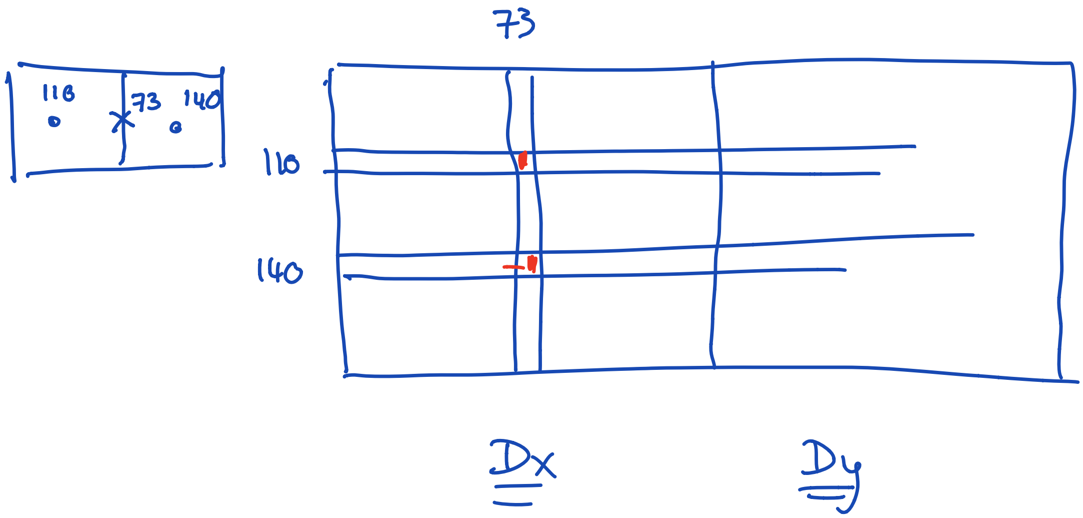
$\text{sum}(\underline{\underline{Din}}, 1) \Rightarrow$ 1 by Nf vector with non-zero entries in the position of the external faces

$$dof\_f\_bnd = Grid.dof\_f(\underbrace{abs(sum(\underline{\underline{Din}}, 1)) > \varepsilon}_{\text{vector of } 0,1});$$

3) Find the cells along the crater boundary

Given a vector of face dof_f what are the associated cells?

Again info is $\underline{\underline{D}}$



The non-zero entries in a column show which cells are associated with given face

To find cells along bnd.

1) Select all columns of $\underline{D}$ corresponding to dof_f_bnd

$\underline{Db} = \underline{D}(:, \text{dof\_f\_bnd});$

2) Sum rows

sum ($\underline{Db}$, 2) $\Rightarrow$ N by 1 column vector with non-zero entries in locations corresponding to cells along the boundary

$dof\_bnd = Grid.dof(abs(sum(\underline{\underline{Db}},2)) > \varepsilon);$

3) Split dof_bnd into cells in active domain

intersecting dof_bnd with dof_out

or just by comparing ~~radius~~ distance from

center with radius.