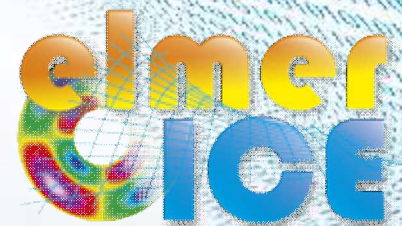




## Elmer/Ice – New Generation Ice Sheet Model



# Elmer/Ice – 2D glacier toy model

Thomas Zwinger



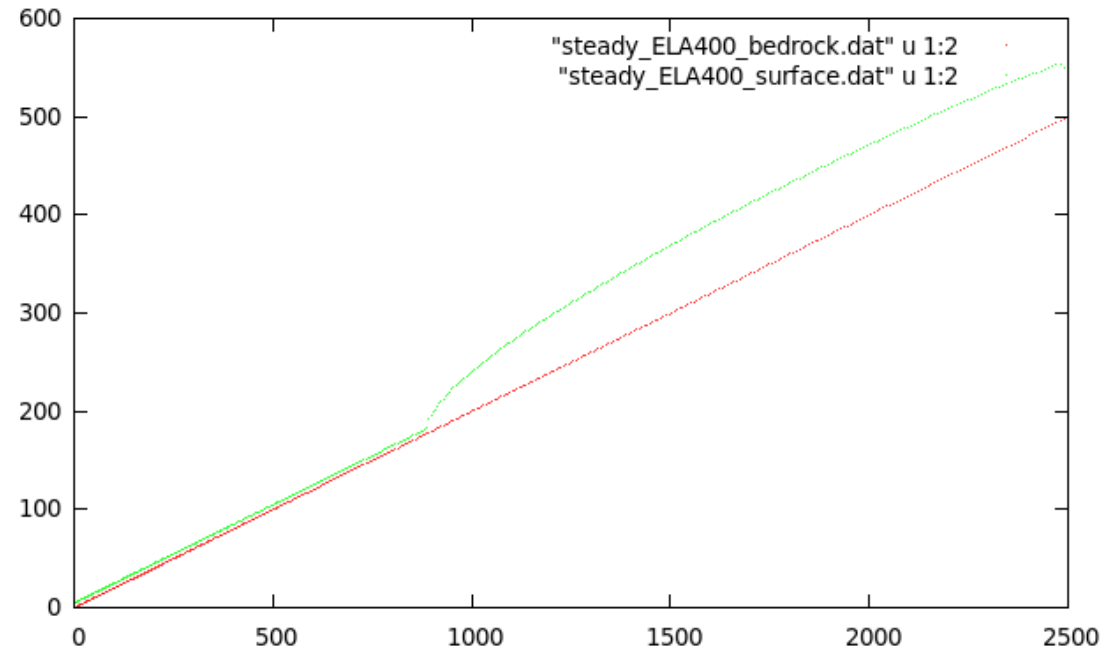
# DIAGNOSTIC RUN

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

- Build the geometry and mesh it
- Set up runs on fixed geometry
- Introduce sliding
- Introduce heat transfer (thermo-mechanical coupling)

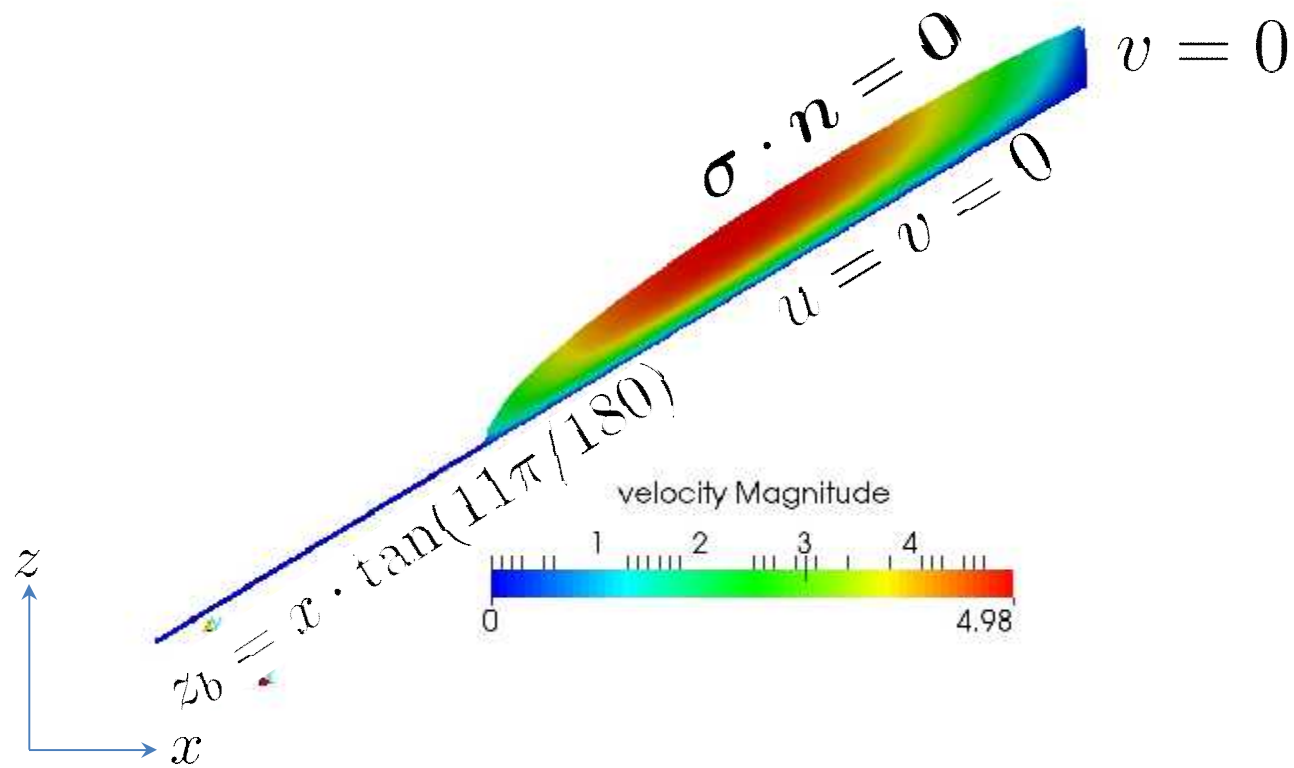
# The diagnostic problem

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



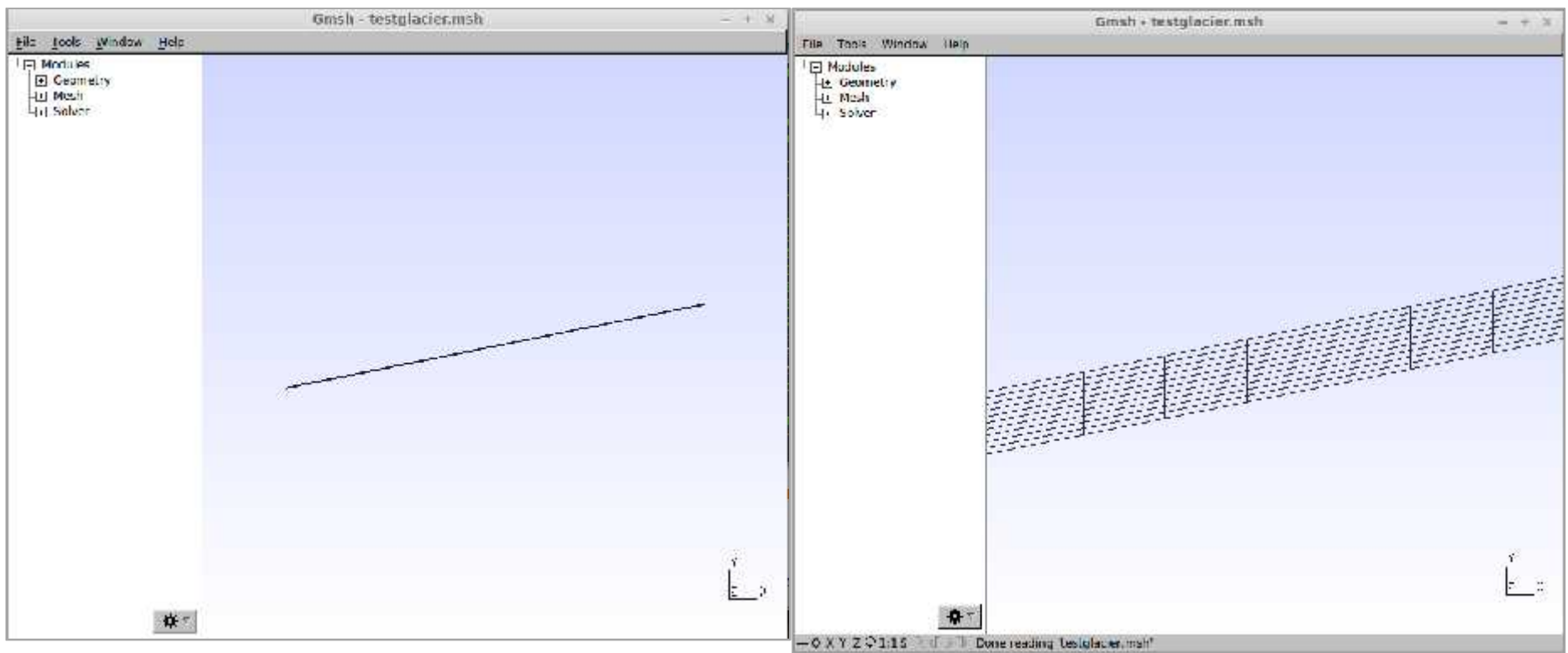
- The distributions are given in the files:  
**steady\_ELA400\_bedrock.dat** , **steady\_ELA400\_surface.dat**

