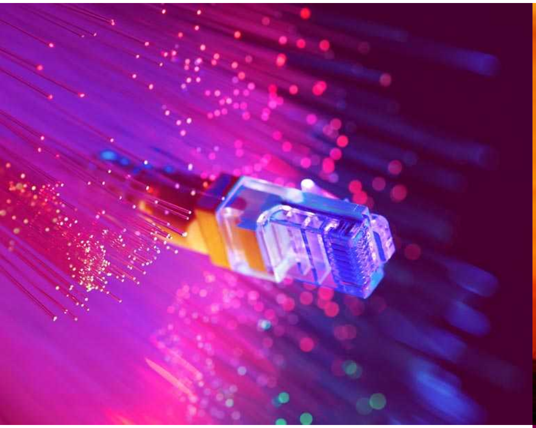


Elmer/Ice – New Generation Ice Sheet Model

Thomas Zwinger, Elmer/Ice beginner's course


CSC – Finnish research, education, culture and public administration ICT knowledge center

2D GLACIER TOY MODEL

These sessions shall introduce into the **basics of Elmer/Ice**. It follows the strategy of having a possibly **simple flow-line** setup, but **containing all elements** the user needs in real world examples, such as reading in DEM's, applying temperature and accumulation distributions, etc.

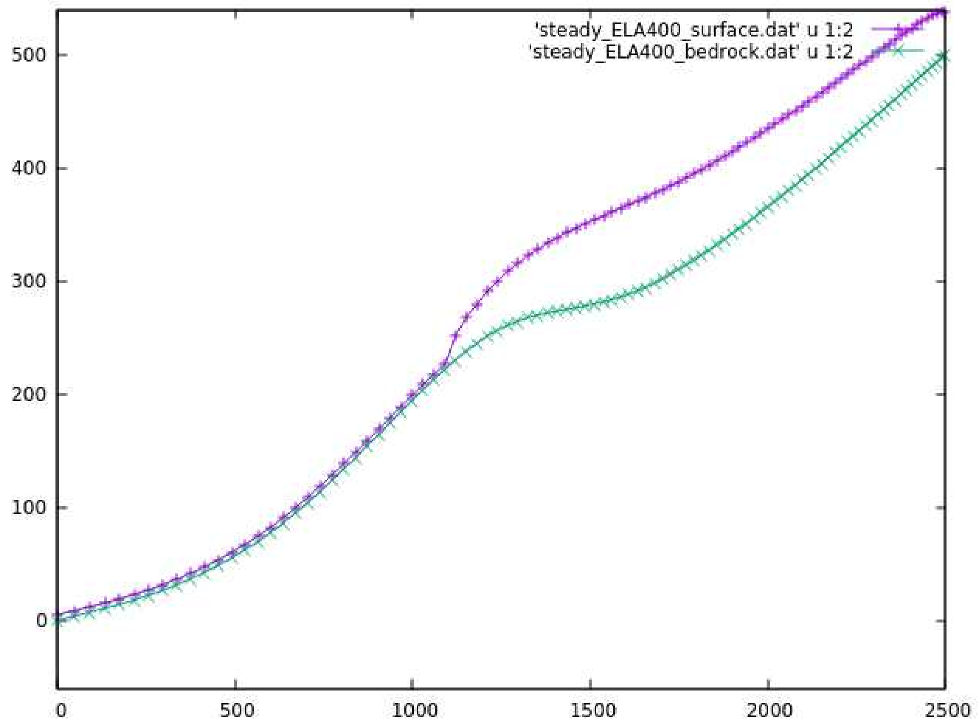
DIAGNOSTIC RUN

Starting from a given point-distribution (DEM) in 2D we show how to:

- Create the mesh
- Set up runs on fixed geometry (diagnostic)
- Introduce sliding
- Manipulate (structured) mesh shape inside Elmer
- Use tables to interpolate values
- Write a simple MATC function (interpreted functions)
- Post-process results

The diagnostic problem

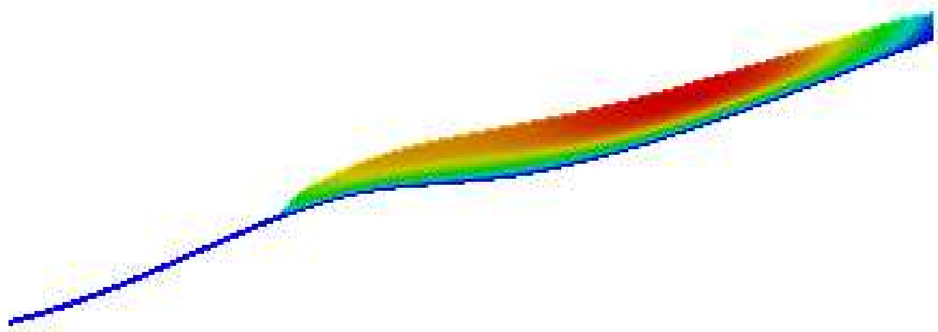
- We start from a distribution of surface and bedrock points that have been created driving a prognostic run into steady state



- The distributions are given in the files:
  `steady_ELA400_bedrock.dat, steady_ELA400_surface.dat`

The diagnostic problem

- We will study a ~11 deg inclined glacier
- We will start with a flat mesh (produced with Gmsh) of unit-height



The diagnostic problem

- If you have not already saved the mesh from Gmsh, do the following (find Gmsh instructions at end of slides):

```
$ gmsh -2 testglacier.geo
```

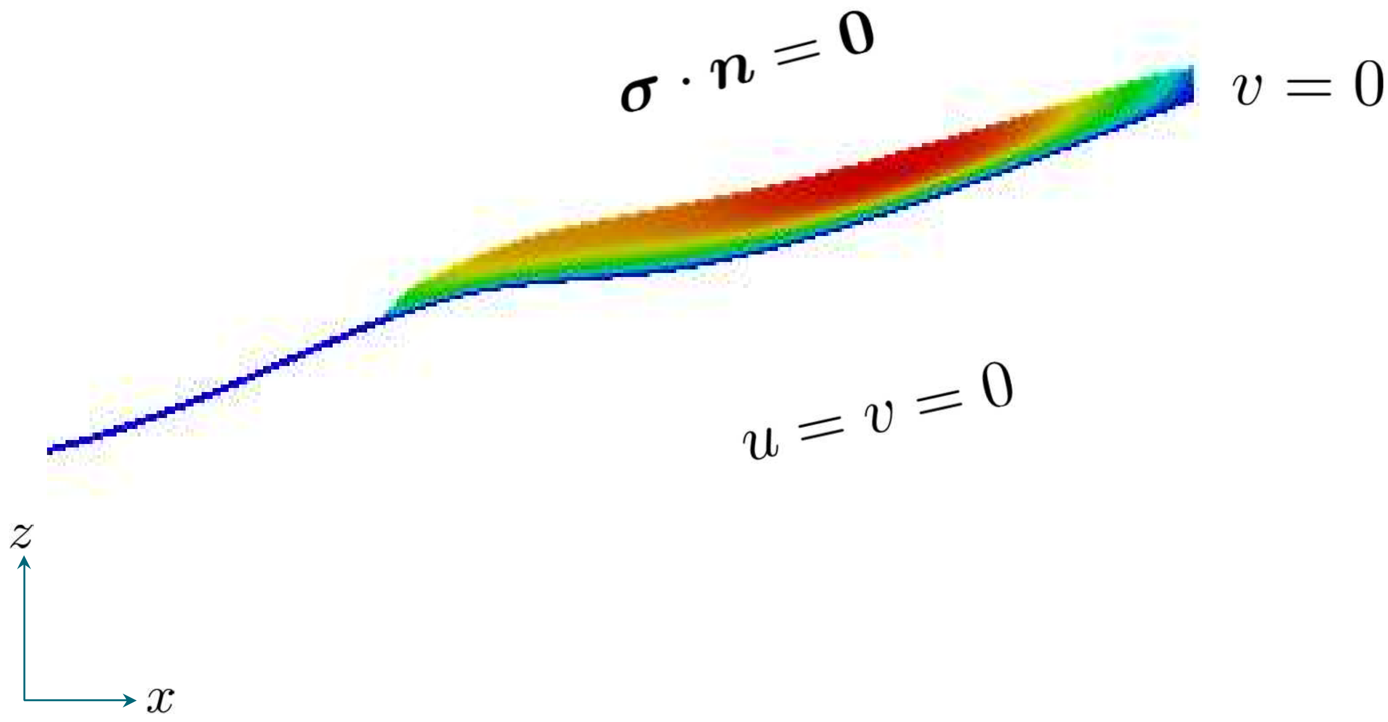
- Use ElmerGrid to convert the mesh:

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

Needed to clean up geometry

Orders the numbering in x y z –directions (highest number fastest)

The diagnostic problem



The diagnostic problem

- We will do a diagnostic simulation, i.e., we ignore the time derivative in ANY equation
  - Stokes anyhow has no explicit time dependence

    $$\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g} = \mathbf{0}$$

  - That also means, that the surface velocity distribution is a result of the given geometry and cannot be prescribed (no accumulation)
- Open the Solver Input File (SIF)

```
$ emacs Stokes_diagnostic.sif &
```

The diagnostic problem

```
!echo on
Header
  !CHECK KEYWORDS Warn
  Mesh DB "." "testglacier"
  Include Path ""
  Results Directory ""
End


Simulation
  Max Output Level = 4
  Coordinate System = "Cartesian 2D"
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = "Steady"
  Steady State Max Iterations = 1
  Output Intervals = 1
  Output File = "Stokes_ELA400_diagnostic.result"
  Post File = "Stokes_ELA400_diagnostic.vtu" ! use .ep suffix for leagcy format
  Initialize Dirichlet Conditions = Logical False
End
```

This declares our mesh; capital/small letters matter

The coordinate system (incl. Dimension)

Steady State = diagnostic

The diagnostic problem

```
Body 1
  Name = "Glacier"
  Body Force = 1
  Equation = 1
  Material = 1
  Initial Condition = 1
End
```

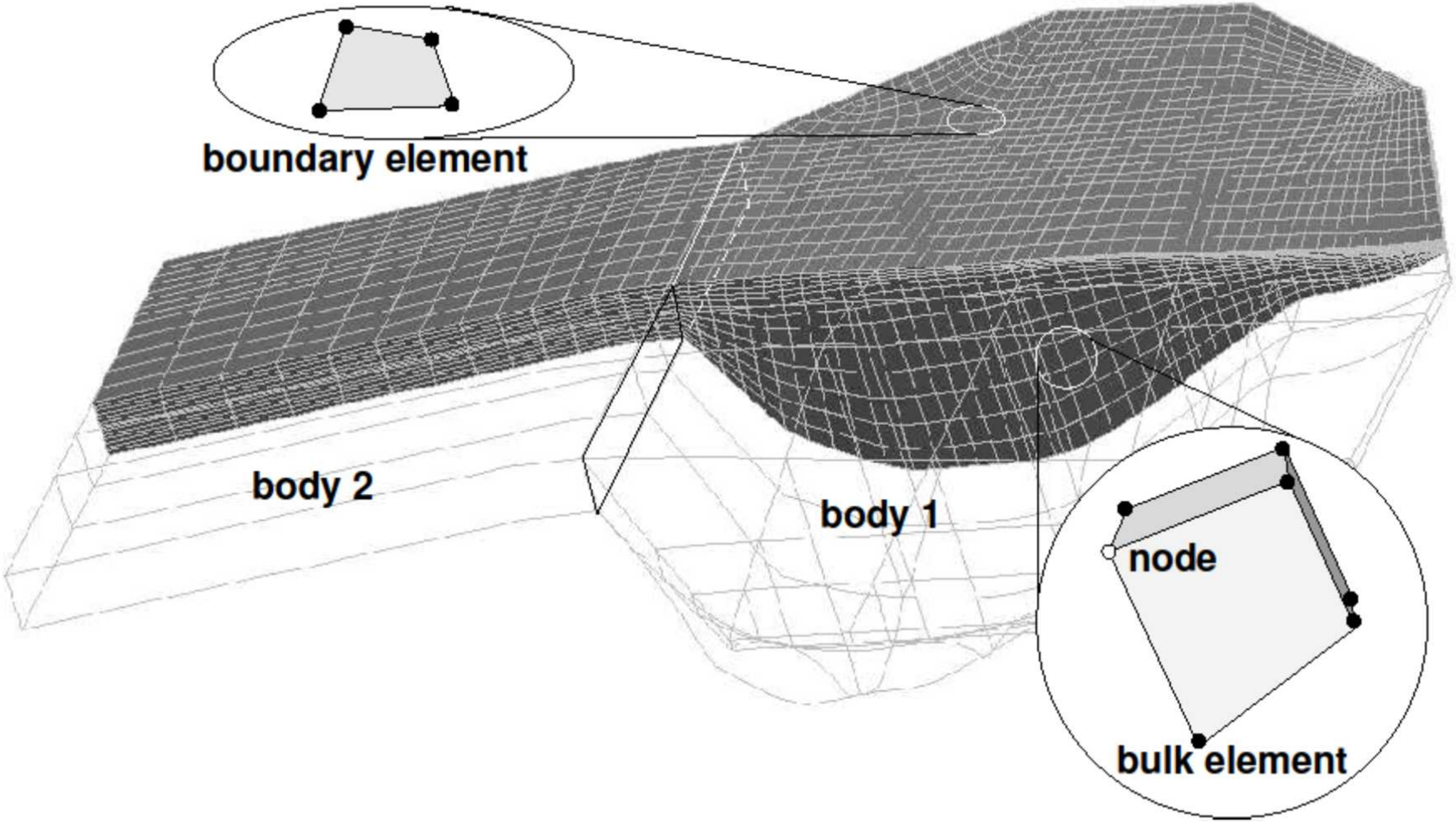
Assigns the Equation/Material/Body Force/and Initial conditionto a body

The Equation for Body 1 (see above); declares set of Solvers

```
Equation 1
  Name = "Equation1"
  Convection = "computed"
  Flow Solution Name = String "Flow Solution"
  Active Solvers(3) = 1 2 3
End

Initial Condition 1
  Velocity 1 = 0.0
  Velocity 2 = 0.0
  Pressure = 0.0
  Depth = Real 0.0
End
```

Well, as the name suggests: initial values for variables

On Bodies and Boundaries

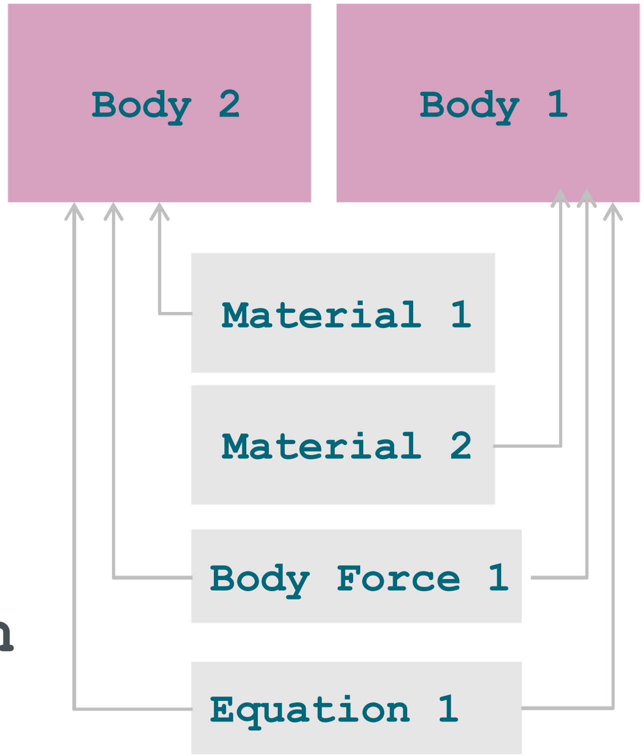


On Bodies and Boundaries

- Each **Body** has to have an **Equation** and **Material** assigned

  Body Force, **Initial Condition** optional

- Two bodies can have the same **Material/Equation/Body Force/Initial Condition** section assigned



