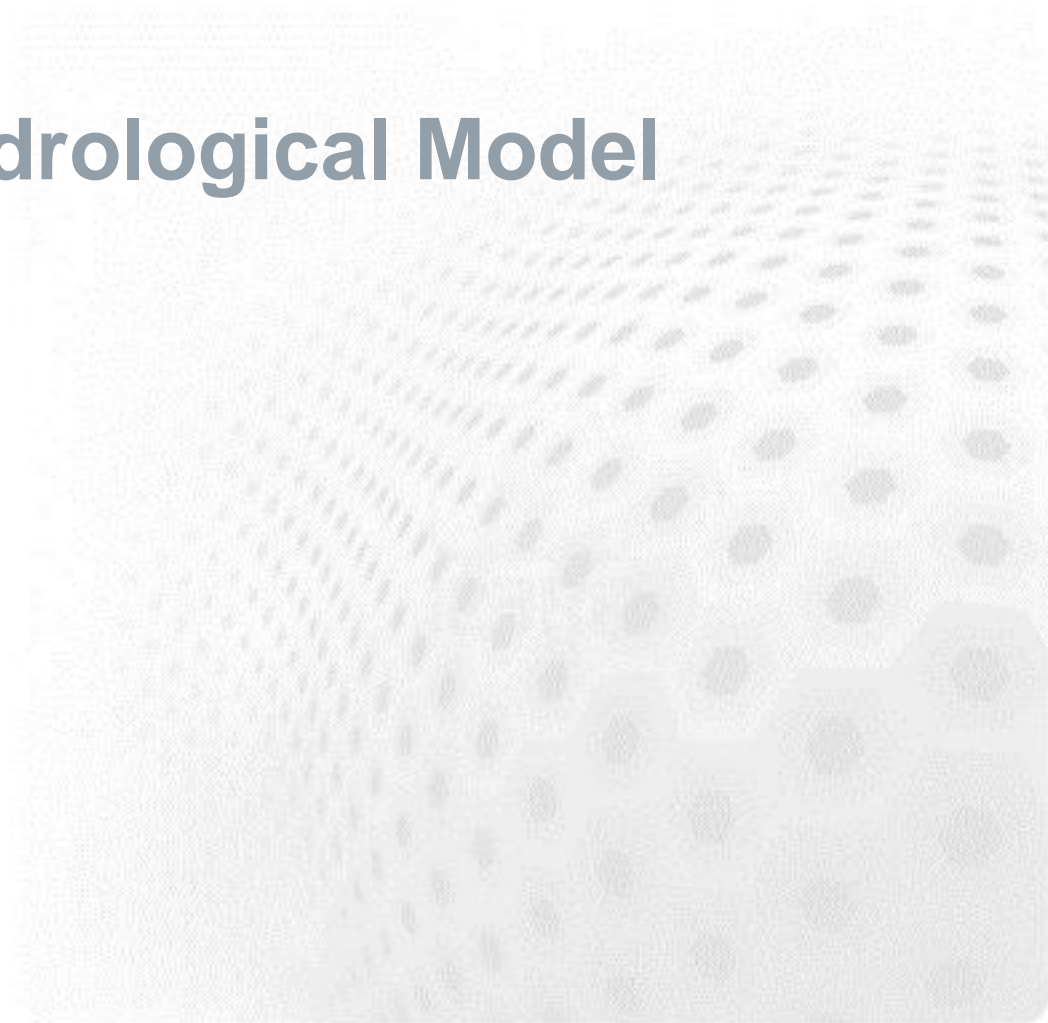




Elmer/Ice advanced course

Simple Hydrological Model



Contents

The aim of this exercise is to compute a simplified hydrological potential from given geometry

- Definition of hydrological potential
- Steps of implementation in a user defined solver
 - Principle structure
 - Connection to Solver Input File
 - Compilation
- Setup of Solver Input File