# The diagnostic problem



# The diagnostic problem

- We use a 11 deg inclined rectangular mesh (produced with Gmsh) of unit-height



# The diagnostic problem

- Open the Gmsh file:

```
$ gmsh testglacier.geo
```

- Go to Mesh and press the 2D button
- Save the mesh
- 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

- Open the Solver Input File (SIF)

```
$ emacs Stokes_diagnostic.sif
```

- Steady state simulation = diagnostic

## 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 legacy format
```

```
Initialize Dirichlet Conditions = Logical False
```

```
End
```



# 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
  Compute Normals = Logical True
  ! include the bedrock DEM, which has two columns
  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 columns
  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:

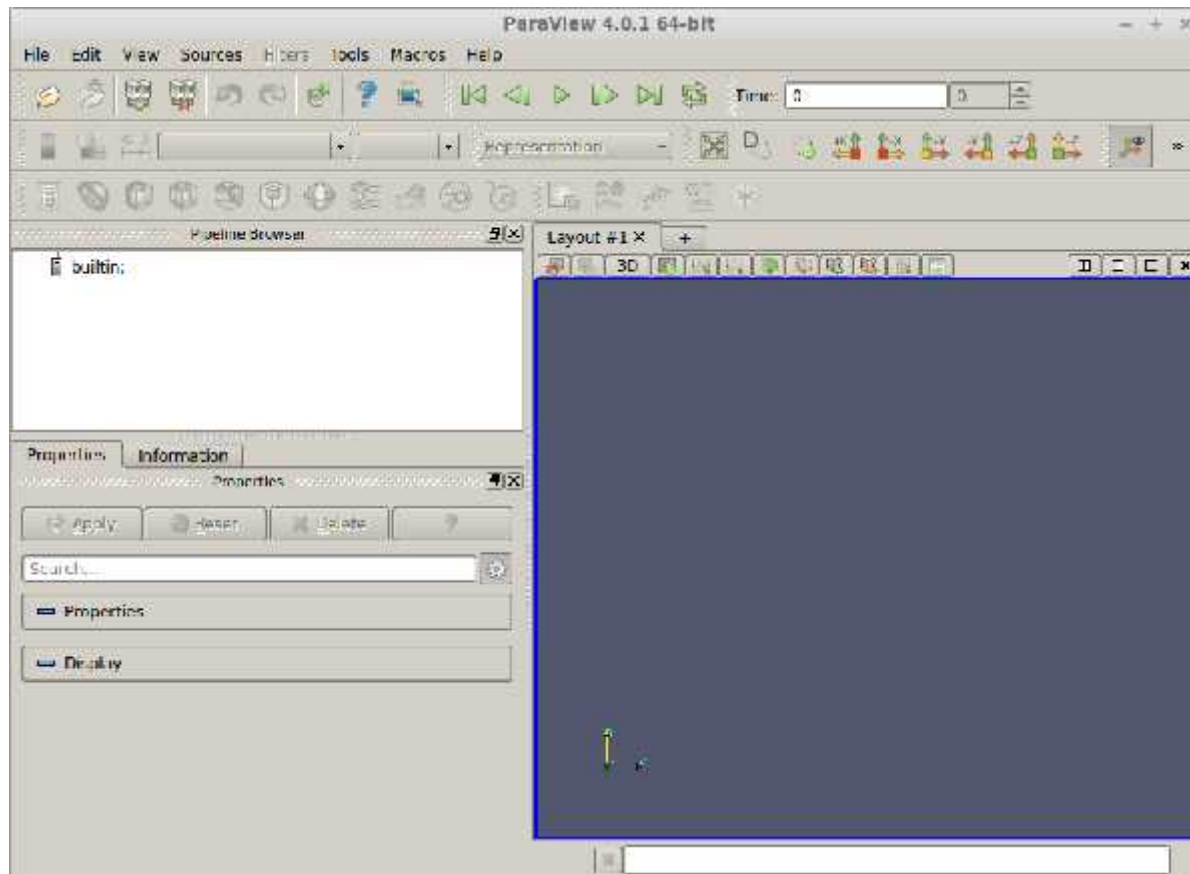
```
$ ElmerSolver Stokes_diagnostic.sif
```

- 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
```