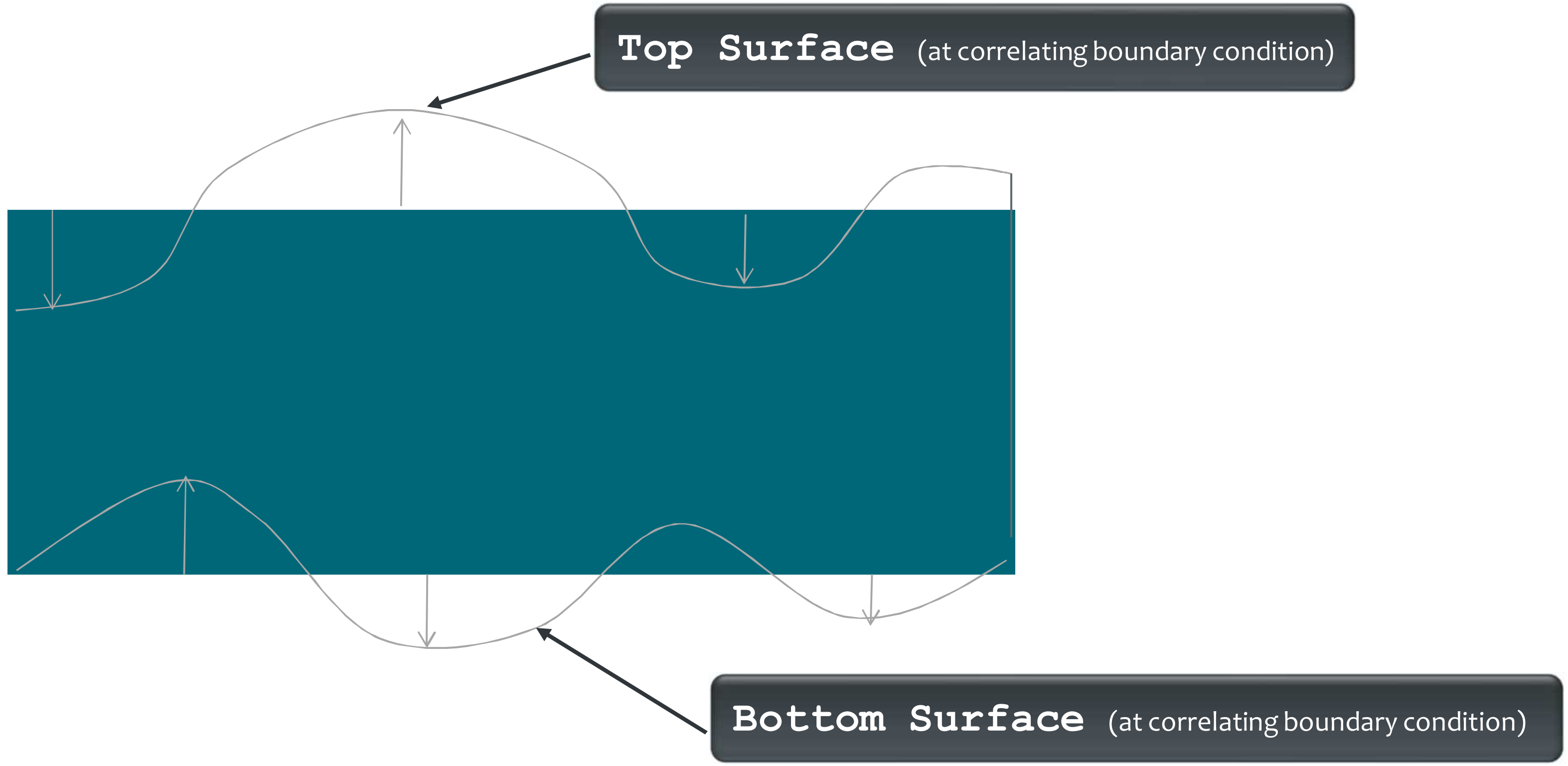
The diagnostic problem

```
! maps DEM's at the very beginning
! to originally rectangular mesh
! see Top and Bottom Surface in BC's
Solver 1
  Exec Solver = ”Before Simulation"
  Equation = "MapCoordinate"
  Procedure = "StructuredMeshMapper" "StructuredMeshMapper"
  Active Coordinate = Integer 2! the mesh-update is y-direction
! For time being this is currently externally allocated
  Mesh Velocity Variable = String "Mesh Velocity 2"
! The 1st value is special as the mesh velocity could be unrelistically high
  Mesh Velocity First Zero = Logical True
! The accuracy applied to vector-projections
  Dot Product Tolerance = Real 0.01
End
```

The primary criterion for order of execution is the **`Exec Solver`** keyword, thereafter the numbering

This solver simply projects the shape given in the input files before the run (see Exec Solver keyword) to the initially flat mesh; See **`Top Surface`** and **`Bottom Surface`** keywords later

The diagnostic problem



The diagnostic problem

```
Solver 2
  Equation = "HeightDepth"
   Procedure = "StructuredProjectToPlane" "StructuredProjectToPlane"
  Active Coordinate = Integer 2
  Operator 1 = depth
  Operator 2 = height
End
```

Computes flow depth and height based on vertically aligned ("structured") mesh

The diagnostic problem

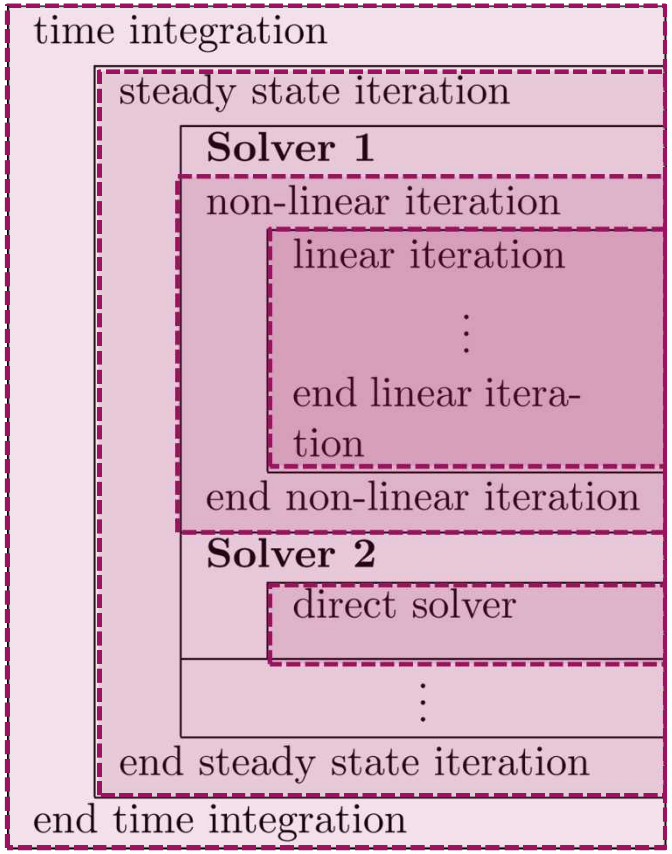
```
! the central part of the problem: the Stokes solver
Solver 3
!  Exec Solver = "Never" # uncommenting would switch this off
  Equation = "Navier-Stokes"
  Optimize Bandwidth = Logical True
  ! direct solver
  Linear System Solver = Direct
  Linear System Direct Method = "UMFPACK"
  ! alternative to above - Krylov subspace iterative solution
!  Linear System Solver = "Iterative"
!  Linear System Iterative Method =  "GCR"     !or "BICGStab"
  Linear System Max Iterations = 5000
  Linear System Convergence Tolerance = 1.0E-06
  Linear System Abort Not Converged = False
  Linear System Preconditioning = "ILU1"
  Linear System Residual Output = 1


  Steady State Convergence Tolerance = 1.0E-05
!  Stabilization Method can be [Stabilized,P2/P1,Bubbles]
  Stabilization Method = Stabilized

  Nonlinear System Convergence Tolerance = 1.0E-04
  Nonlinear System Convergence Measure = Solution
  Nonlinear System Max Iterations = 50
  Nonlinear System Newton After Iterations = 3
  Nonlinear System Newton After Tolerance =  1.0E-01
!  Nonlinear System Relaxation Factor = 0.75
End
```

`Linear System Solver` keyword chooses type of solution of the linearized problem

You need that in Stokes and also in PDE's with significant amount of convection

Account for non-linearity of the rheology

On iteration methods



1. `Timestep Intervals`
2. `Steady State Max Iterations`
3. `Nonlinear Max Iterations`
4. `Linear System Max Iterations`
4. `Linear System Convergence Tolerance`
3. `Nonlinear System Convergence Tolerance`
2. `Steady State Convergence Tolerance`
1.

The diagnostic problem

```
! we use m-yr-MPa system 1 yr = 31556926.0 sec
Material 1
  Name = "ice-ice-baby"
  Density = Real $910.0*1.0E-06*(31556926.0)^(-2.0)
  !----------------
  ! vicosity stuff
  !----------------
  Viscosity Model = String "Glen"
  ! Viscosity has to be set to a dummy value
  ! to avoid warning output from Elmer
  Viscosity = Real 1.0
  Glen Exponent = Real 3.0
  Critical Shear Rate = Real 1.0e-10
  ! Rate factors (Paterson value in MPa^-3a^-1)
  Rate Factor 1 = Real 1.258e13
  Rate Factor 2 = Real 6.046e28
  ! these are in SI units - no problem, as long as
  ! the gas constant also is
  Activation Energy 1 = Real 60e3
  Activation Energy 2 = Real 139e3
  Glen Enhancement Factor = Real 1.0
```

This is for scaling reasons (see next slide)

$$D_{ij} = A\tau_e^{n-1} S_{ij} \quad ; \quad S_{ij} = A^{-1/n} I_{D_2}^{(1-n)/n} D_{ij}$$

where $I_{D_2}^2 = D_{ij}D_{ij}/2$ and $D_{ij} = 1/2(\partial u_i/\partial x_j + \partial u_j/\partial x_i)$

$$A = A(T') = A_0 \exp^{-Q/RT'}$$

On the choice of units

Elmer(/Ice) does not assume any choice of units. This is on you, BUT, units have to be consistent amongst each other and with the mesh geometry units.
The order of magnitude in numbers do not change results, as matrix is pivoted

For the Stokes problem, one should give values for:

- the density: $\rho \quad (= 910 \text{ kg/m}^3)$
- the gravity: $g \quad (= 9.81 \text{ m s}^{-2})$
- the viscosity: $\eta_0 \quad (\text{Pa s}^{1/n}) \qquad (1 \text{ Pa} = 1 \text{ kg s}^{-2} \text{ m}^{-1})$

kg – m – s [SI] : velocity in m/s and time-step in seconds 

kg – m – a : velocity in m/a and timesteps in years  $1 \text{ a} = 31\,557\,600 \text{ s}$

MPa – m – a : velocity in m/a and Stress in MPa 

(What we have in our SIF)

On the choice of units

To give you an example: for ISMIP tests A-D, the value for the constants would be

- the density: $\rho = 910\ \mathrm{kg/m^3}$
- the gravity: $g \quad = 9.81\ \mathrm{m\ s^{-2}}$
- the fluidity: $A = 10^{-16}\ \mathrm{Pa^{-3}\ a^{-1}}$

	USI kg - m - s		kg - m - a		MPa - m - a	
g =	9.81	$\mathrm{m / s^2}$	9.7692E+15	$\mathrm{m / a^2}$	9.7692E+15	$\mathrm{m / a^2}$
ρ =	910	$\mathrm{kg / m^3}$	910	$\mathrm{kg / m^3}$	9.1380E-19	$\mathrm{MPa\ m^{-2}\ a^2}$
A =	3.1689E-24	$\mathrm{kg^{-3}\ m^3\ s^5}$	1.0126E-61	$\mathrm{kg^{-3}\ m^3\ a^5}$	100	$\mathrm{MPa^{-3}\ a^{-1}}$
η =	5.4037E+07	$\mathrm{kg\ m^{-1}\ s^{-5/3}}$	1.7029E+20	$\mathrm{kg\ m^{-1}\ a^{-5/3}}$	0.1710	$\mathrm{MPa\ a^{1/3}}$

The diagnostic problem

```
  ! the variable taken to evaluate the Arrhenius law
  ! in general this should be the temperature relative
  ! to pressure melting point. The suggestion below plugs
  ! in the correct value obtained with TemperateIceSolver
  !  Temperature Field Variable = String "Temp Homologous"
  ! the temperature to switch between the
  ! two regimes in the flow law
  Limit Temperature = Real -10.0
  ! In case there is no temperature variable (which here is the case)
  Constant Temperature = Real -3.0

  ! Heat transfer stuff (will come later)
  !Temp Heat Capacity = Variable Temp
  !  Real MATC "capacity(tx)*(31556926.0)^(2.0)"

  !Temp Heat Conductivity = Variable Temp
  !  Real MATC "conductivity(tx)*31556926.0*1.0E-06"

  !Temp Upper Limit = Variable Depth
  !     Real MATC "273.15 - 9.8E-08 * tx * 910.0 * 9.81" !-> this is the correct
ion of the presure melting point with respect to the hydrostatic overburden at t
he point
 End

Body Force 1
  Name = "BodyForce1"
  Heat Source = 1
  Flow BodyForce 1 = Real 0.0
  Flow BodyForce 2 = Real $-9.81 * (31556926.0)^(2.0)  !MPa - a - m
End
```

We set our glacier to be at -3 C

Now commented, needed later

Gravity, scaled to deliver results in m/a and MPa

The diagnostic problem

- Boundary conditions:
  - using array function for reading surfaces
  - **`Real [cubic]`** expects two columned row:

    $x_1 \quad z_1$

    $x_2 \quad z_2$

    …
  - **`include`** just inserts external file (length)
  - Right values interpolated by matching interval of left values for input variable

```
Boundary Condition 1
  Name = "bedrock"
  Target Boundaries = 1
  Conpute Normals = Logical True
! include the bedrock DEM, which has two colums
  Bottom Surface = Variable Coordinate 1
  Real cubic
     include  "steady_ELA400_bedrock.dat"
  End
  Velocity 1 = Real 0.0e0
  Velocity 2 = Real 0.0e0
End

Boundary Condition 2
  Name = "sides"
  Target Boundaries(2) = 3 4 ! combine left and right boundary
  Velocity 1 = Real 0.0e0
End

Boundary Condition 3
  Name = "surface"
  Target Boundaries = 2
! include the surface DEM, which has two colums
  Top Surface = Variable Coordinate 1
  Real cubic
     include  "steady_ELA400_surface.dat"
  End
  Depth = Real 0.0
End
```

The diagnostic problem

- Now, run the case:

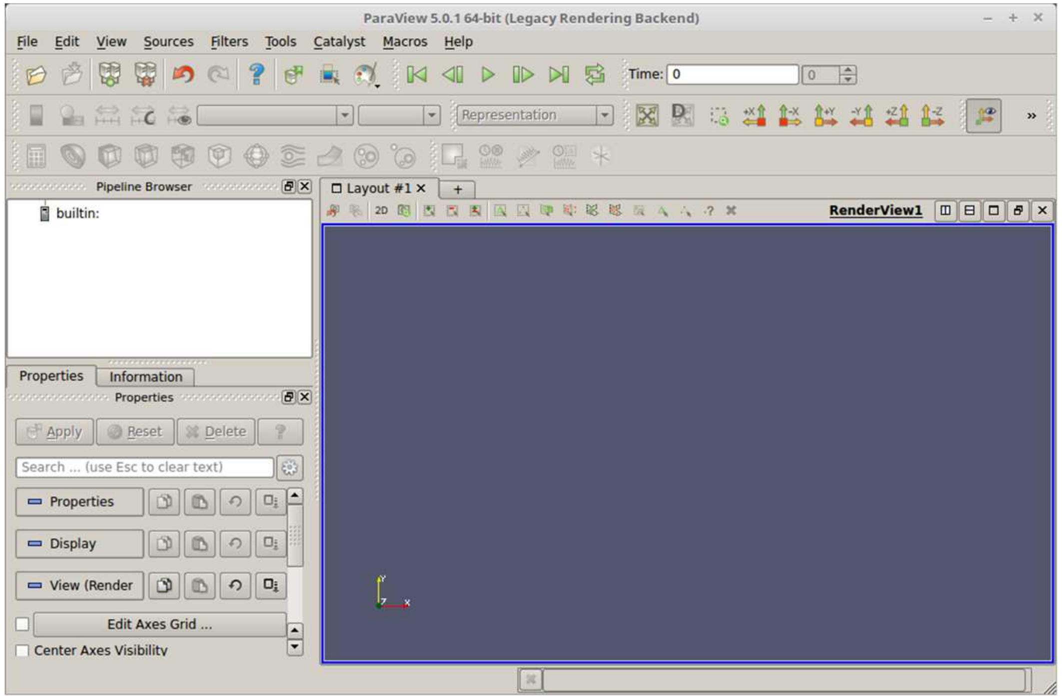
  `$ ElmerSolver Stokes_diagnostic.sif`

  - You will see the convergence history displayed:

```
FlowSolve: --------------------------------------
FlowSolve:  NAVIER-STOKES ITERATION          23
FlowSolve: --------------------------------------
FlowSolve:
FlowSolve: Starting Assembly...
FlowSolve: Assembly done
FlowSolve: Dirichlet conditions done
ComputeChange: NS (ITER=23) (NRM,RELC): (  1.6112696
0.90361030E-03 ) :: navier-stokes
FlowSolve: iter:   23 Assembly: (s)    0.26    6.04
FlowSolve: iter:   23 Solve:    (s)    0.11    2.62
FlowSolve:  Result Norm     :    1.6112695610649261
FlowSolve:  Relative Change :    9.0361030224648782E-004
```