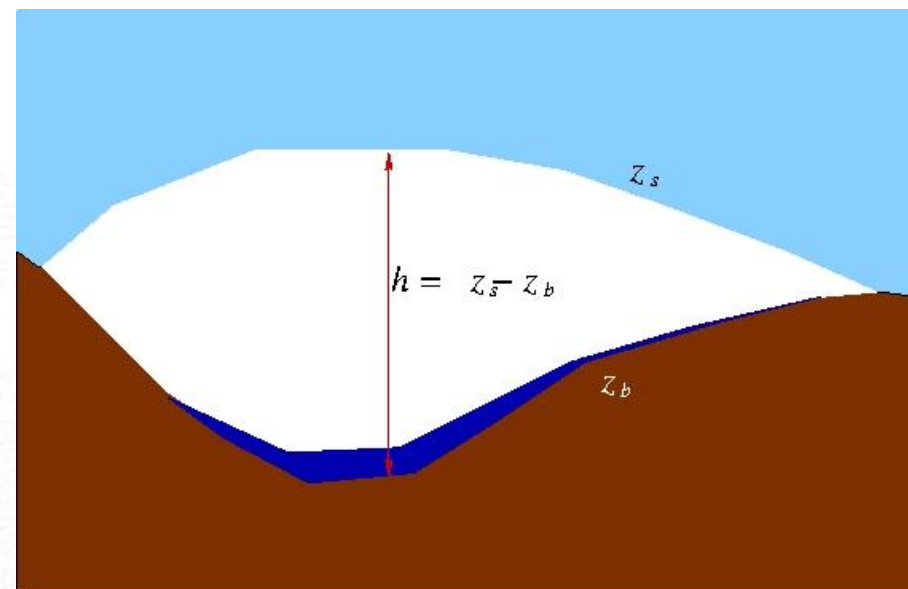
Potential

- Free surface: z_s
- Bedrock: z_b
- Thickness: $h = z_s - z_b$
- Hydraulic Potential:

$$\Phi = \rho_w g z_b + p_w$$

- Water pressure: $p_w = \rho_i g h - N$

- Effective pressure: $N \approx 0 \Rightarrow p_w \approx \rho_i g h$

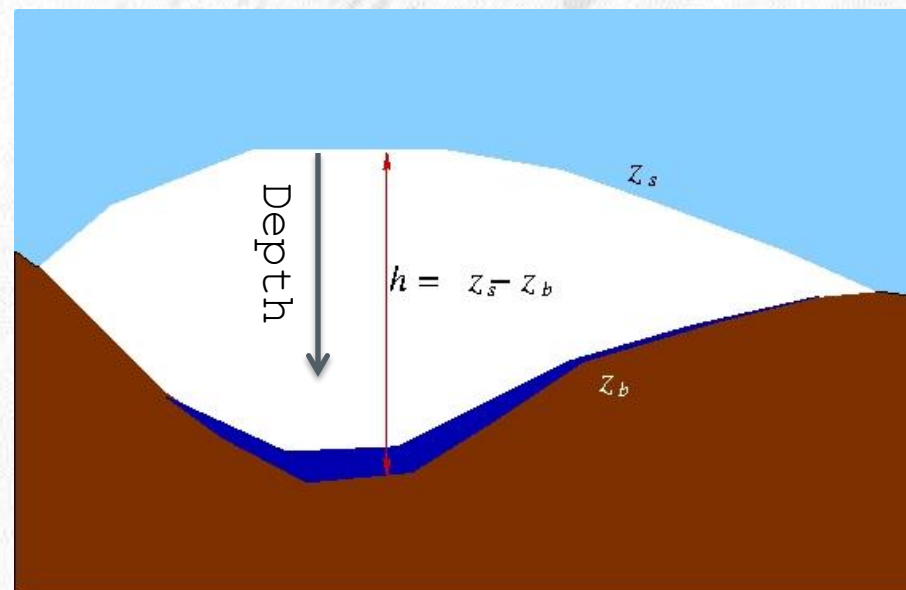


$$\Phi = [\rho_i z_s + (\rho_w - \rho_i) z_b] g$$

Potential

- In Elmer: Depth
- @ bedrock: $\text{Depth} = h$
- Write routine that solves for:
 - Reads existing variable Depth
 - Reads in Material parameters ρ_w, ρ_i
 - Reads in gravity from body force g