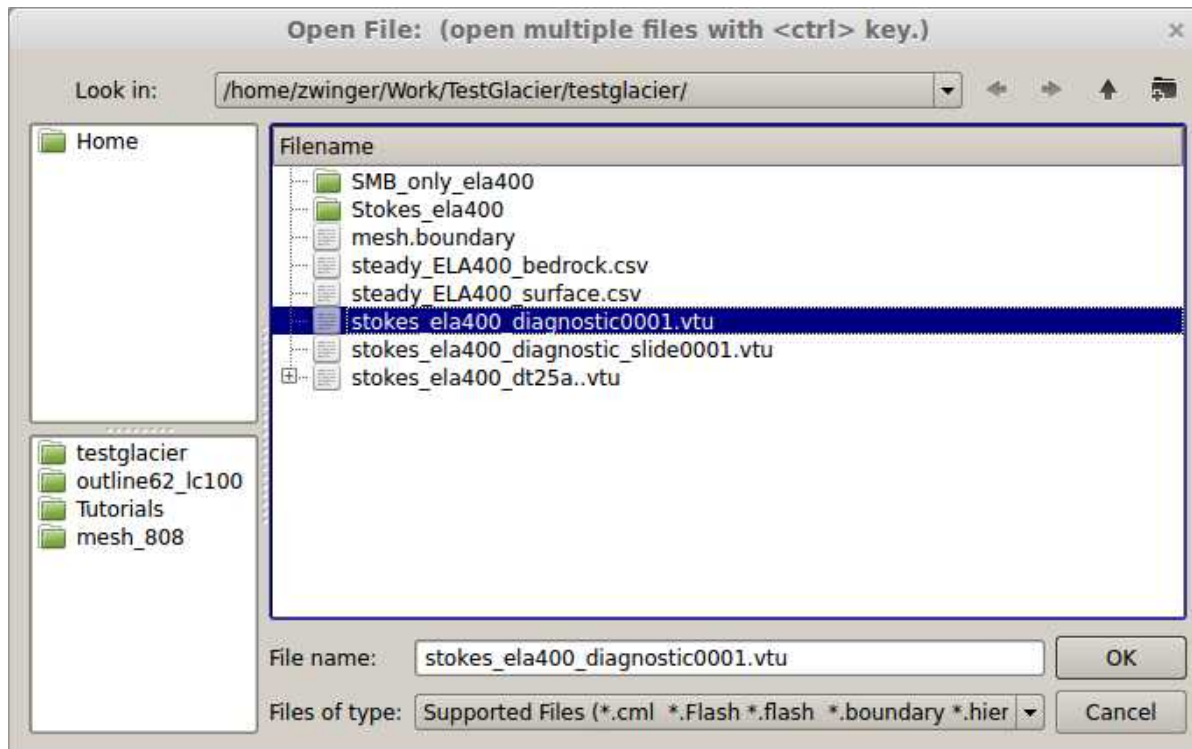
# The diagnostic problem

- Post-processing using ParaView: `$ paraview`



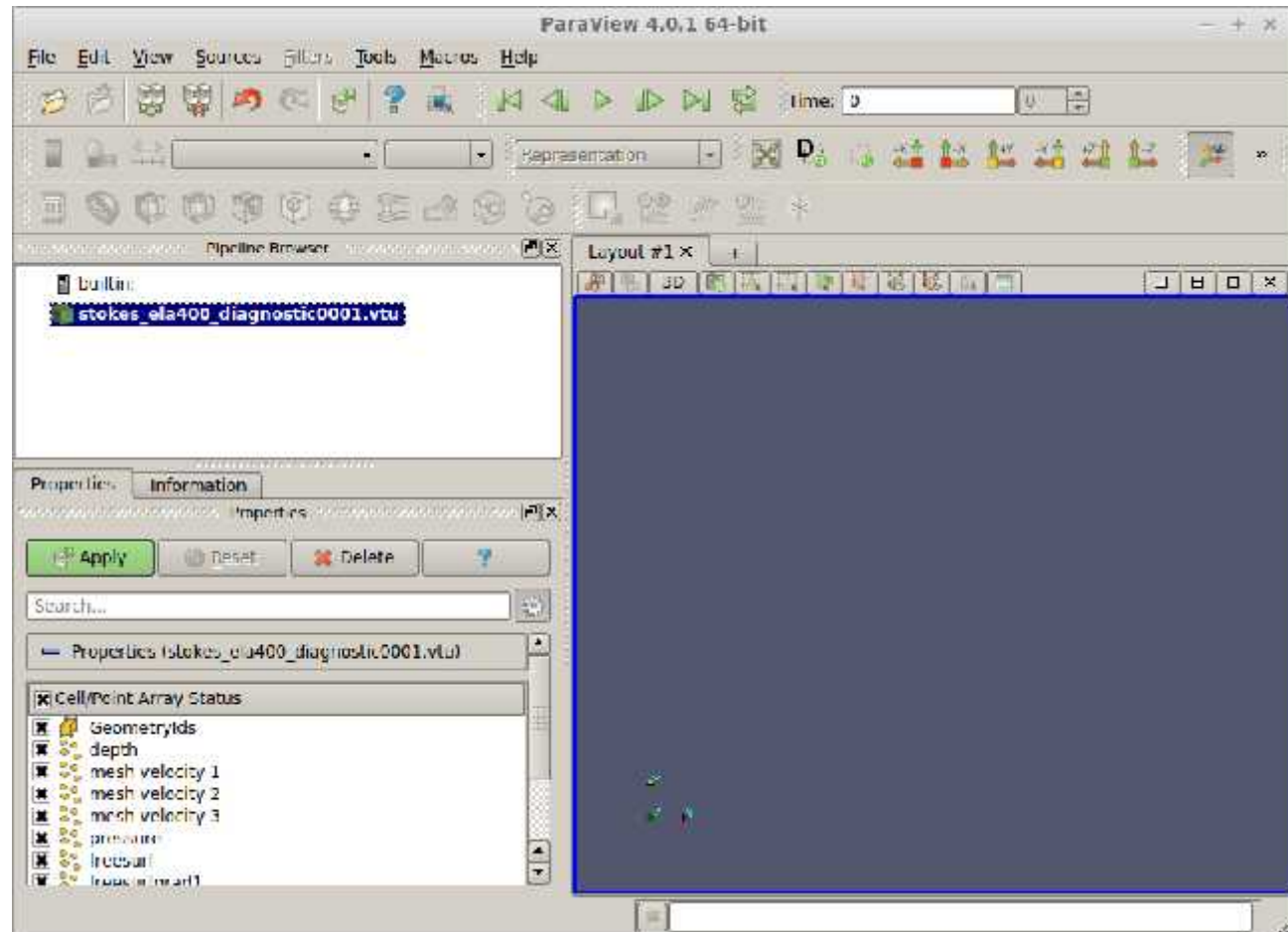
# The diagnostic problem

- File → Open `stokes_ela400_diagnostic0001.vtu`



# The diagnostic problem

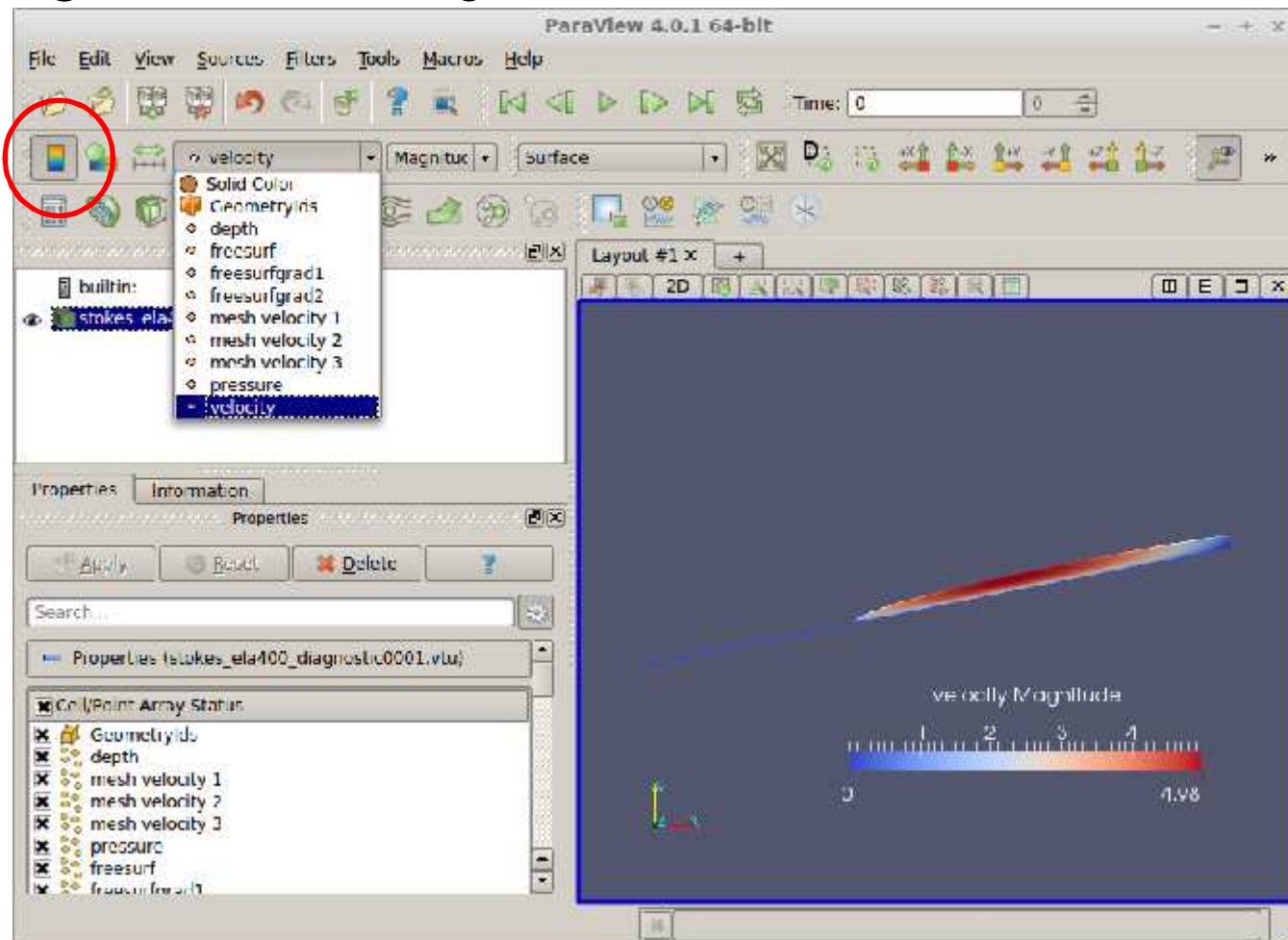