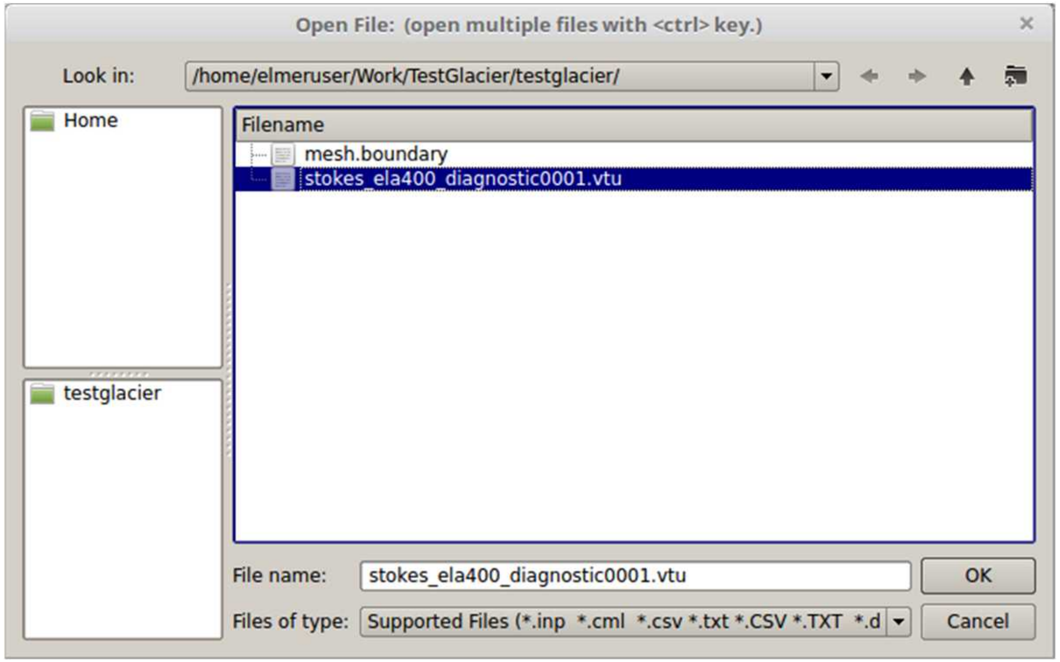
The diagnostic problem

- Post-processing using ParaView: `$ paraview`



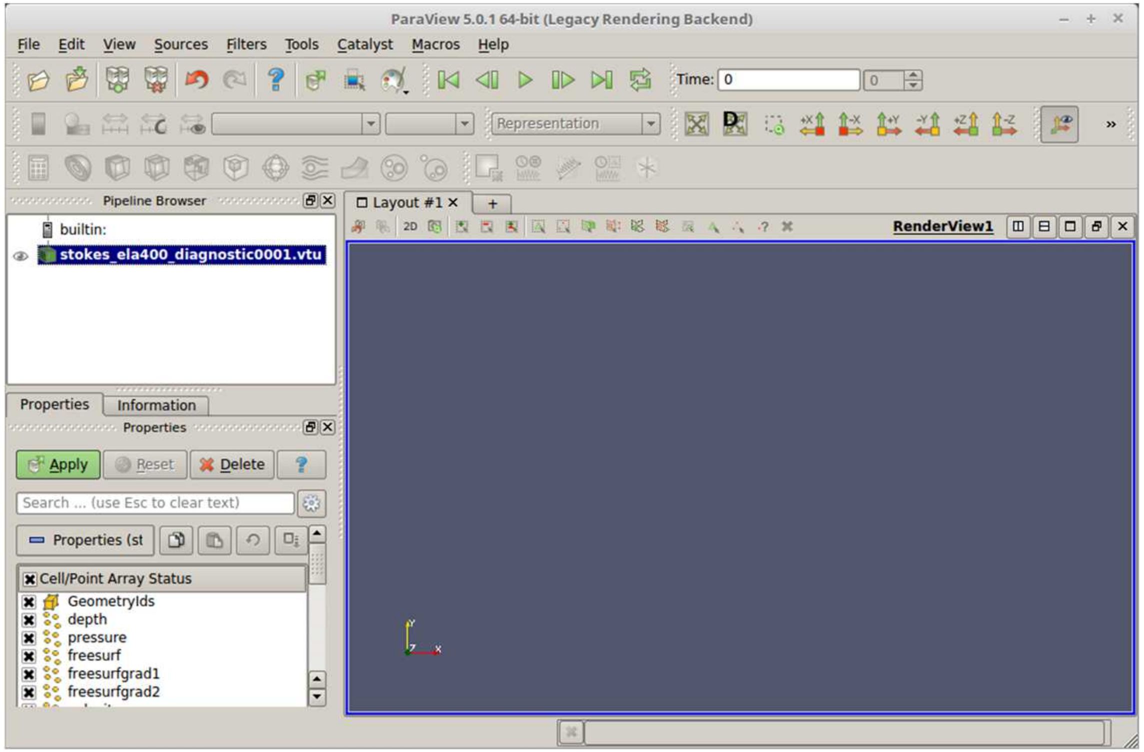
The diagnostic problem

- File → Open `stokes_ela400_diagnostic0001.vtu`



The diagnostic problem

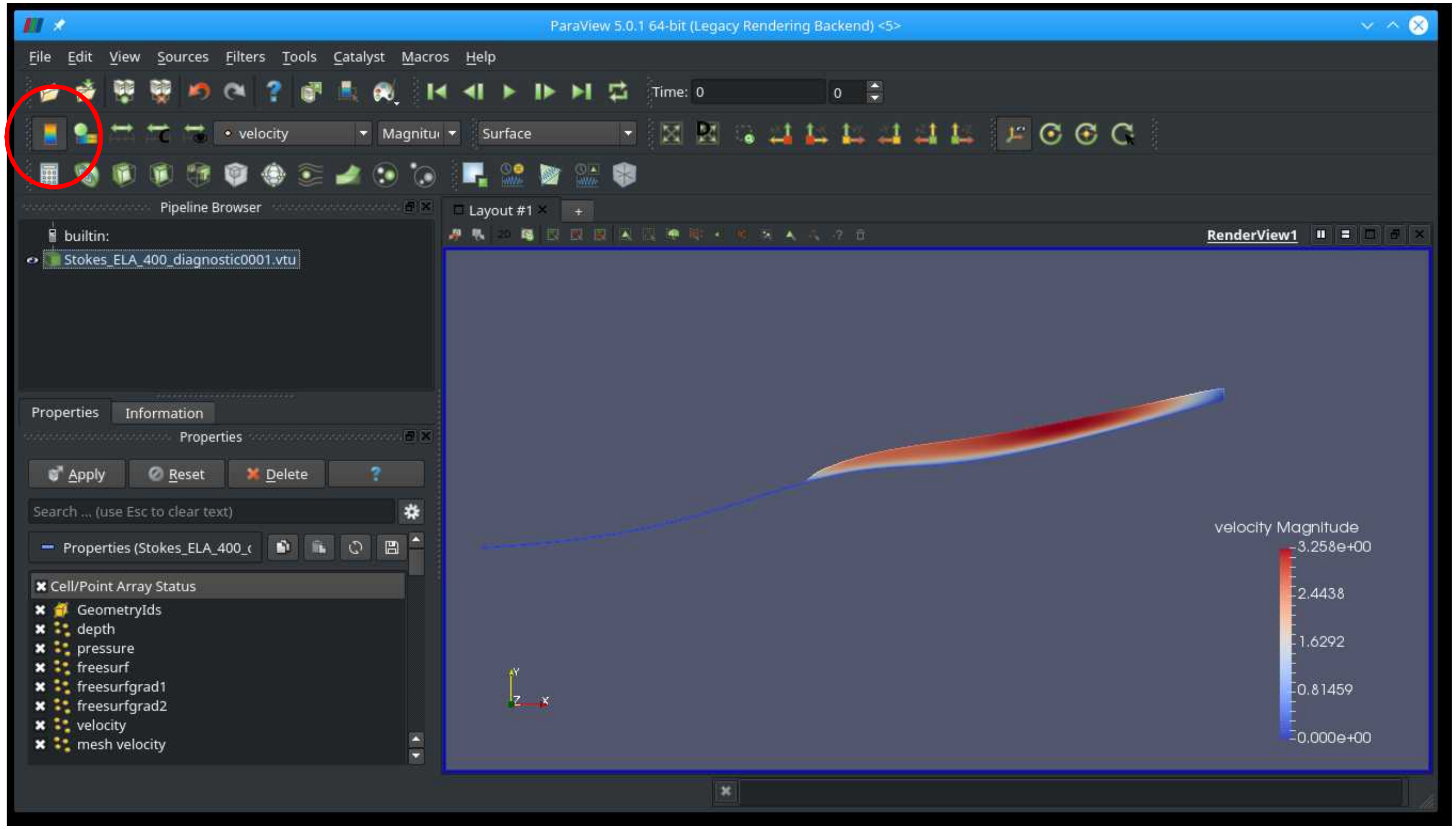
- Apply



The diagnostic problem

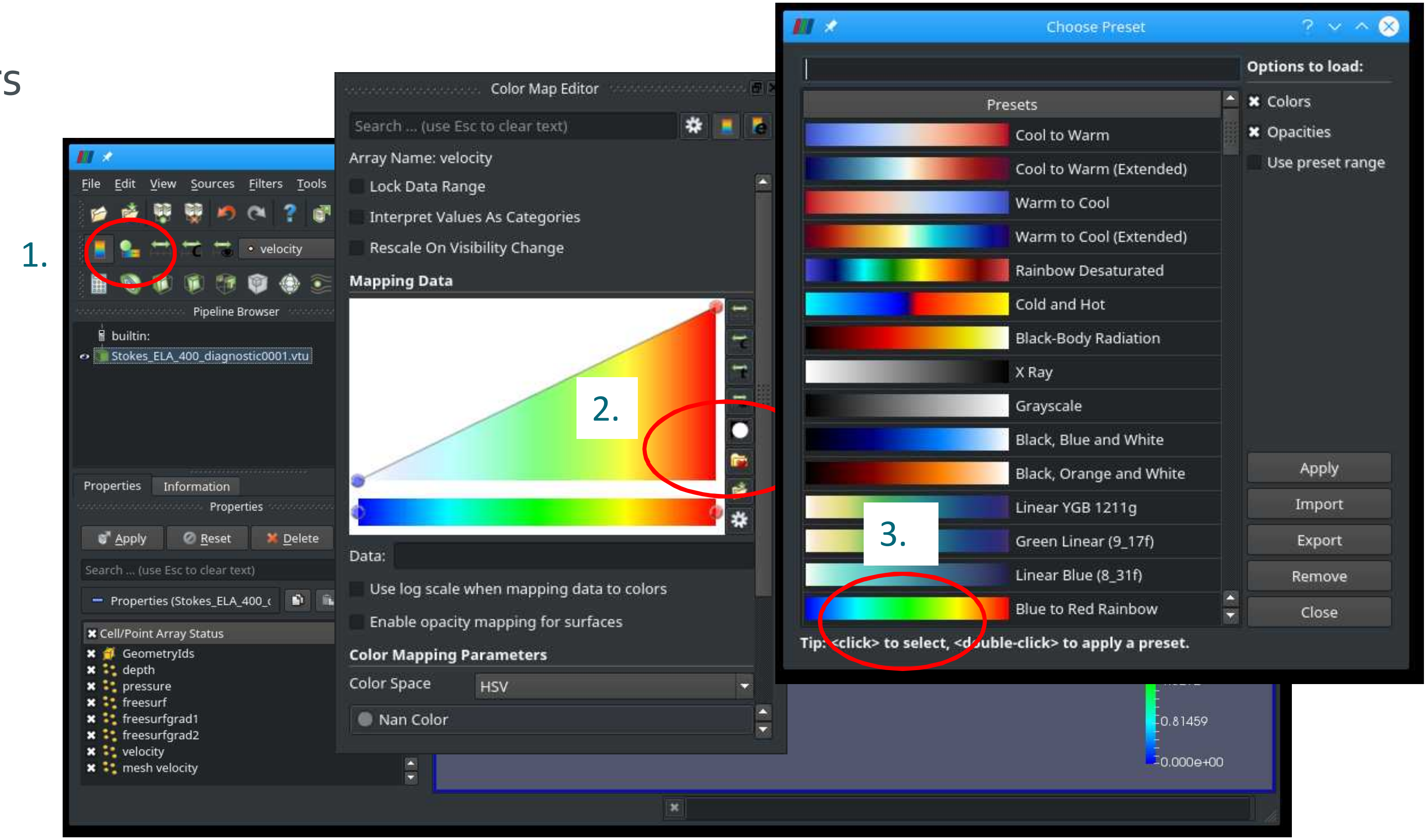
- Change to **velocity**

Press to activate colour bar



The diagnostic problem

- Change colours



Sliding

- Different sliding laws in Elmer
- Simplest: Linear Weertman $\boldsymbol{\tau} = \beta^2 \boldsymbol{u}$
  - This is formulated for the traction $\boldsymbol{\tau}$ and velocity $\boldsymbol{u}$ in tangential plane
- In order to define properties in normal-tangential coordinates:

```
Normal-Tangential Velocity = True
```

- β^{-2} is the `Slip Coefficient {2,3}` (for the tangential directions 2 and 3) (for 3D, in 2d only direction 2)
- Setting normal velocity to zero (no-penetration)

```
Velocity 1 = 0.0
```

Sliding

- Now we introduce sliding
  - We deploy a sliding zone between z=300 and 400m

```
Boundary Condition 1
  Name = "bedrock"
  Target Boundaries = 1
  Conpute Normals = Logical True
! include the bedrock DEM, which has two colums
  Bottom Surface = Variable Coordinate 1
  Real cubic
     include  "steady_ELA400_bedrock.dat"
  End
  Normal-Tangential Velocity = True
  Velocity 1 = Real 0.0e0
  Slip Coefficient 2 = Variable Coordinate 2
     Real MATC "(1.0 - (tx > 300.0)*(tx < 400.0))*1000.0 + 1.0/100.0"
End
```

Use normal-tangential coordinate system

Definition of slip Coefficient

Sliding

```
! Flow Depth still for postprocessing, only,
! now replaced by structured version
Solver 2
  Equation = "HeightDepth"
  Procedure = "StructuredProjectToPlane" "StructuredProjectToPlane"
  Active Coordinate = Integer 2
  Operator 1 = depth
  Operator 2 = height
End
```

Replace the **`FlowDepth`** Solver with this one. This solver simply uses the vertically structured mesh to inquire the Depth/Height without solving a PDE (much cheaper).

Sliding

- Restart from previous run (improved initial guess)

```
Simulation
  Max Output Level = 4
  Coordinate System = "Cartesian 2D"
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = "Steady"
  Steady State Max Iterations = 1
  Output Intervals = 1
  Output File = "Stokes_ELA400_diagnostic_slide.result"
  Post File = "Stokes_ELA400_diagnostic_slide.vtu"
  Initialize Dirichlet Conditions = Logical False
  ! Restart from previous run
  Restart File = "Stokes_ELA400_diagnostic.result"
  Restart Position = 0
End
```

Load the last entry in file

Sliding

- Now, run the case:

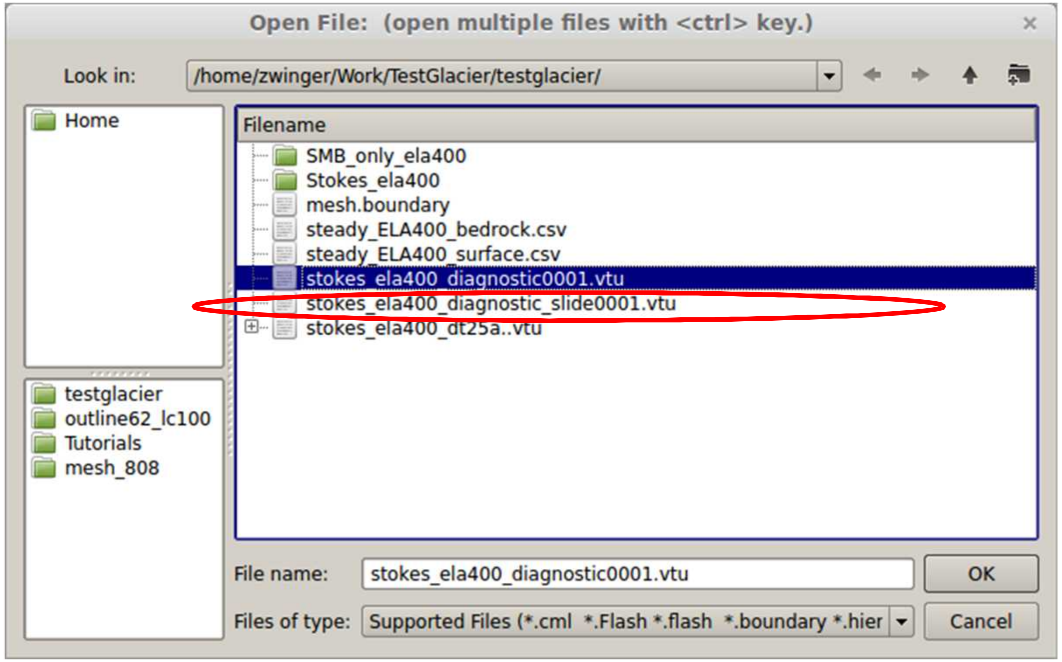
`$ ElmerSolver Stokes_diagnostic_slide.sif`

  - Converged much earlier:

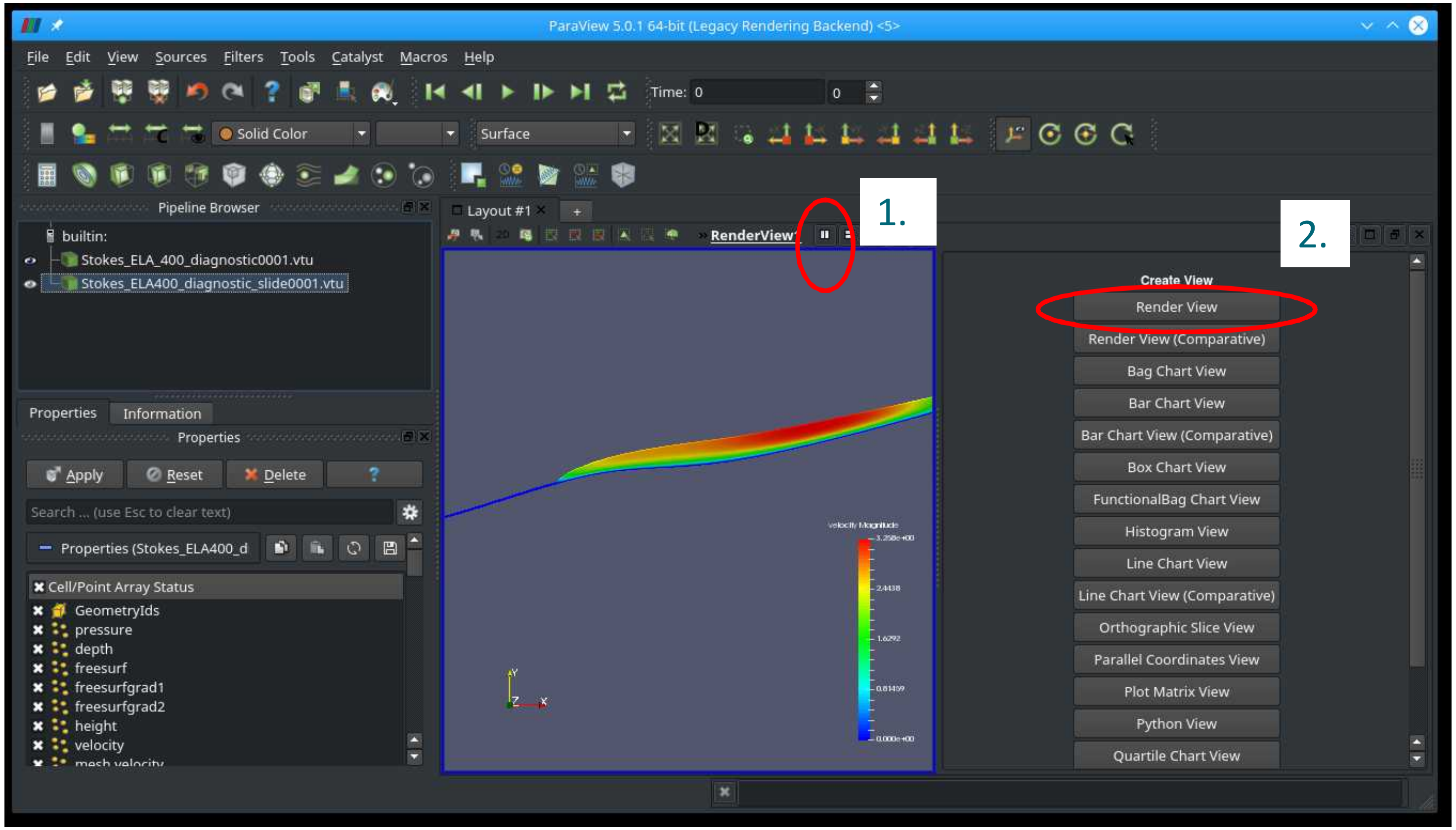
```
FlowSolve: -------------------------------------
FlowSolve:  NAVIER-STOKES ITERATION          12
FlowSolve: -------------------------------------
FlowSolve:
FlowSolve: Starting Assembly...
FlowSolve: Assembly done
FlowSolve: Dirichlet conditions done
ComputeChange: NS (ITER=12) (NRM,RELC): (  3.4915753
0.34732117E-05 ) :: navier-stokes
FlowSolve: iter:   12 Assembly: (s)    0.32    3.53
FlowSolve: iter:   12 Solve:    (s)    0.12    1.38
FlowSolve:  Result Norm      :    3.4915753430899730
FlowSolve:  Relative Change :    3.4732116934487441E-006
ComputeChange: SS (ITER=1) (NRM,RELC): (  3.4915753
2.0000000     ) :: navier-stokes
```

Sliding

- Load parallel to previous file
- **File → Open** `stokes_ela400_diagnostic_slide0001.vtu`



Sliding

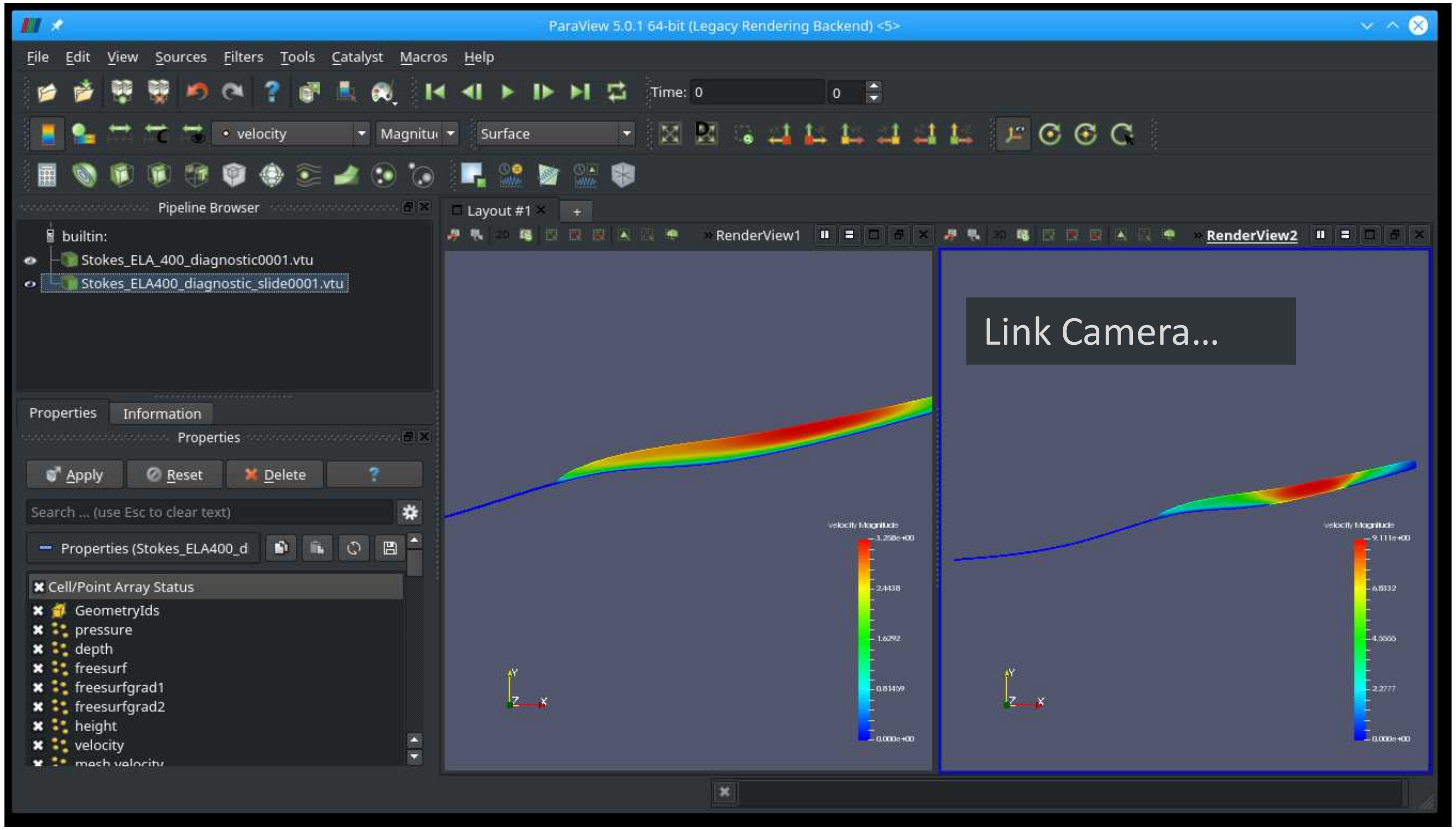


ParaView 5.0.1 64-bit (Legacy Rendering Backend) <5>
File Edit View Sources Filters Tools Catalyst Macros Help
Time: 0
Solid Color
Surface
Pipeline Browser
builtin:
Stokes_ELA_400_diagnostic0001.vtu
Stokes_ELA400_diagnostic_slide0001.vtu
Properties Information
Apply Reset Delete
Search ... (use Esc to clear text)
Properties (Stokes_ELA400_d
Cell/Point Array Status
GeometryIds
pressure
depth
freesurf
freesurfgrad1
freesurfgrad2
height
velocity
Layout #1
RenderView1
1.
2.
Create View
Render View
Render View (Comparative)
Bag Chart View
Bar Chart View
Bar Chart View (Comparative)
Box Chart View
FunctionalBag Chart View
Histogram View
Line Chart View
Line Chart View (Comparative)
Orthographic Slice View
Parallel Coordinates View
Plot Matrix View
Python View
Quartile Chart View

Sliding

Right click right window

Left click on left window

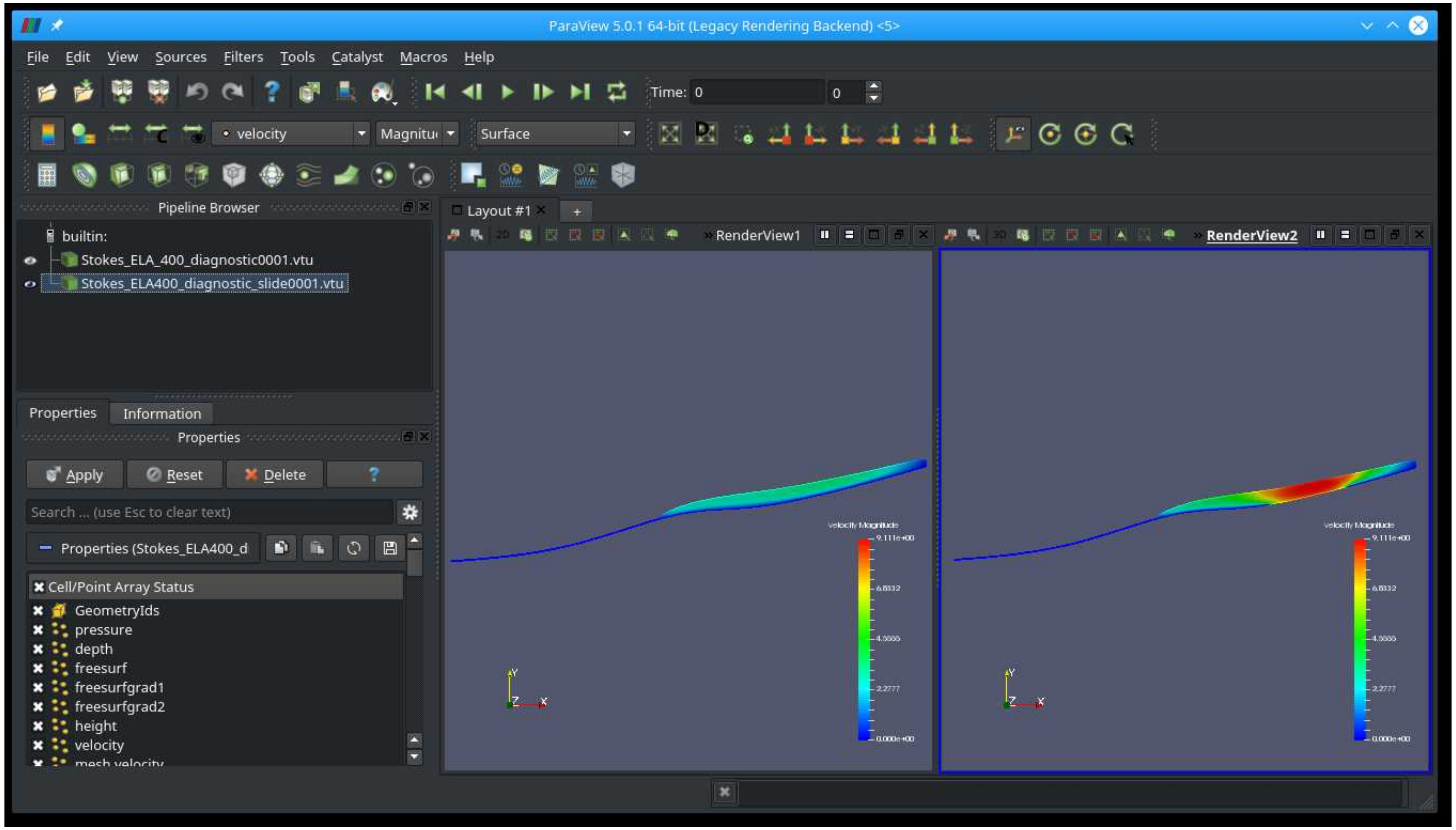


Sliding

Right click right window

Left click on left window

Scales velocity and syncs behaviour between windows



End of first session

What you should know by now:

- Basic diagnostic (= steady state with prescribed geometry) iso-thermal simulation
- Linear system, Non-linear system solution
- Iterative/direct solver
- Read-in of simple DEM, manipulation of initial mesh (structured)
- Using tabulated value interpolation
- Writing interpreted MATC function
- Basic Paraview post-processing

HEAT TRANSFER

Starting from the diagnostic setup of the previous session we:

- Compute the temperature for a given velocity field and boundary conditions
- Introduce heat transfer
- Account for pressure-melting point
- Add Thermo-mechanical coupling (viscosity-temperature)

Heat transfer

- Adding heat transfer to **`Stokes_diagnostic_slide.sif`**:
  - Add `ElmerIceSolvers TemperateIceSolver` with variable name `Temp` (see next slide)
  - Surface temperature distribution: linear from 273.15 K at z=0m to 263.15 K at z=1000m

```
Temp = Variable Coordinate 2
     Real
           0.0   273.15
        1000.0   263.15
     End
```

  - Geothermal heat flux of 200 mW m^{-2} at bedrock
  - Make sure you restart from `Stokes_ELA400_diagnostic_slide.result`

Heat transfer

```
Solver 5
  Equation = String "Homologous Temperature Equation"
  Procedure =  File "ElmerIceSolvers" "TemperateIceSolver"
  Variable = String "Temp"
  Variable DOFs = 1
  Stabilize = True
  Optimize Bandwidth = Logical True
  Linear System Solver = "Iterative"
  Linear System Direct Method = UMFPACK
  Linear System Convergence Tolerance = 1.0E-06
  Linear System Abort Not Converged = False
  Linear System Preconditioning = "ILU1"
  Linear System Residual Output = 0
  Nonlinear System Convergence Tolerance = 1.0E-05
  Nonlinear System Max Iterations = 100
  Nonlinear System Relaxation Factor = Real 9.999E-01
  Steady State Convergence Tolerance = 1.0E-04
End
```

Heat transfer

- Material parameters in Material section

```
Material 1
…
 ! Heat transfer stuff
  Temp Heat Capacity = Variable Temp
    Real MATC "capacity(tx)*(31556926.0)^(2.0)"

  Temp Heat Conductivity = Variable Temp
    Real MATC "conductivity(tx)*31556926.0*1.0E-06"
End
```

- Using defined MATC-functions for
  - Capacity: $$c(T) = 146.3 + (7.253 \cdot T[\mathrm{K}])$$
  - Conductivity: $$\kappa(T) = 9.828 \exp\left(-5.7 \times 10^{-3} \cdot T[\mathrm{K}]\right)$$

Heat transfer

- Material parameters in Material section

```
!! conductivity
$ function conductivity(T)  { _conductivity=9.828*exp(-5.7E-03*T)}
!! capacity
$ function capacity(T) { _capacity=146.3+(7.253*T)}
```

- Using defined MATC-functions for
  - Capacity: $$c(T) = 146.3 + (7.253 \cdot T[\mathrm{K}])$$
  - Conductivity: $$\kappa(T) = 9.828 \exp\left(-5.7 \times 10^{-3} \cdot T[\mathrm{K}]\right)$$

Heat transfer

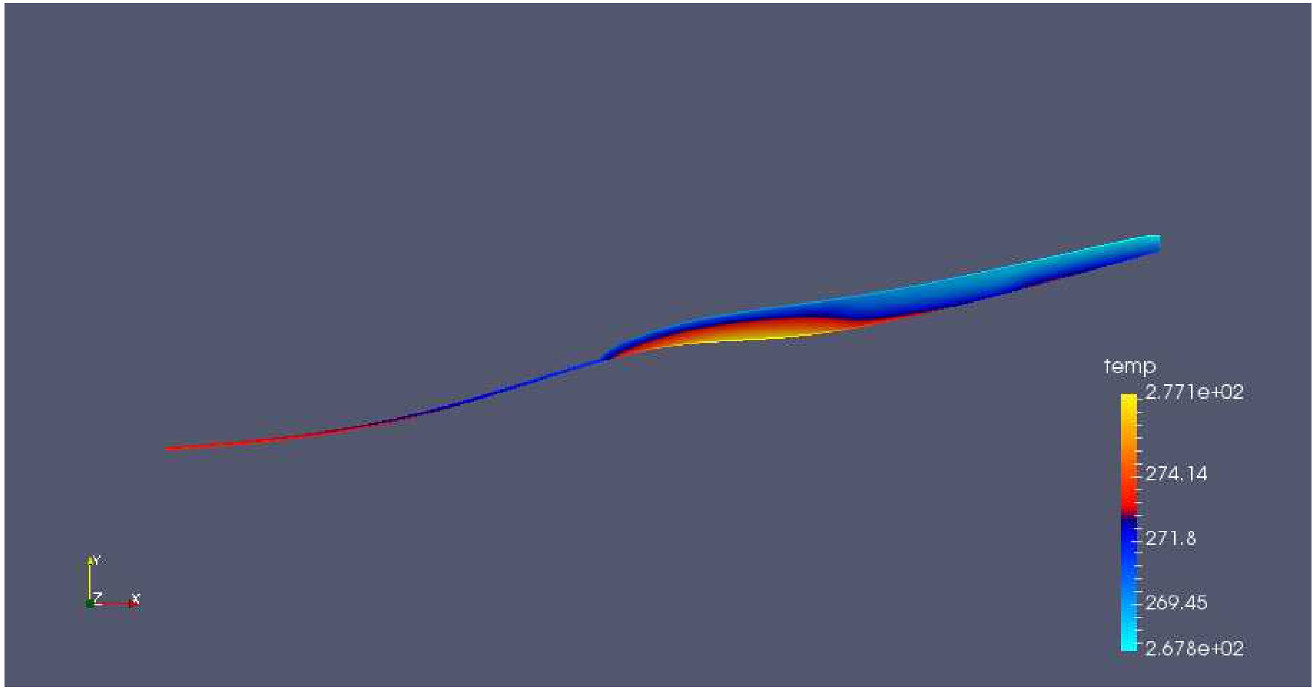
- Now, run the case:

`$ ElmerSolver Stokes_diagnostic_temp.sif`

- It goes pretty quick, as we only have <u>one-way coupling</u> and hence <u>don't even execute</u> the Stokes solver

```
Solver 3
  Exec Solver = "Never" ! we have a solution from previous case
  Equation = "Navier-Stokes"
```

Heat transfer



- Due to high geothermal heatflux we have areas above pressure melting point
- We have to account for this

Heat transfer

- Constrained heat transfer:
  - Including following lines in Solver section of `TemperateIceSolver`

```
! the contact algorithm (aka Dirichlet algorithm)
!-----------------------------------------------------
 Apply Dirichlet = Logical True
! those two variables are needed in order to store
! the relative or homologous temperature as well
! as the residual
!-----------------------------------------------
Exported Variable 1 = String "Temp Homologous"
Exported Variable 1 DOFs = 1
Exported Variable 2 = String "Temp Residual"
Exported Variable 2 DOFs = 1
```

Heat transfer

- Constrained heat transfer:
  - Also introduce the upper limit for the temperature (a.k.a. pressure melting point) in the Material section

```
Temp Upper Limit = Variable Depth
       Real MATC "273.15 - clausclap * tx * 910.0 * 9.81"
```

$$T_{\mathrm{pm}} = T_0 + \beta_{\mathrm{c}} p \qquad p \approx \rho_{\mathrm{ice}} g \, d$$

Heat transfer

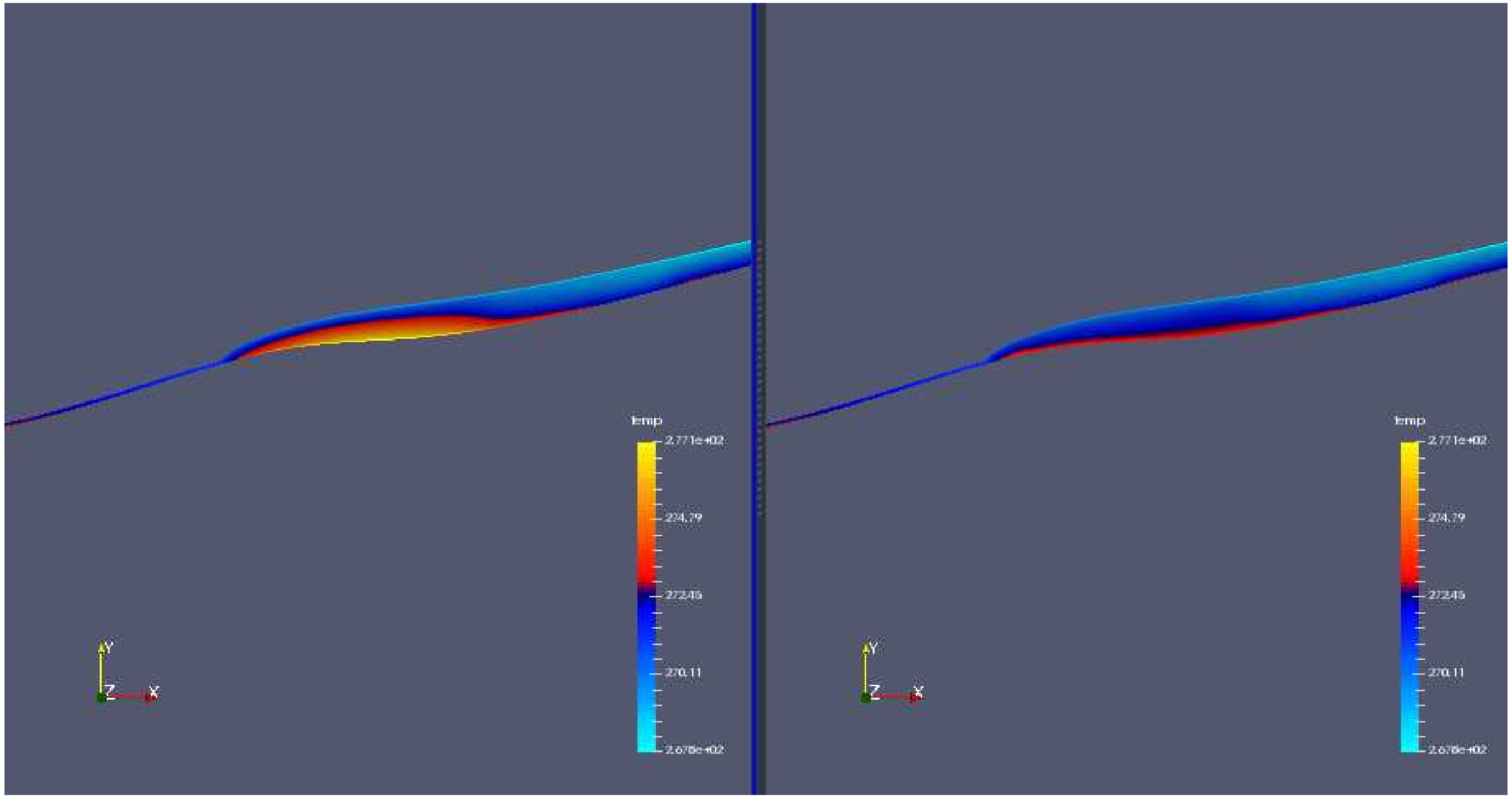
- Now, run the case:

```
$ ElmerSolver   \
  Stokes_diagnostic_temp_constrained.sif
```

- Already from the norm (~ averaged nodal values) it comes clear that values are in general now lower

```
TemperateIceSolver (temp): iter:    5 Assembly: (s)    1.36    6.77
TemperateIceSolver (temp): iter:    5 Solve:    (s)    0.00    0.01
TemperateIceSolver (temp):  Result Norm   :    271.78121462656480
TemperateIceSolver (temp):  Relative Change :
5.0215061382786350E-006
ComputeChange: SS (ITER=1) (NRM,RELC): (  271.78121      2.0000000
) :: homologous temperature equation
```

Heat transfer



Unconstrained

Constrained

Heat transfer

- **Thermo-mechanically coupled** simulation:
  - We have to iterate between Stokes and HTEq.

```
Steady State Max Iterations = 20
```

  - Coupling to viscosity in Material section

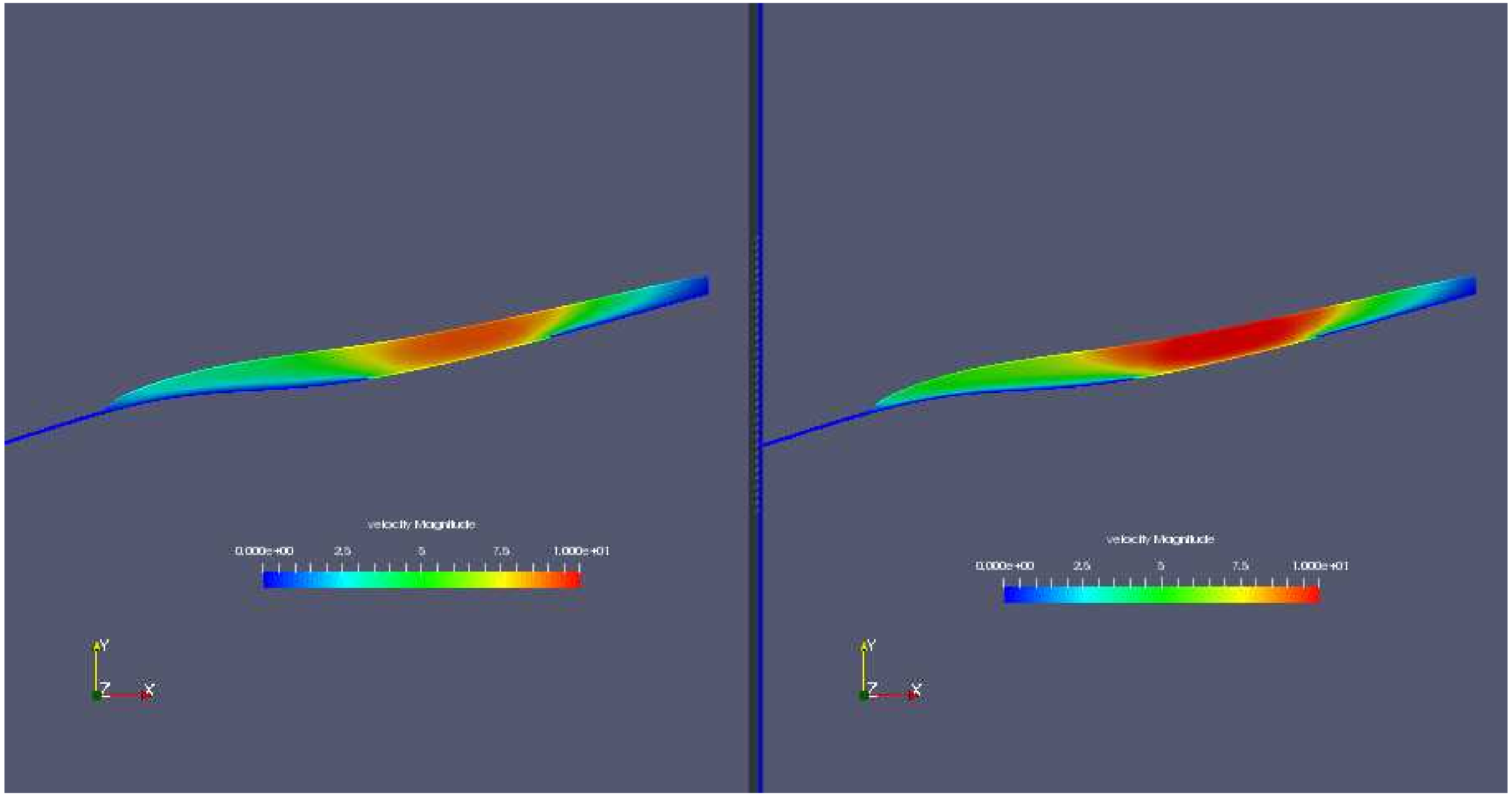
```
! the variable taken to evaluate the Arrhenius law
! in general this should be the temperature relative
! to pressure melting point. The suggestion below plugs
! in the correct value obtained with TemperateIceSolver
Temperature Field Variable = String "Temp Homologous"
```

Newton Iterations

- We need Picard (=fixed-point) iterations instead of Newton iterations at the beginning of each new non-linear iteration loop

```
Solver 1
!  Exec Solver = "Never"
  Equation = "Navier-Stokes"
…
  Nonlinear System Reset Newton = Logical True
  !Nonlinear System Relaxation Factor = 0.75
End
```

Heat transfer



Uncoupled

Thermo-mechanically coupled

End of third session

What you should know on top of the previous session:

- Basic diagnostic (= steady state with prescribed geometry) simulation including heat transfer equation (HTEq)
- Introduction of constraint (pressure-melting) into HTEq
- Thermo-mechanically coupled system

PROGNOSTIC RUN

- Starting from a deglaciated situation we show
- How to move to a transient run, i.e., introduce the
  - Free surface solution
  - Including coupling to climate via prescribing an accumulation/ablation function
- How to write a less simple MATC function
- How to write a (faster than MATC) Lua function

The prognostic problem

- Glacier with ~11 deg constant inclination
- Standard accumulation/ablation function

$$a(z) = \lambda\, z + a(z=0)$$

- Or in terms of ELA (equilibrium line altitude):

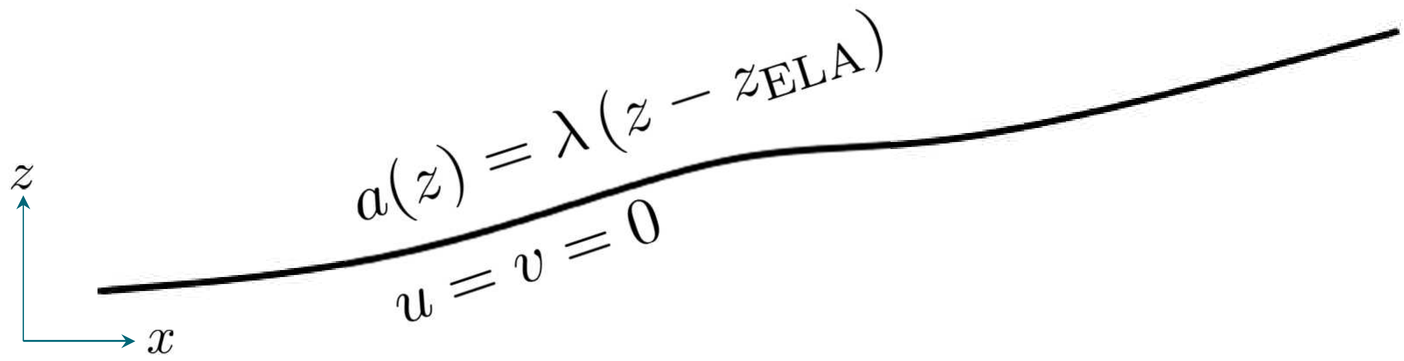
$$a_{\mathrm{ELA}} = \lambda\, z_{\mathrm{ELA}} + a_0 = 0$$

- We know lapserate, λ , and z_{ELA} and have to define $a_0 = -\lambda\, z_{\mathrm{ELA}}$

The Problem

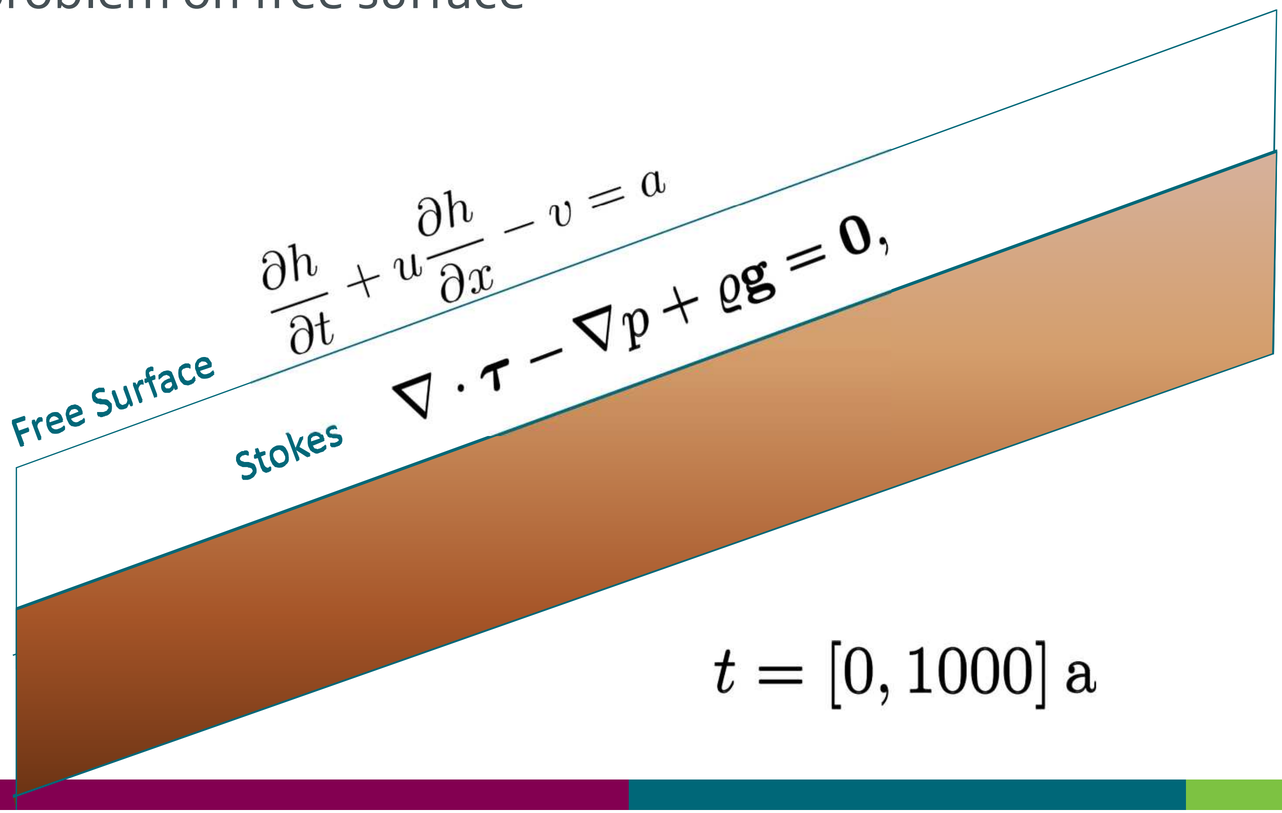
- From x=[0 :2500], z=[0:500]
- Setting mesh with 10 vertical levels with 5m flow depth

$$\lambda = 11/2500\,(\mathrm{m/a})\,\mathrm{m}^{-1}$$
$$z_{\mathrm{ELA}} = 400\,\mathrm{m}$$