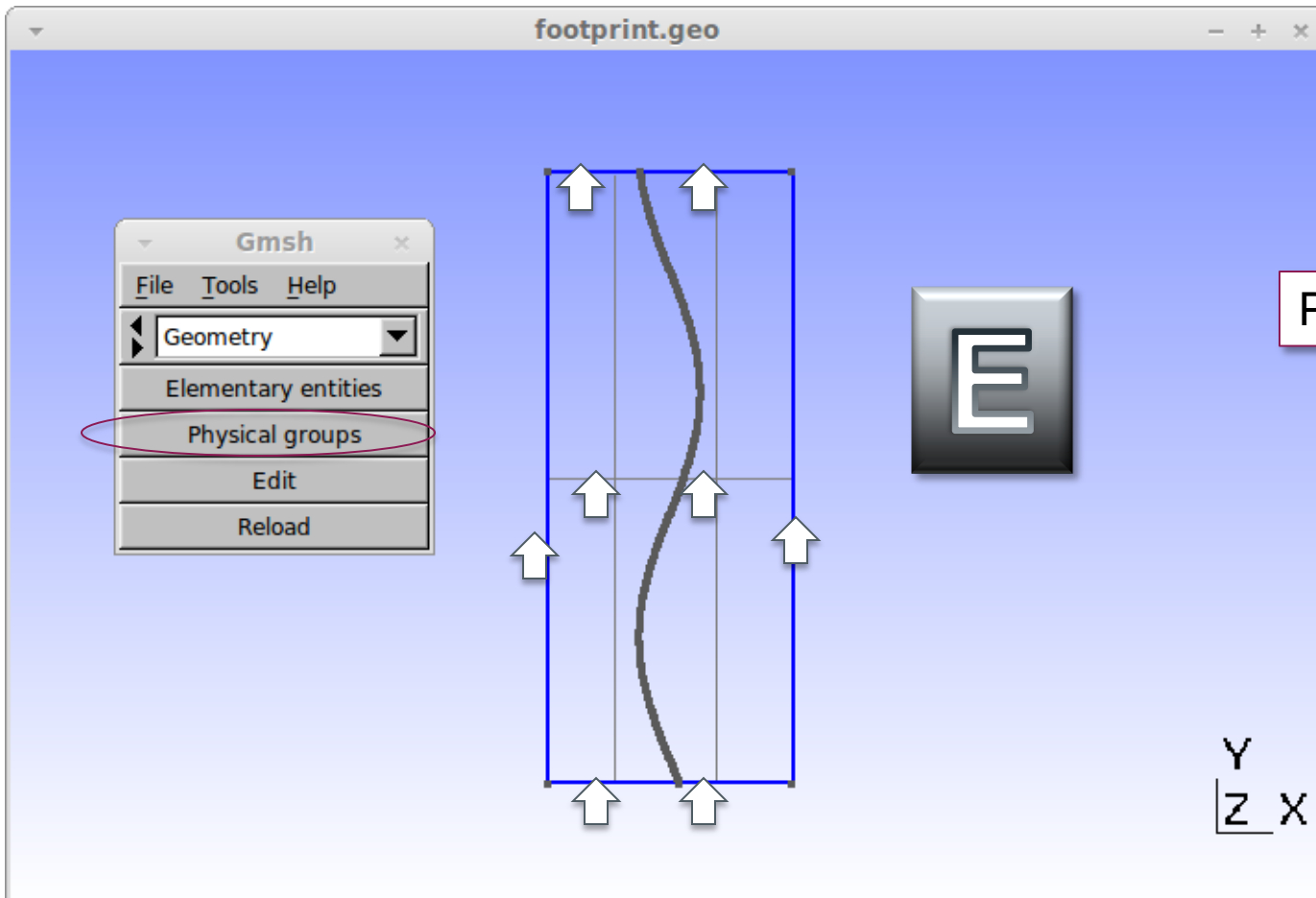
$$\Phi = [\rho_i h + \rho_w z_b] g$$



Prerequisites

- Instructions to do everything from scratch found in the README file
- The material contains a Python script that would produce a DEM and a footprint mesh
- We start from an existing DEM and footprint mesh in GMSH format
- Open the mesh:

```
> cd Mesh  
> gmsh footprint.geo
```



Physical groups

Add

Surface

Line

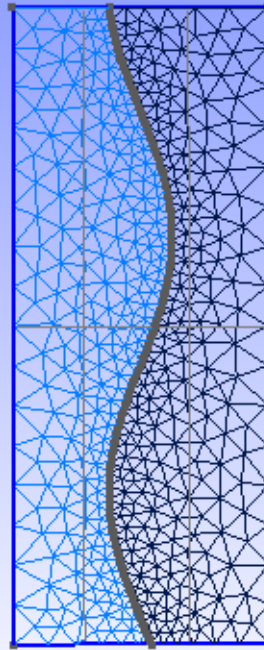
```
Info : -----
Info : Gmsh version   : 2.5.1
Info : Build OS      : Linux
Info : Build options  : Ann Bamg Bfgs Blas Blossom Chaco DIntegration Dlopen FlTree Fltk Gmm Have6
Info : Build date    : 20111108
Info : Build host    : allspice
Info : Packager      : buildd
Info : Home directory : /home/zwinger/
Info : Launch date   : Tue Oct 29 10:21:21 2013
Info : Command line  : gmsh footprint.geo
Info : -----
Info : Reading 'footprint.geo'...
Info : Done reading 'footprint.geo'
```



Select lines
[Press 'e' to end selection or 'q' to abort]