- Apply



# The diagnostic problem

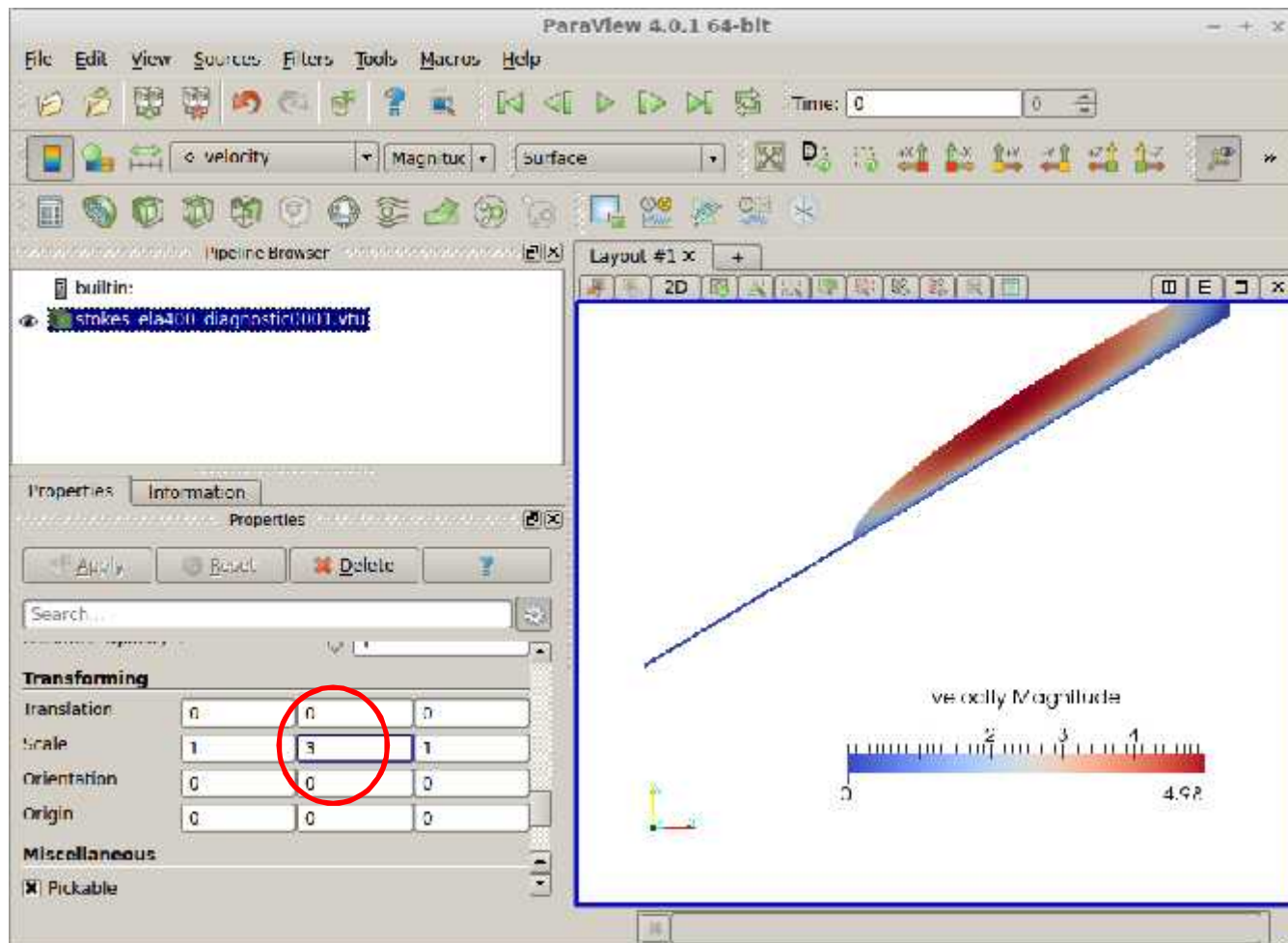
- Change to velocity

Press to activate colour bar



# The diagnostic problem

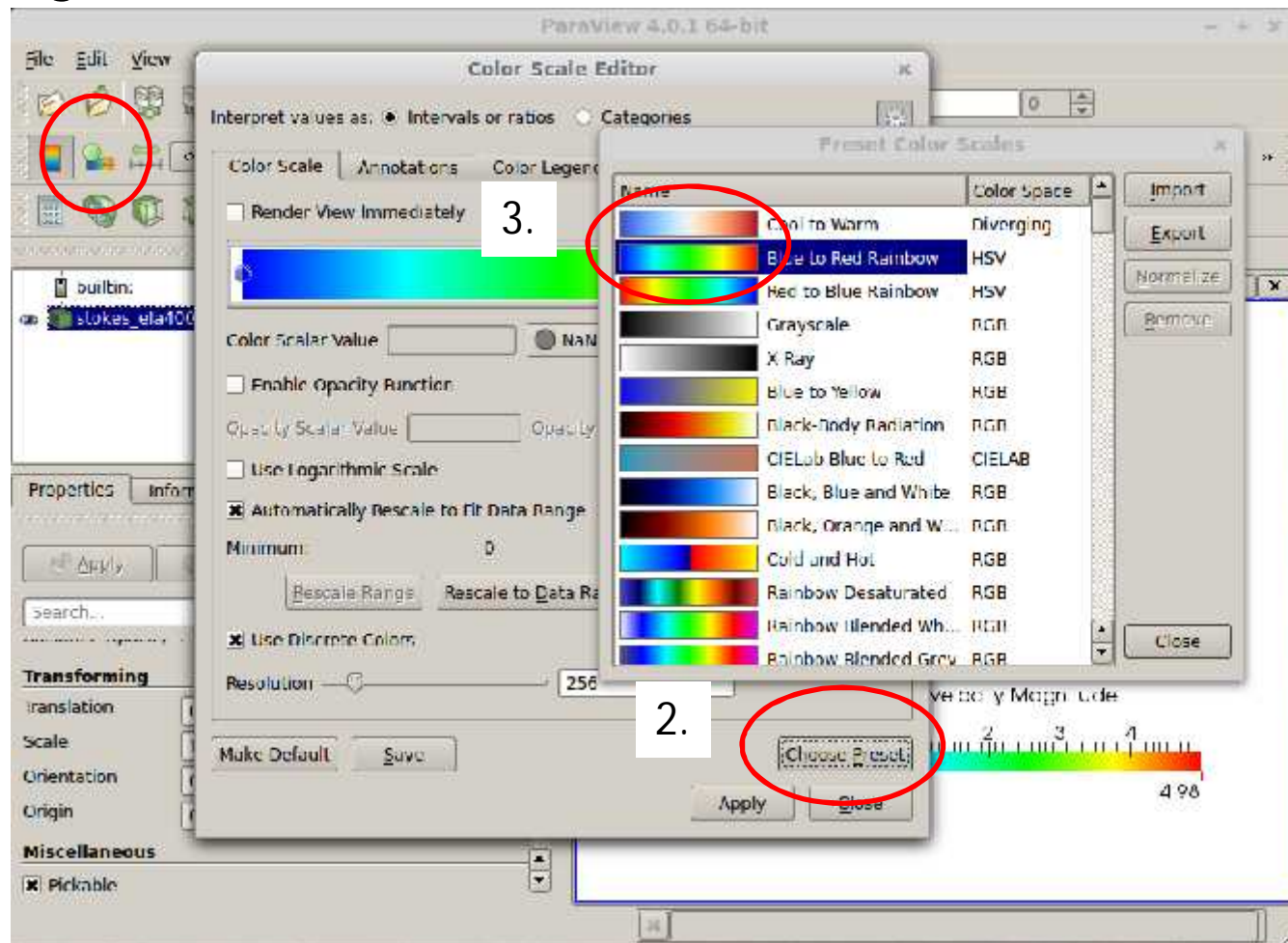
- Scale



# The diagnostic problem

- Change colours

1.