The Problem

- Flow problem (Navier-Stokes) in ice
- Free-surface problem on free surface



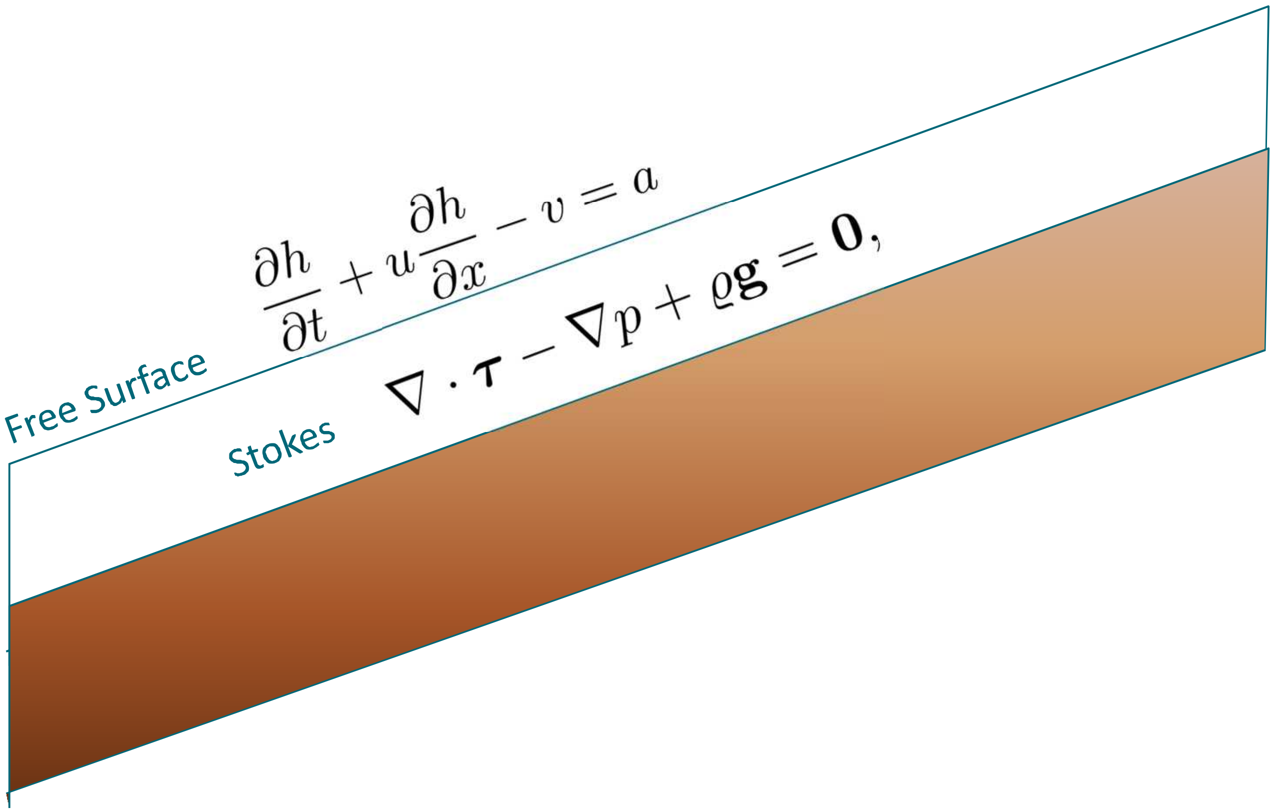
Time Stepping

```
Simulation
  Max Output Level = 4
  Coordinate System = "Cartesian 2D"
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = "Transient"
  Steady State Max Iterations = 1
  Timestepping Method = "BDF"
  BDF Order = 1
  Timestep Sizes = 10.0
  Timestep Intervals =  200
  Output Intervals = 10
  !Output File = "Stokes_prognostic_ELA400_SMB.result"
  !Post File = "Stokes_prognostic_ELA400_SMB_noflow.vtu"
  Post File = "Stokes_prognostic_ELA400_SMB_flow.vtu"
  Initialize Dirichlet Conditions = Logical False
End
```

Free Surface Equation

- Free surface equation is only run on - surprise! – the free surface
- Which renders it a lower-dimensional problem
- We need to declare a new(2nd) body on this surface

```
Boundary Condition 3
  Name = "surface"
  Top Surface = Equals "Zs"
  Target Boundaries = 2
  Body ID = 2 !!! THIS IS ESSENTIAL
End
```



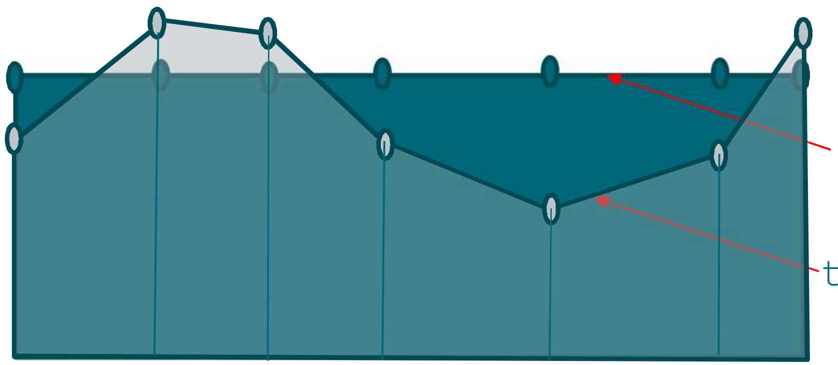
Free Surface Equation

```
Body 1
  Name = "Glacier"
  Body Force = 1
  Equation = 1
  Material = 1
  Initial Condition = 1
End

Body 2
  Name = "Surface"
  Body Force = 2
  Equation = 2
  Material = 2
  Initial Condition = 2
End
```

```
Equation 1
  Name = "Equation1"
  Convection = "computed"
  Flow Solution Name = String "Flow Solution"
  Active Solvers(3) = 1 2 3
End

Equation 2
  Name = "Equation2"
  !Convection = "none"
  Convection = "computed" !!! CHANGE TO THIS ONE
  Active Solvers(1) = 4
  Flow Solution Name = String "Flow Solution"
End
```

Free Surface Equation

```
Boundary Condition 3
  Name = "surface"
  Top Surface = Equals "Zs"
  Target Boundaries = 2
  Body ID = 2 !!! THIS IS ESSENTIAL
End
```



Free Surface Equation

- Starting with same values for both variables set to the bedrock shape of the diagnostic example

```
Initial Condition 2
  Zs =   Variable Coordinate 1
    Real cubic
     include  "steady_ELA400_bedrock.dat"
    End
  RefZs = Variable Coordinate 1
    Real cubic
     include  "steady_ELA400_bedrock.dat"
    End
End
```

- Using the latter to keep minimal height (in Material)

```
Material 2
  Min Zs = Variable RefZs
    Real MATC "tx - 0.1"
  Max Zs = Variable RefZs
    Real MATC "tx + 600.0"
End
```

Free Surface Equation

- And here comes the coupling to climate (as a general MATC function)

```
Body Force 2
  Name = "Climate"
  Zs Accumulation Flux 1 = Real 0.0e0
  Zs Accumulation Flux 2 = Variable Coordinate 1, Coordinate 2
    Real MATC "accum(tx)"
End
```

```
$ function accum(X) {\
  lapserate = (11.0/2750.0);\
  ela = 400.0;\
  asl = -ela*lapserate;\
  if (X(0) > 2500)\
    {_accum = 0.0;}\
  else\
   { _accum = lapserate*X(1) + asl;}\
}
```

$$a(z) = \lambda\,(z - z_{\mathrm{ELA}})$$

$$\lambda = 11/2500\,(\mathrm{m/a})\,\mathrm{m}^{-1}$$

$$z_{\mathrm{ELA}} = 400\,\mathrm{m}$$

Free Surface Equation

```
Solver 4
   Exec Solver = always
   Equation = "Free Surface"
   Variable = String "Zs"
   Variable DOFs =  1
   ! needed for evaluating the contact pressure
   Exported Variable 1 = -dofs 1 "Zs Residual"
   ! needed for storing the initial shape (needed for updates)
   Exported Variable 2 = -dofs 1 "RefZs"
   Procedure = "FreeSurfaceSolver" "FreeSurfaceSolver"
  ! This would take the contrained points out of solution
  ! Use in serial run, only
  ! Before Linsolve = "EliminateDirichlet" "EliminateDirichlet"
   Linear System Solver = Iterative
   Linear System Max Iterations = 1500
   Linear System Iterative Method = BiCGStab
   Linear System Preconditioning = ILU0
   Linear System Convergence Tolerance = Real 1.0e-7
   Linear System Abort Not Converged = False
   Linear System Residual Output = 1
   Nonlinear System Max Iterations = 100
   Nonlinear System Convergence Tolerance  = 1.0e-6
   Nonlinear System Relaxation Factor = 0.60
   Steady State Convergence Tolerance = 1.0e-03
   Stabilization Method = Bubbles
   ! Apply contact problem
   Apply Dirichlet = Logical True
End
```

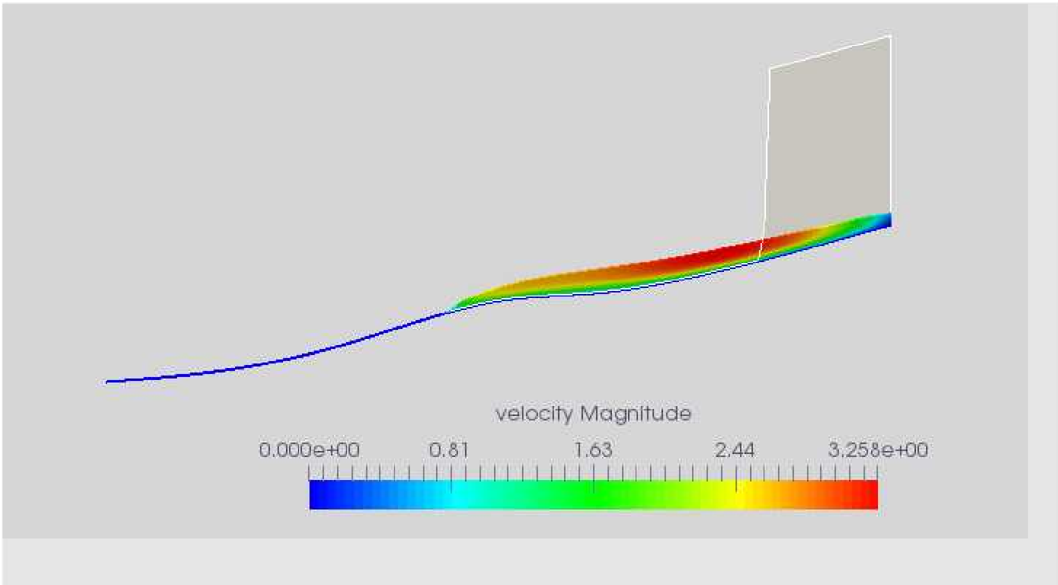
Passive elements

- We further switch the (Navier-)Stokes solution to passive in regions with flow-depth below threshold
- This usually brings more stable ice-fronts (uncomment to see difference)

```
Body Force 2
  Name = "Climate"
  Zs Accumulation Flux 1 = Real 0.0e0
  Zs Accumulation Flux 2 = Variable Coordinate 1, Coordinate 2
    Real MATC "accum(tx)"
End
```

The Solution

- Starting with no-flow problem, i.e., only surface mass balance, simply by setting Convection = “none” and (saves time) not executing Navier-Stokes, compare to run with coupled flow
- $ **ElmerSolver Stokes_prognostic.sif**



LUA – the faster alternative to MATC

- Similar syntax than MATC, but much faster

```
Body Force 2
  Name = "Climate"
  Zs Accumulation Flux 1 = Real 0.0e0
  Zs Accumulation Flux 2 = Variable Coordinate 1, Coordinate 2
  Real lua "accum(tx[0],tx[1])"
End
```

```
!---LUA BEGIN
! -- this is our accumulation rate
! function accum(X,Z)
!  if (X > 2500) then
!    return 0.0
!  else
!    return 11.0*Z/2750 - 400.0*11.0/2750.0
!  end
!end
!---LUA END
```

End of fourth session

What you should know on top of previous sessions:

- Basic prognostic (= time dependent with prescribed surface mass balance) simulation
- Introduced passive elements
- Introduced general MATC function to prescribe accumulation/ablation function
- Introduced general LUA function to prescribe accumulation/ablation function

USER DEFINED FUNCTION

In a follow-up session, by changing the previous setup we show:

- How to write, compile and include a self-written user defined function
- How to introduce time changing variables

User Defined Function

- Replace the MATC/Lua function with a user defined function (UDF)

All UDF's have the same header in Elmer(/Ice)

```
FUNCTION getAccumulation(  Model, Node, InputArray) RESULT(accum)
  ! provides you with most Elmer functionality
  USE DefUtils
  ! saves you from stupid errors
  IMPLICIT NONE
  ! the external variables
  !----------------------------------------------------------------------------
  TYPE(Model_t) :: Model     ! the access point to everything about the model
  INTEGER :: Node            ! the current Node number
  REAL(KIND=dp) :: InputArray(2) ! Contains the arguments passed to the function
  REAL(KIND=dp) :: accum      ! the result
```

User Defined Function

```
!------------------------------------------------------------------------------
! internal variables
!------------------------------------------------------------------------------
REAL(KIND=dp) :: lapserate, ela0, dElaDt, elaT, accumulationAtSl,&
     inittime, time, elevation, cutoff, offset
LOGICAL :: FirstTime=.TRUE.
! Remember this value
SAVE FirstTime, inittime

! lets hard-code our values (if we have time we can later make them being read from SIF)
lapserate = 11.0_dp/2750.0_dp
ela0 = 400.0_dp
dElaDt = -0.05_dp
cutoff = 600.0_dp
offset = 1500.0

! copy input (should match the arguments!)
elevation = InputArray(1)
time = InputArray(2)
WRITE (Message, '(A,E10.2,A,E10.2)')  "elevation=", elevation, "time=", time
CALL INFO("getAccumulation", Message, Level=9)
```

User Defined Function

```
! store the initial time, to be sure to have relative times
IF (FirstTime) THEN
   inittime = time
   FirstTime = .FALSE.
END IF


! get change of ELA with time
IF (time > offset) THEN
   elaT = ela0 - dElaDt * (time - offset)
ELSE
   elaT = ela0
END IF

! lets do the math
accumulationAtSl = -elaT*lapserate
IF (elevation > cutoff) elevation = cutoff
accum = lapserate*elevation + accumulationAtSl

RETURN

END FUNCTION getAccumulation
```

User Defined Function

The body-force section changes to:

```
Body Force 2
  Name = "Climate"
  Zs Accumulation Flux 1 = Real 0.0e0
  Zs Accumulation Flux 2 = Variable Coordinate 2, Time
    Real Procedure "accumulation" "getAccumulation"
End
```

Compilation is done with:

```
$ elmerf90 accumulation.f90 -o accumulation.so
```

Speedup MATC-LUA-UDF

==> MATC.log <==

SOLVER TOTAL TIME(CPU,REAL): 1027.24 **1080.45**

ELMER SOLVER FINISHED AT: 2020/10/31 23:42:46

==> LUA.log <==

SOLVER TOTAL TIME(CPU,REAL): 458.81 **471.08**

ELMER SOLVER FINISHED AT: 2020/10/31 23:34:12

==> UDF.log <==

SOLVER TOTAL TIME(CPU,REAL): 434.00 **446.95**

ELMER SOLVER FINISHED AT: 2020/11/01 00:01:21

DON'T USE MATC in performance critical parts

Lua almost as fast as compiled code

End of second session

What you should know on top of previous sessions:

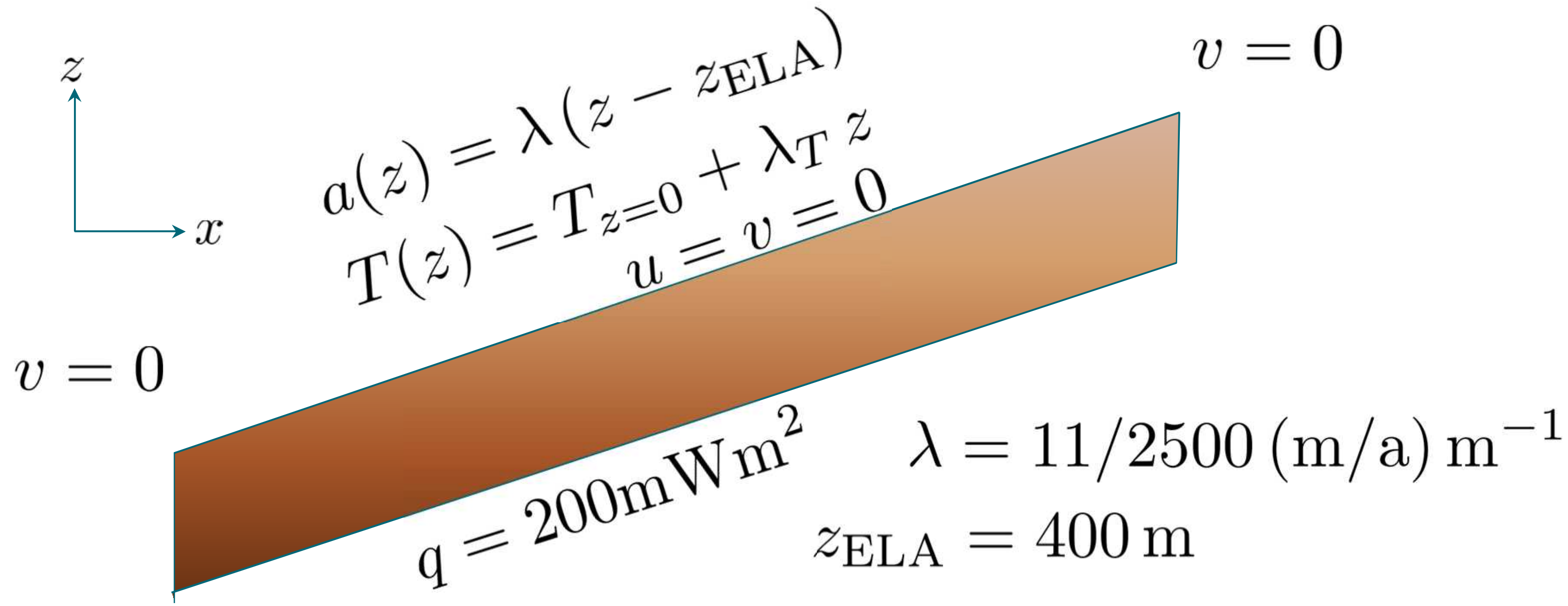
- Replacing (usually slow) MATC function by a compiled Fortran User Defined Function (UDF)

For those, who want to go continue …

EXERCISE

Exercise

- If time permits, lets put all things together and make a thermo-mechanically coupled prognostic run. What do we need to add?