Gmsh

- File
- Tools
- Help
- Mesh
- Define
- 1D
- 2D
- 3D
- Optimize 3D
- Optimize 3D (Netgen)
- Set order
- Inspect
- Refine by splitting
- Partition
- Reclassify 2D
- Delete
- Save



Y
Z X

Mesh
2d
Save

File
Exit

```

Info : ...
Info : ...t.geo'
Info : ...
Info : Meshing 1D...
Info : : Meshing curve 1 (Line)
Info : : Meshing curve 2 (Line)
Info : : Meshing curve 3 (Line)
Info : : Meshing curve 4 (Line)
Info : : Meshing curve 5 (Line)
Info : : Meshing curve 6 (Line)
Info : : Meshing curve 7 (Nurb)
Info : : Done meshing 1D (0.032002 s)
Info : : Meshing 2D...
Info : : Meshing surface 9 (Plane, Delaunay)
Info : : Meshing surface 11 (Plane, Delaunay)
Info : : Done meshing 2D (0.016001 s)
Info : : 744 vertices 1295 elements

```


Convert Mesh

● Conversion using ElmerGrid:

```
> ElmerGrid 14 2 footprint.msh  
-autoclean -order 1.0 0.1 0.01
```

● If you want to have a look in ElmerPost:

```
> ElmerGrid 14 3 footprint.msh  
-autoclean -order 1.0 0.1 0.01
```