# Sliding

- Different sliding laws in Elmer
- Simplest: Linear Weertman  $\tau = \beta^2 \mathbf{u}$ 
  - This is formulated for the traction  $\tau$  and velocity  $\mathbf{u}$  in tangential plane
- In order to define properties in normal-tangential coordinates: **Normal-Tangential Velocity = True**
- $\beta^{-2}$  is the **slip Coefficient** {2,3} (for the tangential directions 2 and 3)
- 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  $400\text{m}$

```
Boundary Condition 1
  Name = "bedrock"
  Target Boundaries = 1
  Compute Normals = Logical True
  ! include the bedrock DEM, which has two columns
  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

- 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 Take last entry
End
```

# The diagnostic problem

- Now, run the case:

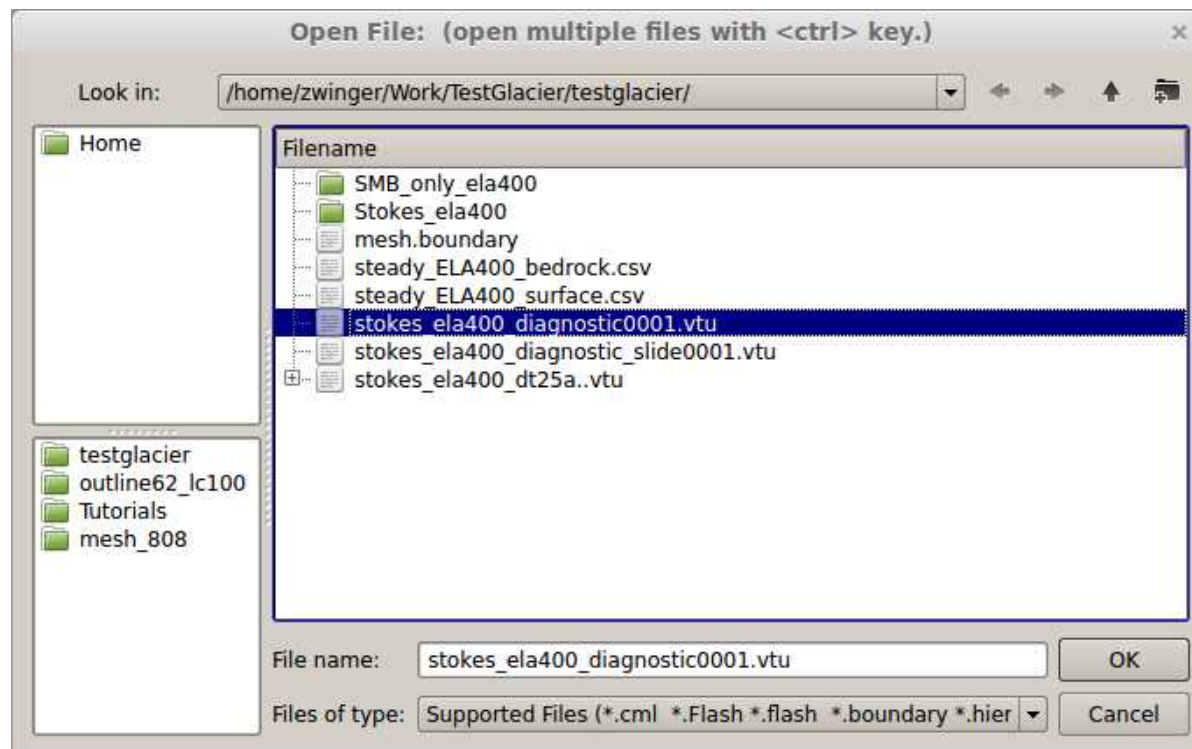
```
$ ElmerSolver Stokes_diagnostic_slide.sif
```

– Converged much earlier:

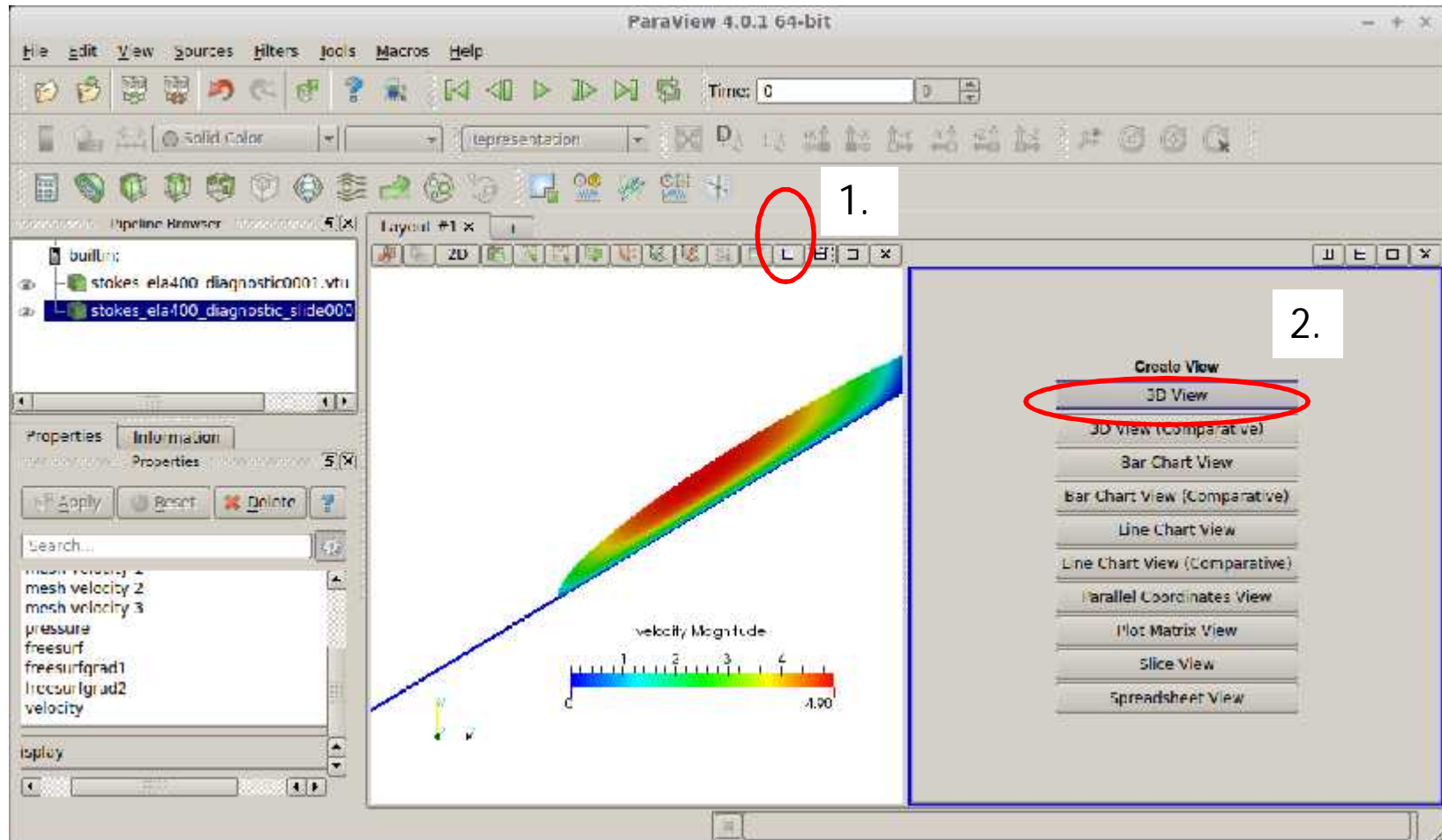
```
FlowSolve: -----  
FlowSolve:  NAVIER-STOKES ITERATION          12  
FlowSolve: -----  
FlowSolve:  
FlowSolve: Starting Assembly...  
FlowSolve: Assembly done  
FlowSolve: Dirichlet conditions done  
ComputeChange: NS (ITER=12) (NRM,RELC): (  3.4915753  
0.34732117E-05 ) :: navier-stokes  
FlowSolve: 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

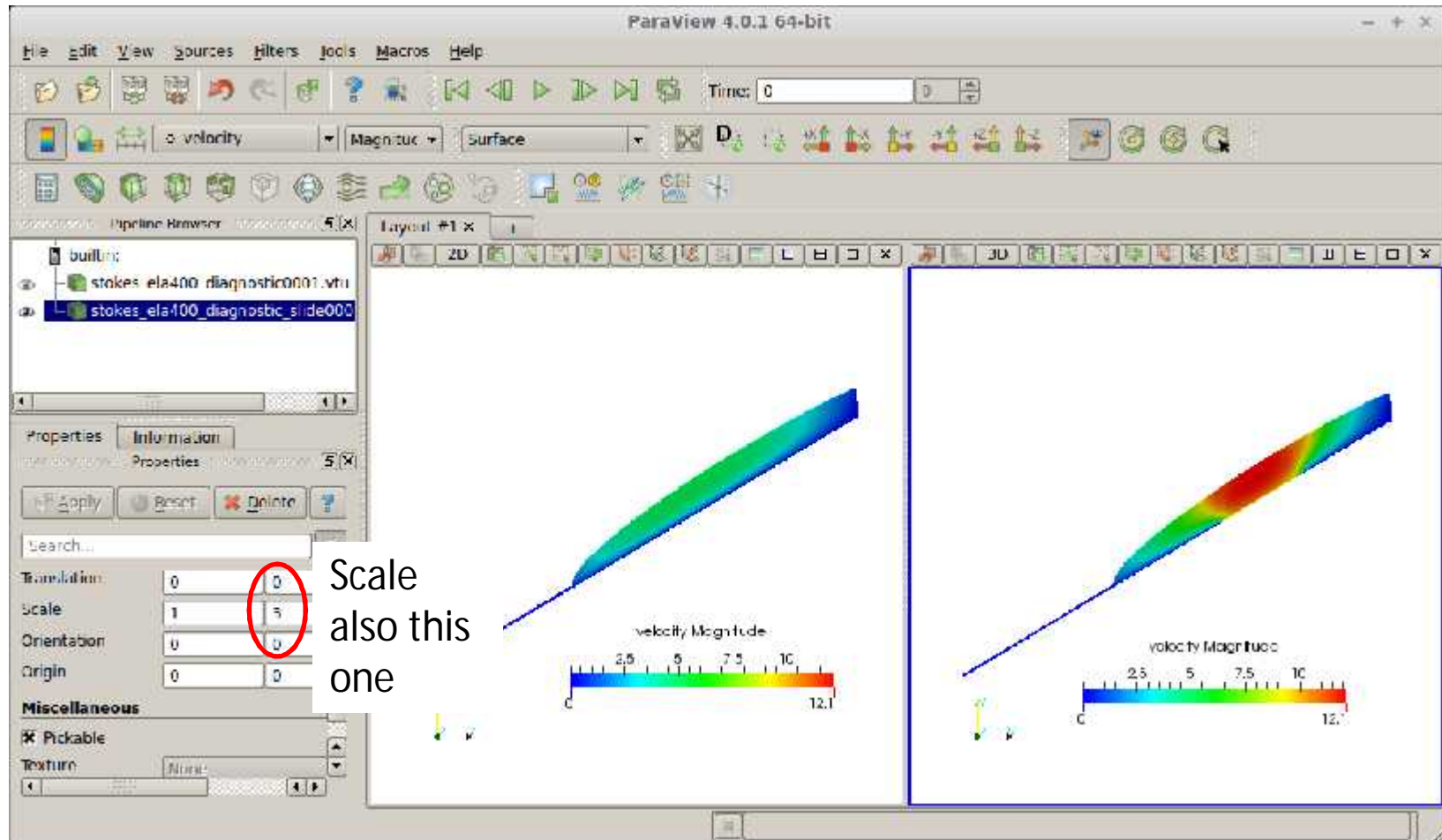
- Load parallel to previous file
- File → Open  
`stokes_ela400_diagnostic_slide0001.vtu`



# Sliding



# Sliding



# Heat transfer

- Adding heat transfer:
  - Add `ElmerIceSolvers TemperatureIceSolver` with variable name `Temp` (see next slide)
  - Surface temperature distribution: linear from 273.15K at  $z=0\text{m}$  to 263.15K at  $z=1000\text{m}$

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

- Geothermal heat flux of  $200 \text{ mW m}^{-2}$  at bedrock