Creating a mesh

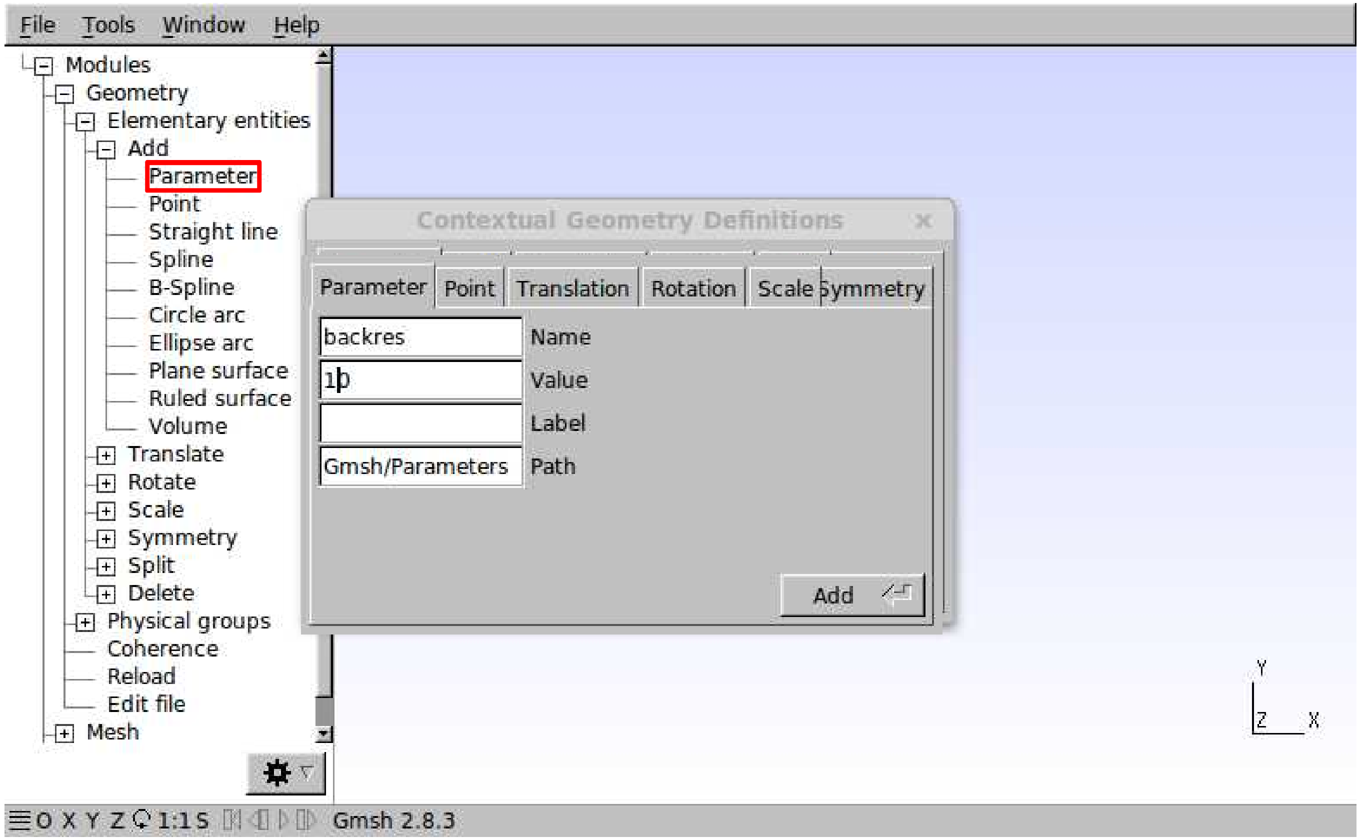
This is additional information on how to create the simple mesh for this run using Gmsh for people to try on their own

Be aware that in the previous example actually we chose a flat mesh which we morphed to the shape of the bedrock, here we are directly producing the bedrock shape in first instance

The Mesh

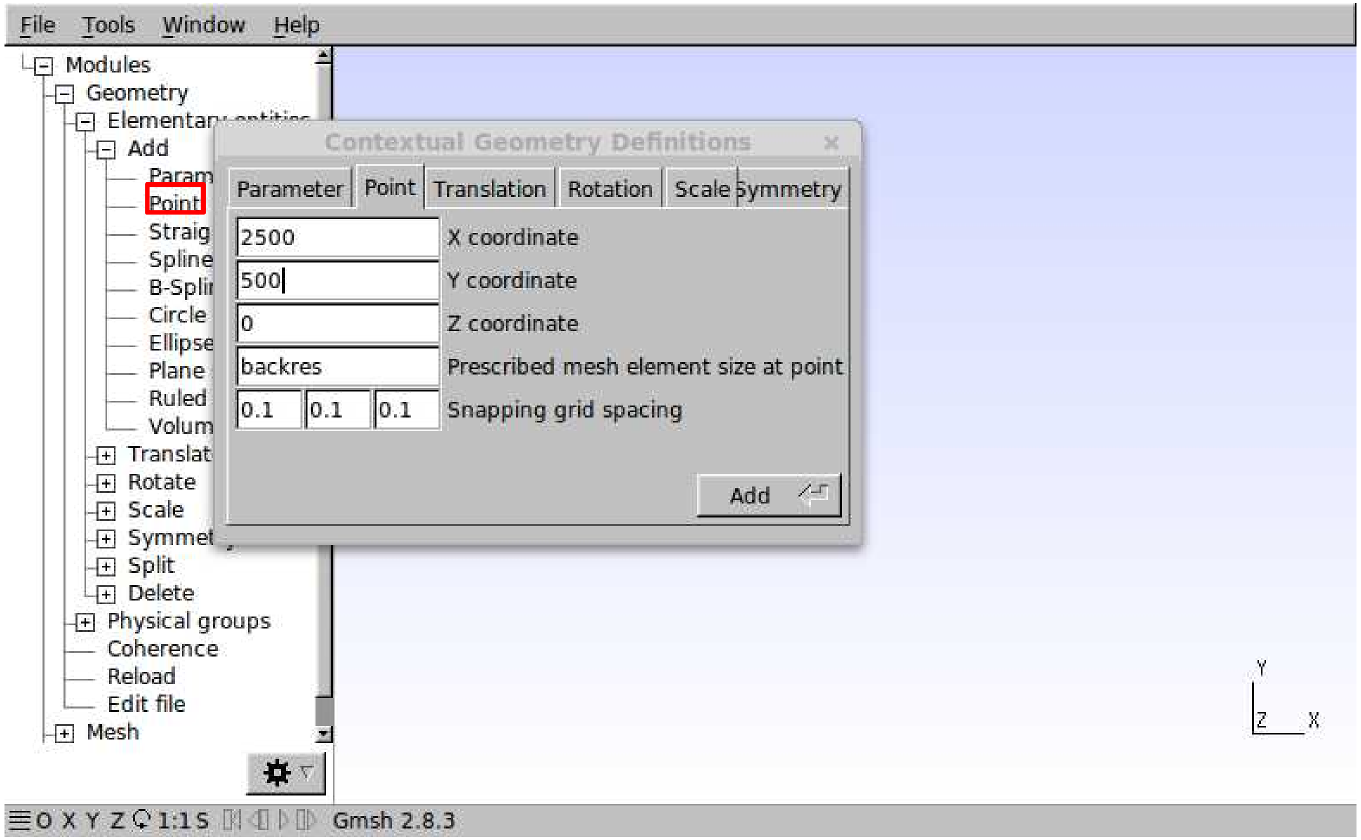
- Using Gmsh
- Simply launch by:
- `$ gmsh testglacier.geo &`
  - Don’t use the existing one in the **`Solution`**-folder, since we want to keep it as a backup, should this one fail

The Mesh



File Tools Window Help
Modules
Geometry
Elementary entities
Add
Parameter
Point
Straight line
Spline
B-Spline
Circle arc
Ellipse arc
Plane surface
Ruled surface
Volume
Translate
Rotate
Scale
Symmetry
Split
Delete
Physical groups
Coherence
Reload
Edit file
Mesh
Contextual Geometry Definitions
Parameter Point Translation Rotation Scale Symmetry
backres
Name
10
Value
Label
Gmsh/Parameters
Path
Add
Y
Z
X
O X Y Z 1:1 S
Gmsh 2.8.3

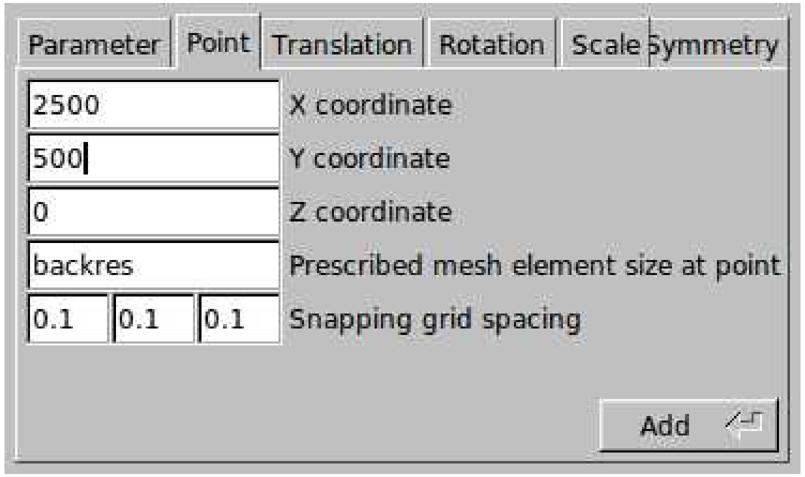
The Mesh


File Tools Window Help
Modules
Geometry
Elementary entities
Add
Param
Point
Straig
Spline
B-Splir
Circle
Ellipse
Plane
Ruled
Volum
Translat
Rotate
Scale
Symmet
Split
Delete
Physical groups
Coherence
Reload
Edit file
Mesh
Contextual Geometry Definitions
Parameter
Point
Translation
Rotation
Scale
Symmetry
2500
X coordinate
500
Y coordinate
0
Z coordinate
backres
Prescribed mesh element size at point
0.1
0.1
0.1
Snapping grid spacing
Add
Y
Z
X
O X Y Z 1:1 S
Gmsh 2.8.3

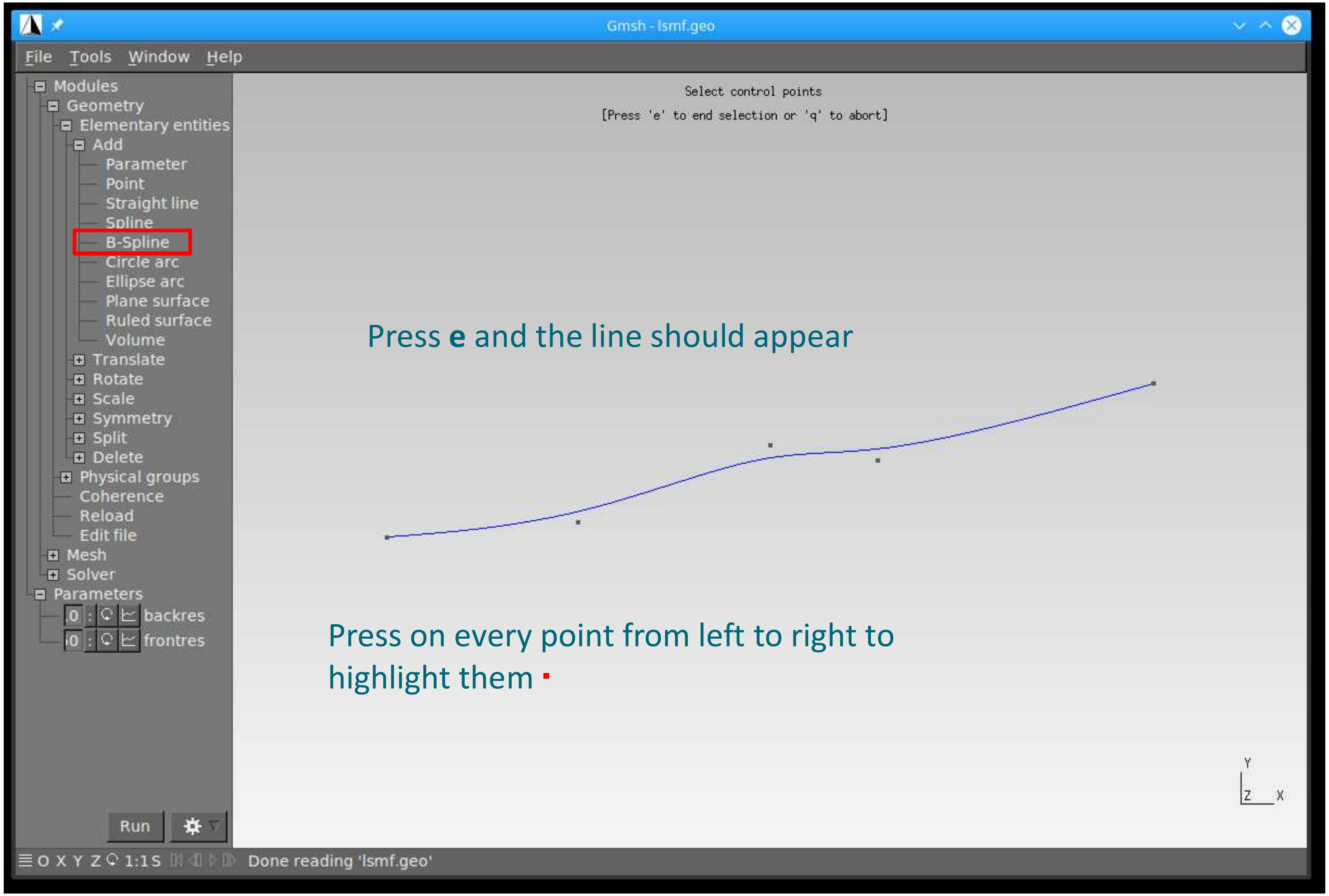
The Mesh

- Do that for any further points

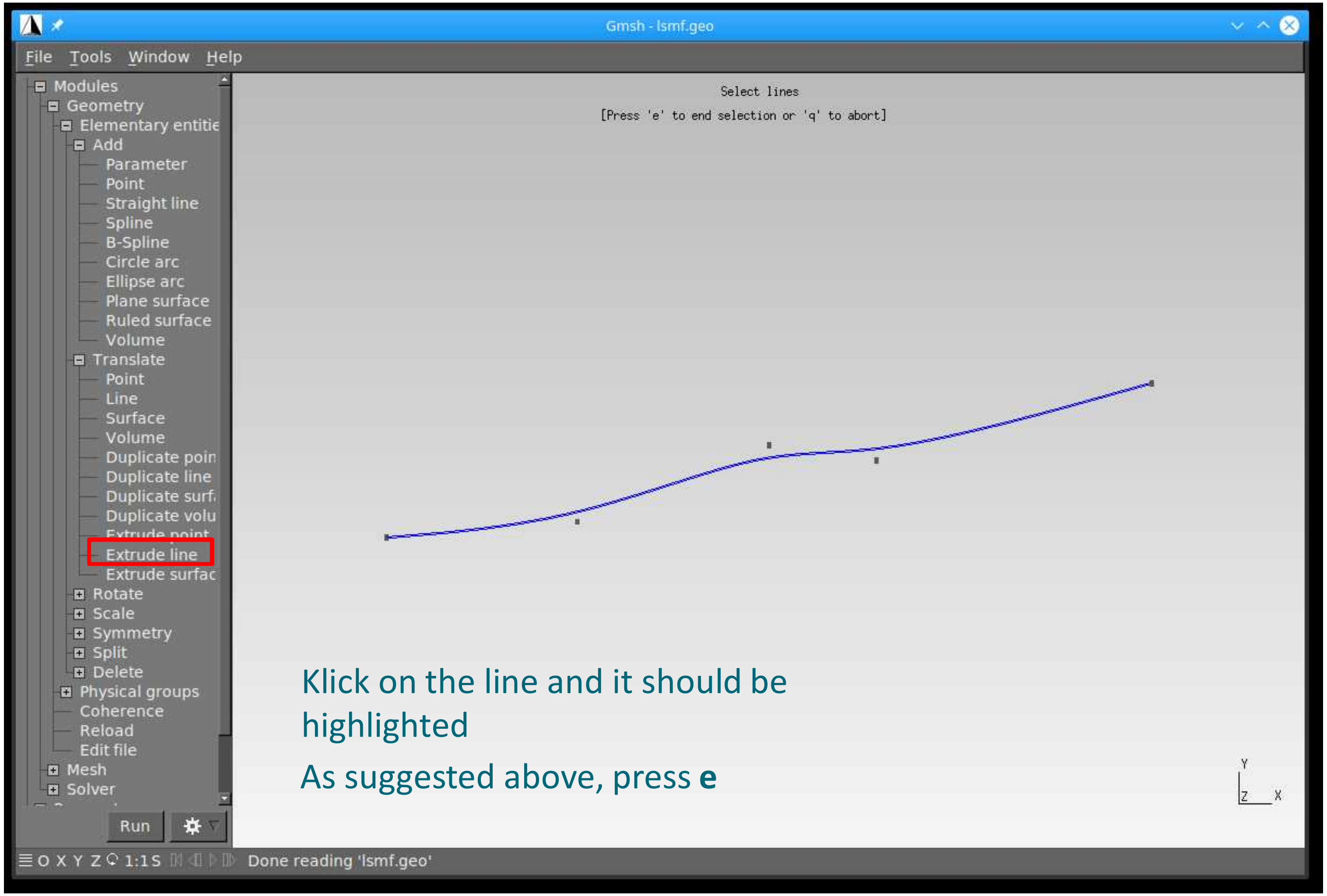
```
Point(1) = {2500, 500, 0, backres};
Point(2) = {0, 0, 0, frontres};
Point(3) = {625, 50, 0, frontres};
Point(4) = {1250, 300, 0, backres};
Point(5) = {1600, 250, 0, backres};
```