External extrusion

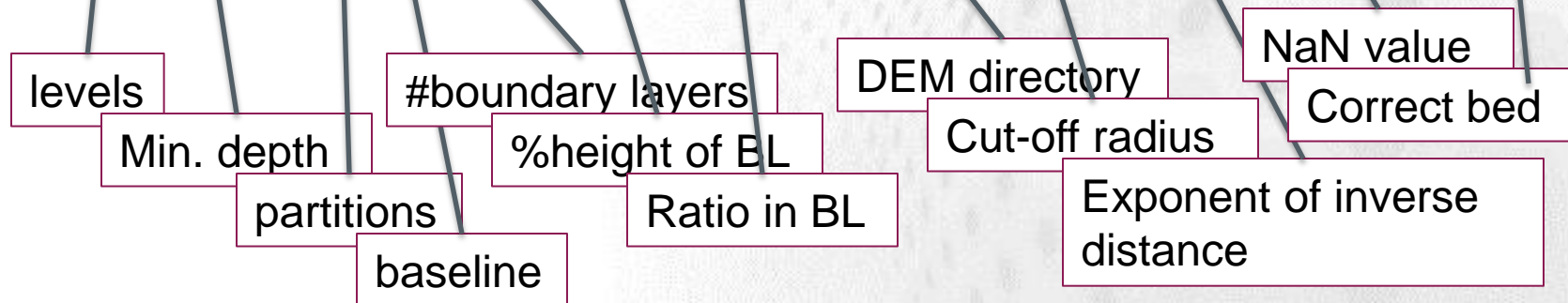
- Extruding mesh using ExtrudeMesh

- You can find it under

- `trunk/elmerice/Meshers/ExtrudeMesh.c`

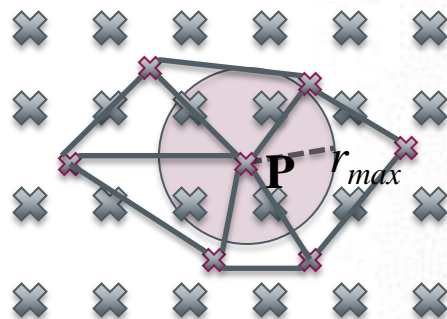
- Extrusion done using DEM:

```
ExtrudeMesh footprint extruded_bounded
10 9.99 1 1 0 0.00 0.00 DEM 15.00 2.00 -9999.0 0
```



External extrusion

➤ Inverse distance



$$h(\mathbf{P}) = \left(\sum_{\|\mathbf{P} - \mathbf{X}_i\| < r_{\max}} h_i \|\mathbf{P} - \mathbf{X}_i\|^m \right) / \left(\sum_{\|\mathbf{P} - \mathbf{X}_i\| < r_{\max}} \|\mathbf{P} - \mathbf{X}_i\|^m \right)$$

Setup of run

➤ Initializing velocity field: **diagnostic.sif**

➤ Solvers:

1. Depth: ElmerIceSolvers FlowDepthSolver

2. Height: ElmerIceSolvers FlowDepthSolver

3. Flow: Equation= "NavierStokes"

4. Free Surface: "FreeSurfaceSolver"
"FreeSurfaceSolver"

← Never executed,
as diagnostic

5. Mesh: Equation = "Mesh Update"

6. VTU (ParaView): "ResultOutputSolve"
"ResultOutputSolver"

> **ElmerSolver diagnostic.sif**

Prognostic run

- New SIF:
- Switching to transient (time evolving)
 - Restart from previous solution
 - Enable mesh deformation
 - Need accumulation ablation function (MATC)

```
$ function accum(X) {\n  x0 = -\n    sin(2.0*3.1415927*X(1)/2000.0)*125.0;\n  dx = abs(x0 - X(0));\n  if (X(2) > 500.0) \n    { _accum = 0.0; }\n  else \n    { netaccum = ((11.0/2750.0) * X(2) -\n      0.85); \n      if (netaccum > 0.0) \n        { _accum = (1.0 - dx/250.0) *\n          netaccum; }\n      else { _accum = netaccum; }\n    } \n}
```

> **ElmerSolver prognostic.sif**

Hydropotential

- Compile function:

- > `elmerf90 SRC/GetHydroPotential.f90`
`-o GetHydroPotential.so`

- Run, by restarting (from post.result)

- > `ElmerSolver hydrotest.sif`