```
Temp Flux BC = Logical True
Temp Heat Flux = Real $ 0.200 * (31556926.0)*1.0E-06
```



# Heat transfer

Solver 4

```
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:  $c(T) = 146.3 + (7.253 \cdot T[\text{K}])$
  - Conductivity:  $\kappa(T) = 9.828 \exp(-5.7 \times 10^{-3} \cdot T[\text{K}])$

# 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:  $c(T) = 146.3 + (7.253 \cdot T[\text{K}])$
  - Conductivity:  $\kappa(T) = 9.828 \exp(-5.7 \times 10^{-3} \cdot T[\text{K}])$

# Heat transfer

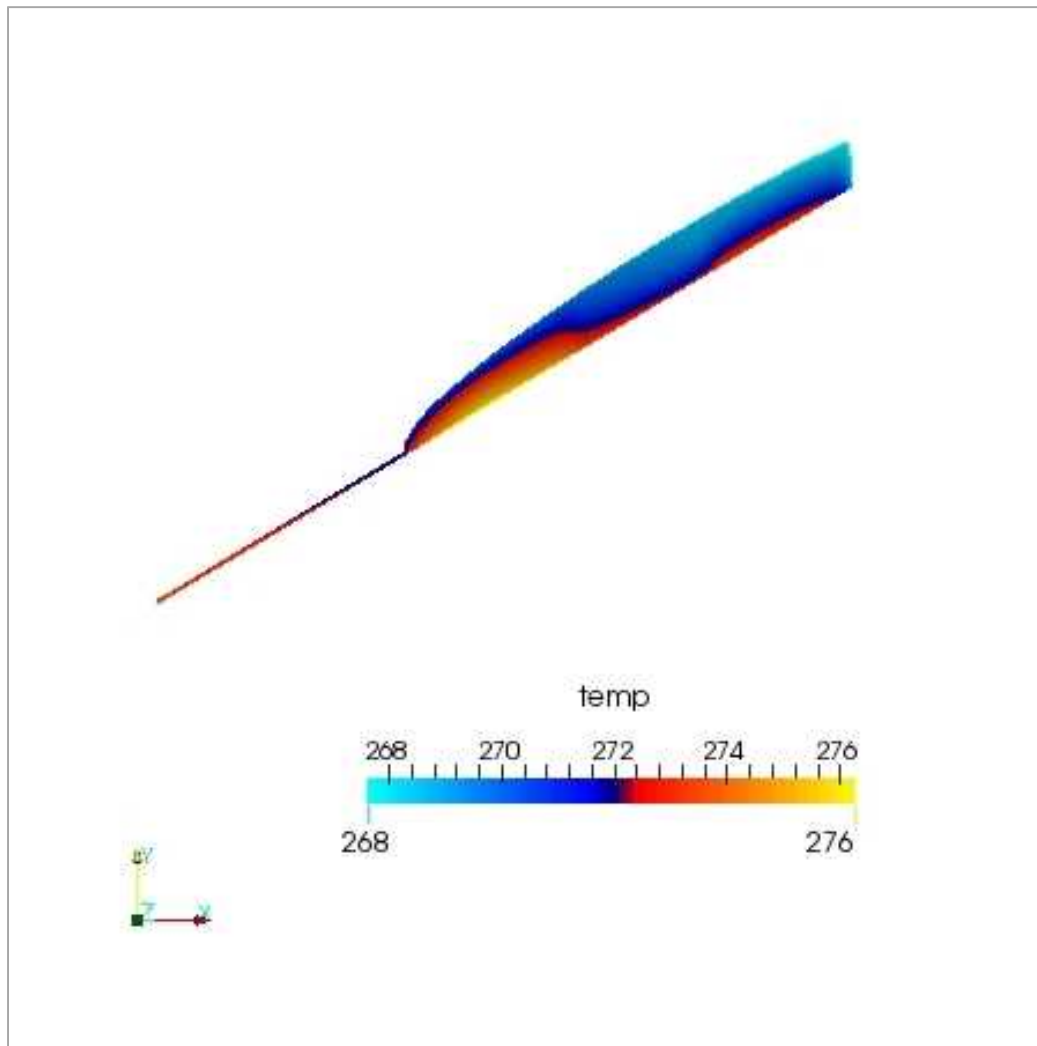
- Now, run the case:

```
$ ElmerSolver Stokes_diagnostic_temp.sif
```