The Mesh


Gmsh - lsmf.geo
File Tools Window Help
Modules
Geometry
Elementary entities
Add
Parameter
Point
Straight line
Spline
B-Spline
Circle arc
Ellipse arc
Plane surface
Ruled surface
Volume
Translate
Rotate
Scale
Symmetry
Split
Delete
Physical groups
Coherence
Reload
Edit file
Mesh
Solver
Parameters
backres
frontres
Run
Select control points
[Press 'e' to end selection or 'q' to abort]
Press **e** and the line should appear
Press on every point from left to right to highlight them
Done reading 'lsmf.geo'

The Mesh



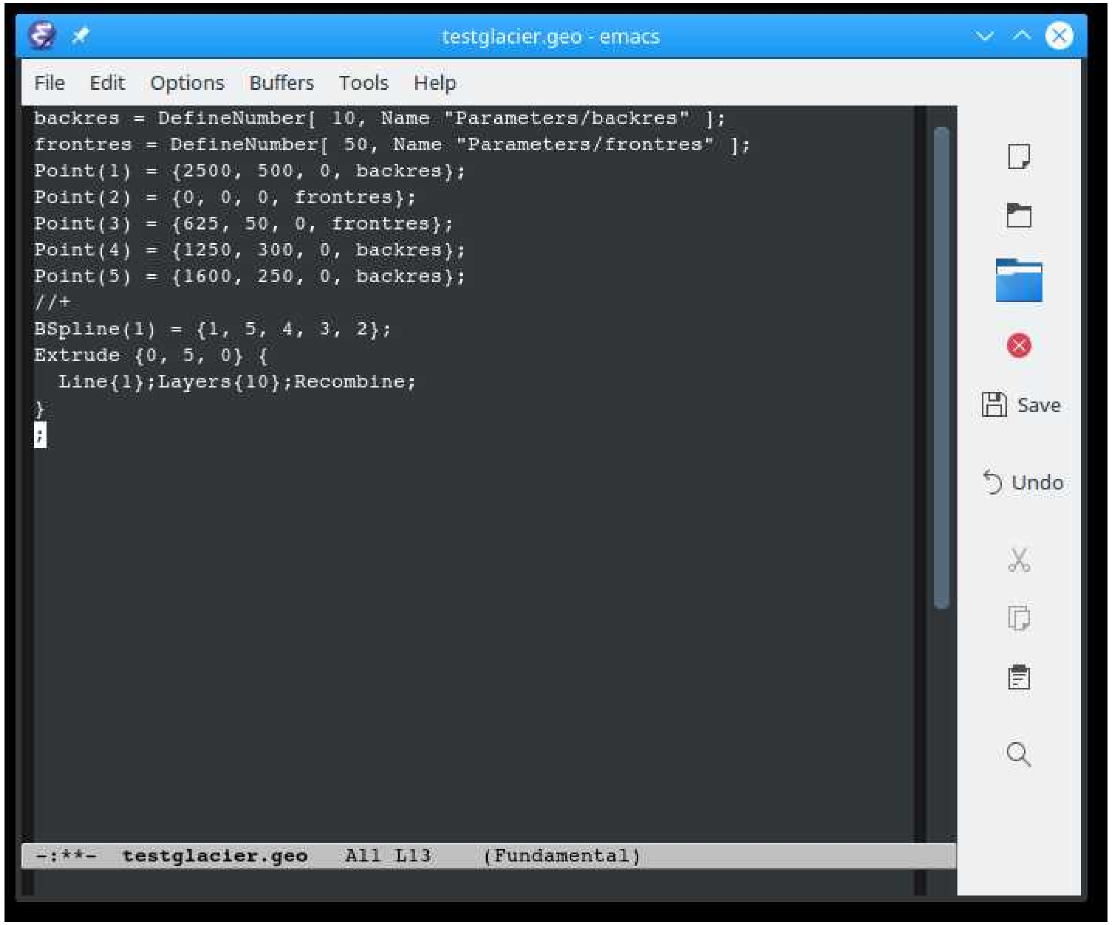
Klick on the line and it should be highlighted

As suggested above, press **e**

The Mesh

- Gmsh does journaling into the **`geo`**-file
  - it immediately writes out your entries
  - This means, that you can drive Gmsh also solely via script
  - It also means that you can make changes and reload
- Before you load:
  - **Tools** →**Options**: go to tab **Advanced**
  - Under **Text editor command: `sensible-editor`** to **`emacs`**
    - You should do a **File**→**Save Options As Default**
  - **Geometry** →**Edit file**

The Mesh



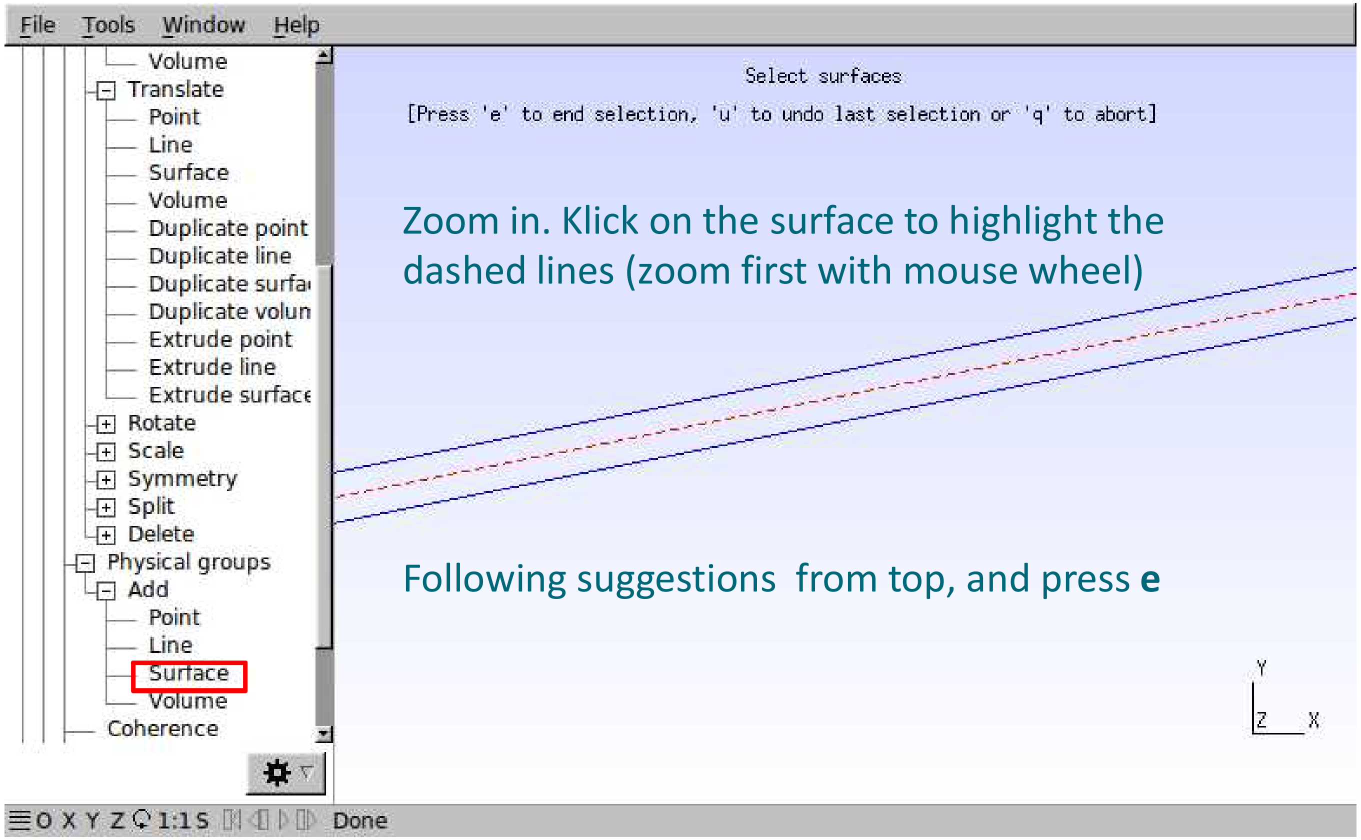
- Add:

`Layers{10};Recombine;`

- Save the changes
- In Gmsh:

  Geometry →Reload

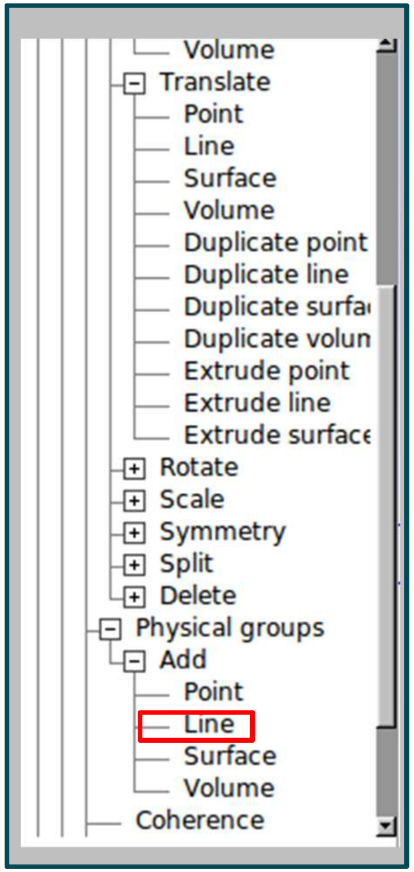
The Mesh

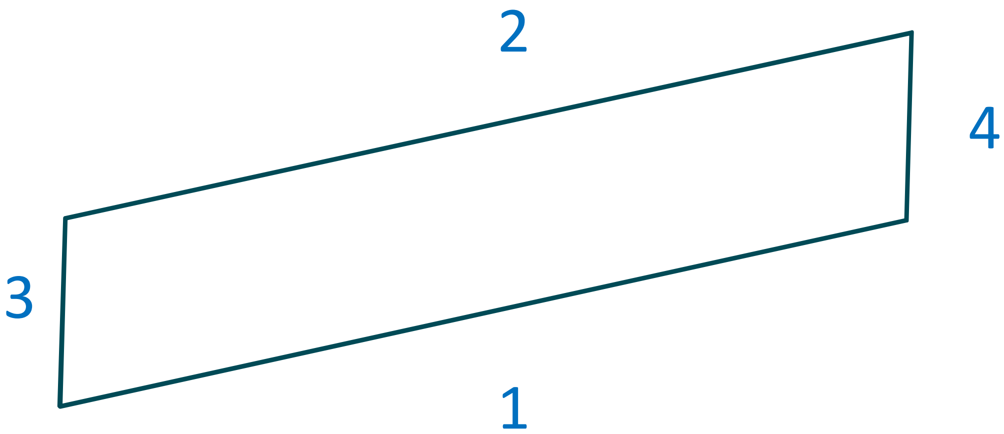


Zoom in. Klick on the surface to highlight the dashed lines (zoom first with mouse wheel)

Following suggestions from top, and press **e**

The Mesh

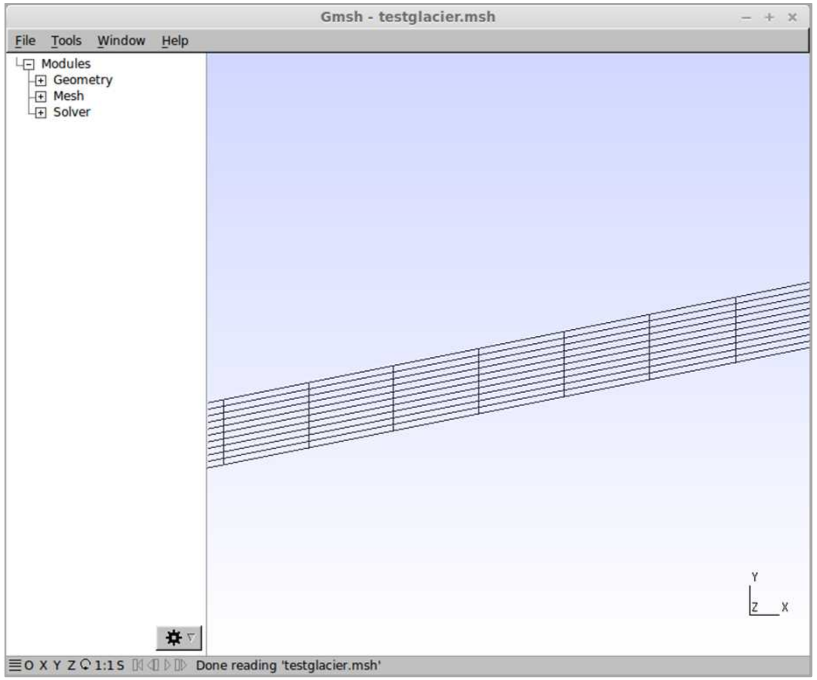




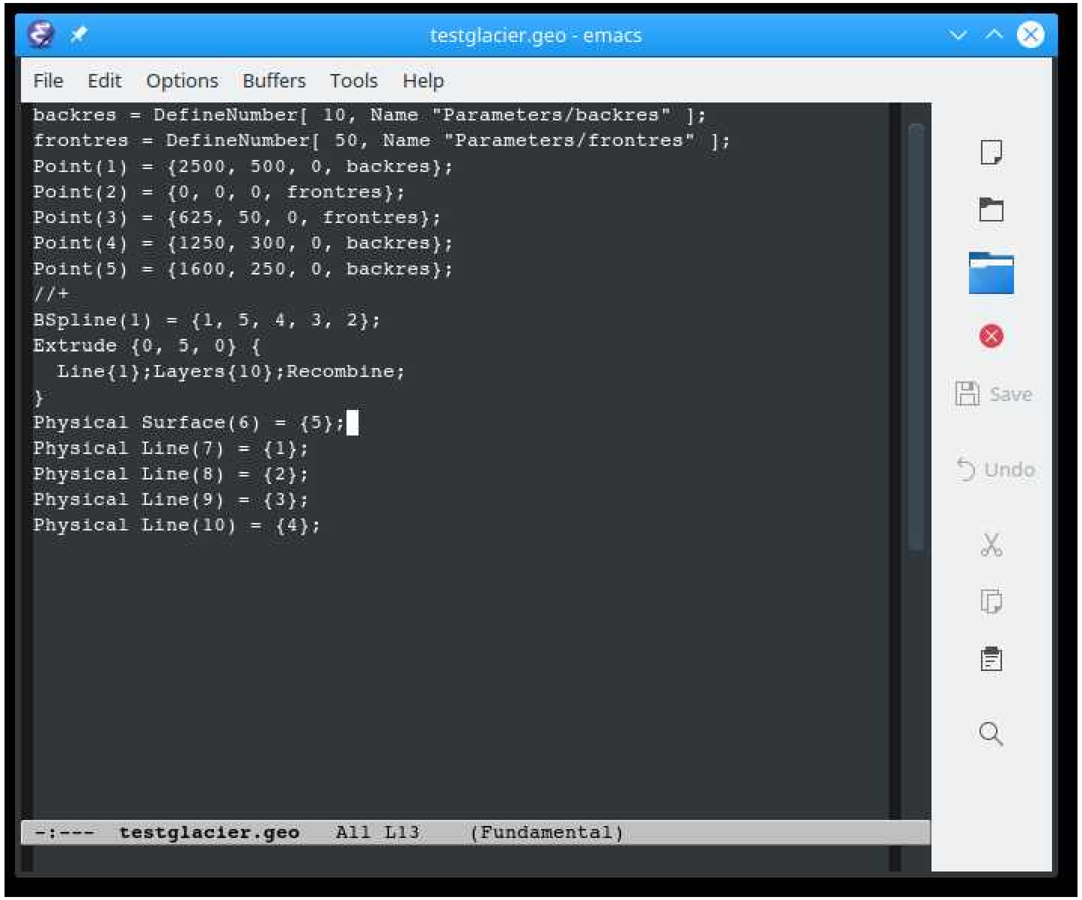
- You have to zoom (mouse wheel) in and out of the model
- and translate (right mouse button)
- Select boundary in the given order (highlights in red) and press “**e**” every time
  - If you selected the wrong boundary, use “**u**” to unselect

The Mesh

- Finally, mesh the geometry: **Mesh→2D**
- And save the mesh: **Mesh→Save**



The Mesh




```
backres = DefineNumber[ 10, Name "Parameters/backres" ];
frontres = DefineNumber[ 50, Name "Parameters/frontres" ];
Point(1) = {2500, 500, 0, backres};
Point(2) = {0, 0, 0, frontres};
Point(3) = {625, 50, 0, frontres};
Point(4) = {1250, 300, 0, backres};
Point(5) = {1600, 250, 0, backres};
//+
BSpline(1) = {1, 5, 4, 3, 2};
Extrude {0, 5, 0} {
  Line{1};Layers{10};Recombine;
}
Physical Surface(6) = {5};
Physical Line(7) = {1};
Physical Line(8) = {2};
Physical Line(9) = {3};
Physical Line(10) = {4};
```

- The whole script looks like this and can be run via terminal:

$ **`gmsh -2 testglacier.geo`**