- It goes pretty quick, as we only have one-way coupling and hence don't even execute 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  
ElmerIceSolvers TemperatureIce

```
! 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 - 9.8E-08 * tx * 910.0 * 9.81"
```

$$T_{\text{pm}} = T_0 + \beta_c p$$

$$p \approx \rho_{\text{ice}} g d$$

# Heat transfer

- Now, run the case:

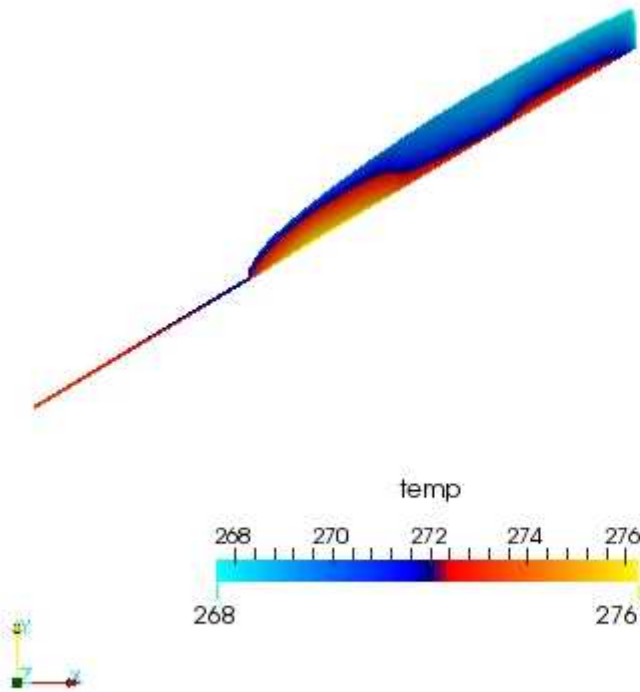
```
$ ElmerSolver \
    Stokes_diagnostic_temp_constrained.sif
```

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

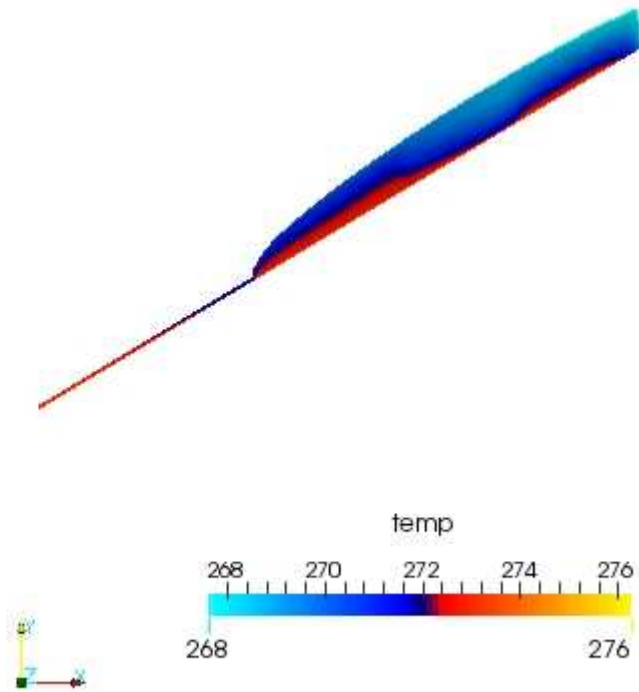
```
TemperateIceSolver (temp): iter:      5 Assembly: (s)      1.36      6.77
TemperateIceSolver (temp): iter:      5 Solve:      (s)      0.00      0.01
TemperateIceSolver (temp): Result Norm   :      271.78121462656480
TemperateIceSolver (temp): Relative Change :
5.0215061382786350E-006
ComputeChange: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



# Heat transfer



Unconstrained



Constrained

# Heat transfer

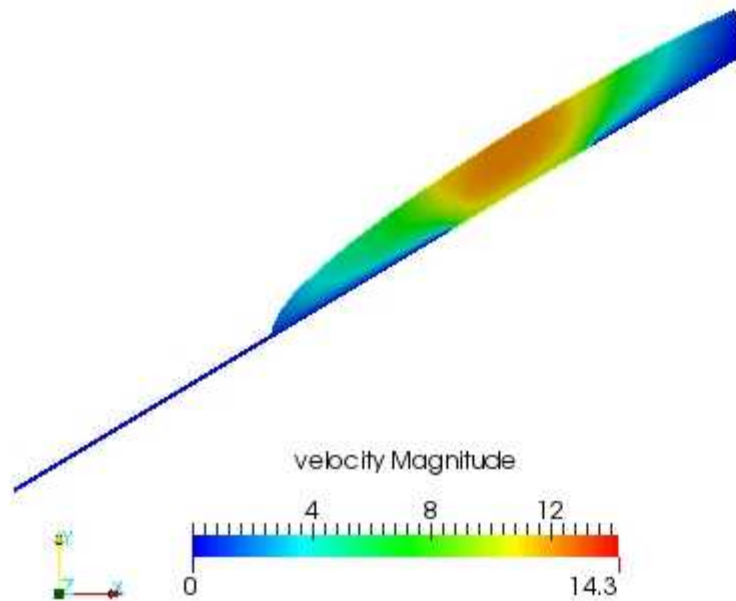
- Thermo-mechanically coupled simulation:
  - We have to iterate between Stokes and HTEq.

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

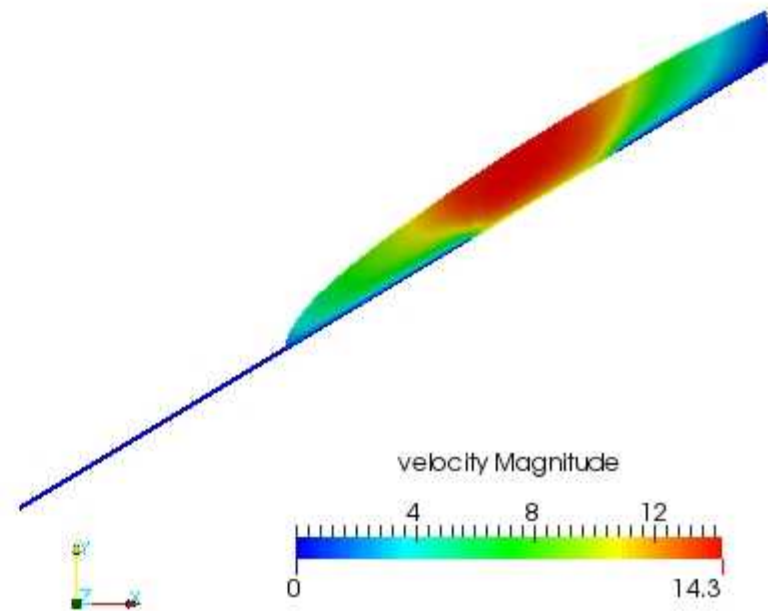
- Coupling to viscosity in Material section

```
! 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"
```

# Heat transfer



Uncoupled (constant T)



Thermo-mechanically coupled

# 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):

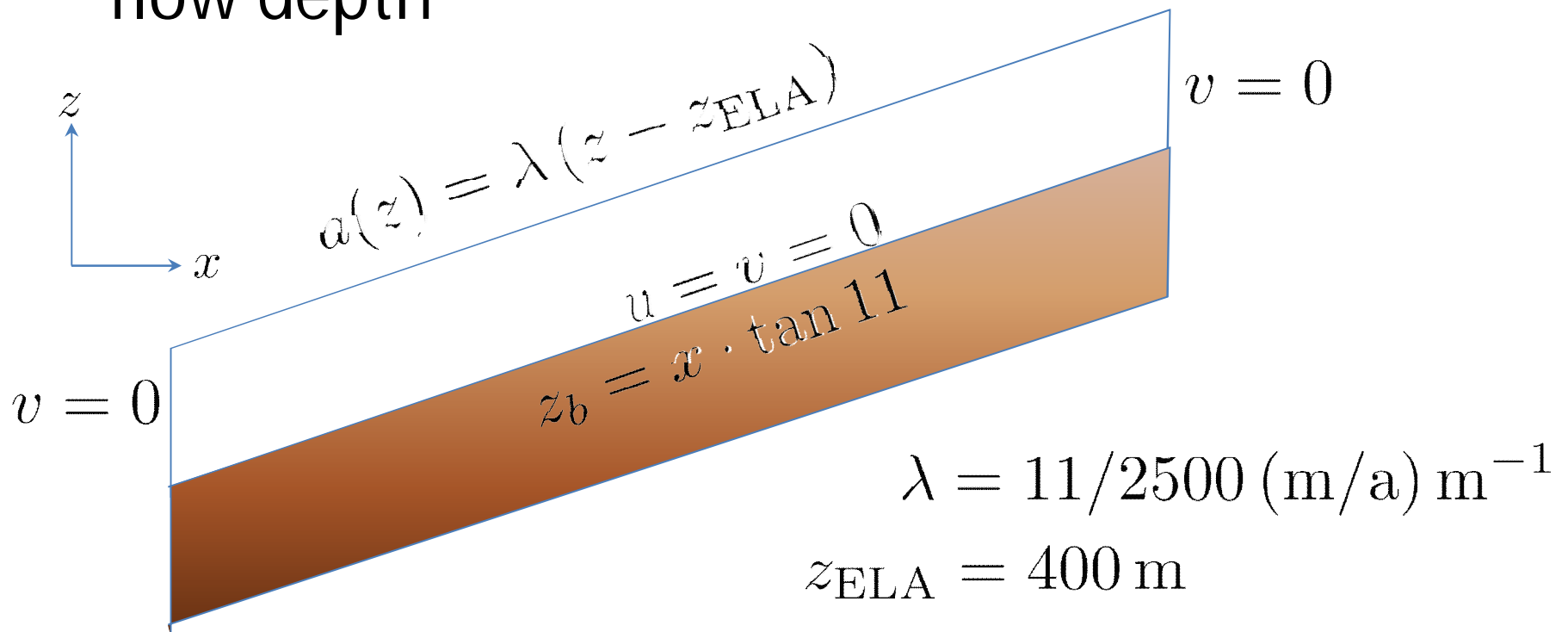
$$a_{\text{ELA}} = \lambda z_{\text{ELA}} + a_0 = 0$$

- We know lapse rate,  $\lambda$ , and  $z_{\text{ELA}}$  and have to define

$$a_0 = -\lambda z_{\text{ELA}}$$

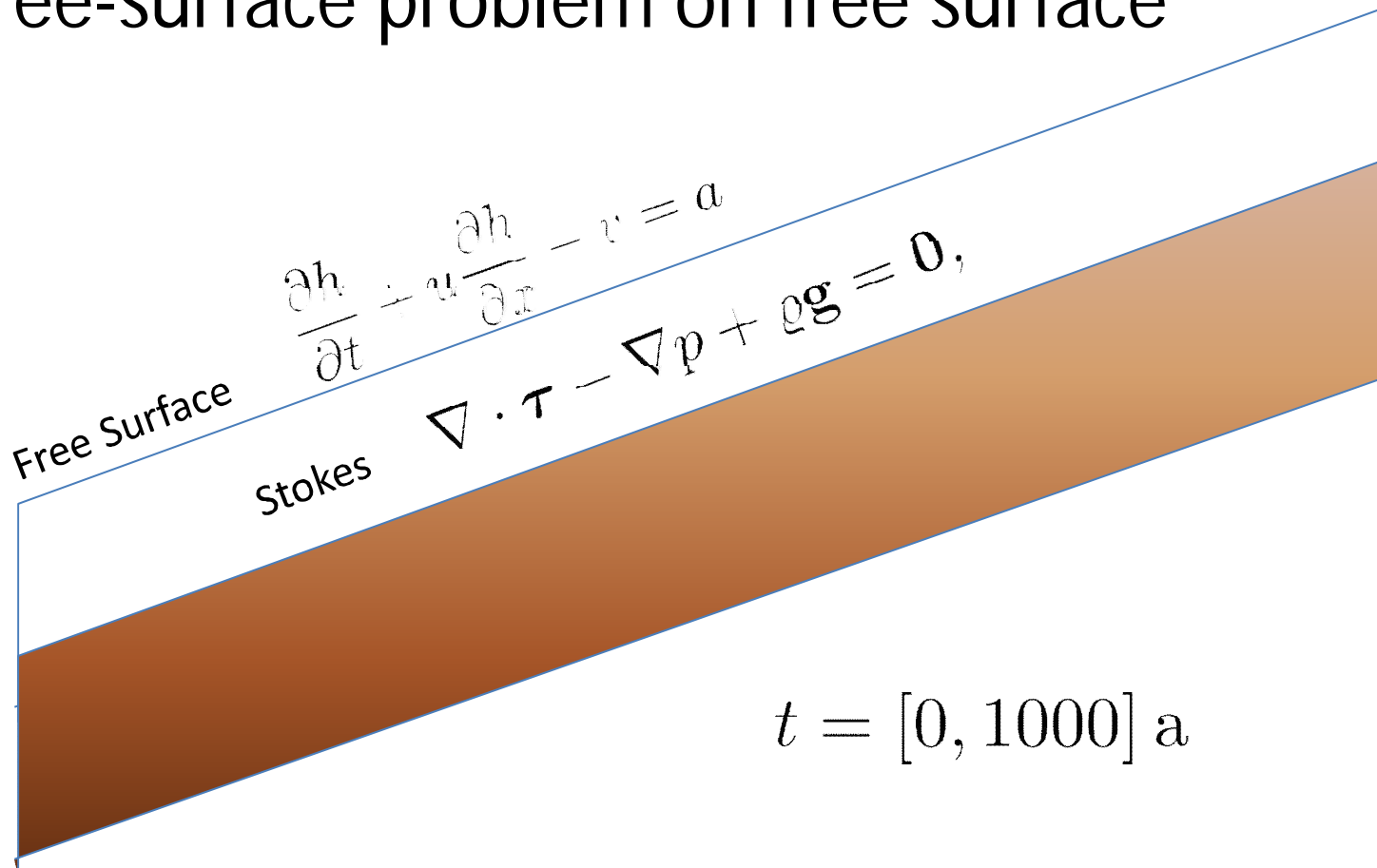
# The prognostic problem

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



# The prognostic problem

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



# The prognostic problem

- Changing to transient case

```
Simulation
...
Simulation Type = "Transient"
...
End
```

- MATC function for accumulation/ablation

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

- **AS** BodyForce for FreeSurfaceSolver

# The prognostic problem

## Solver 3

```
Exec Solver = always
Equation = "Free Surface"
Variable = String "Zs"
Variable DOFs = 1
Exported Variable 1 = -dofs 1 "Zs Residual"
Exported Variable 2 = -dofs 1 "RefZs"
Procedure = "FreeSurfaceSolver" "FreeSurfaceSolver"
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 Dirichlet = Logical True
```

End



# The prognostic problem

- Set initial z-values

```
Initial Condition 2
...
Zs = Equals Coordinate 2
RefZs = Equals Coordinate 2
...
End
```

- Accumulation in Body Force

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

- Maximum and minimum value

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

# The prognostic problem

- Need a solver to move the mesh
  - This one uses the structured extruded mesh

```
Solver 4
```

```
Exec Solver = "after timestep"
```

```
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 unrealistically high
```

```
Mesh Velocity First Zero = Logical True
```

```
Dot Product Tolerance = Real 0.01
```

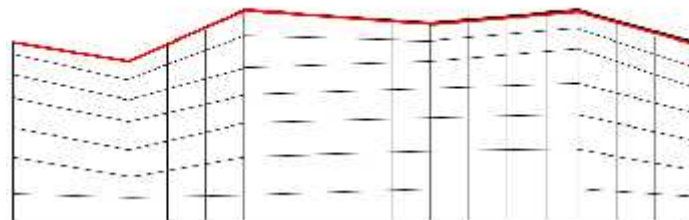
```
End
```

# The prognostic problem

- Coupling of free surface

```
Boundary Condition 3  
Name = "surface"  
Top Surface = Equals "Zs"  
Target Boundaries = 2  
Body ID = 2  
Depth = Real 0.0  
End
```

- Bodies on surfaces
  - Free surface condition is a dimension-1 PDE
  - Need to run it on body defined on surface



# The prognostic problem

