



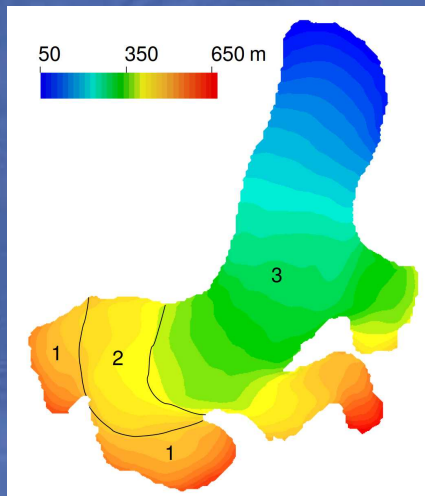
# Arctic Glacier Example

## Midtre Lovénbreen, Svalbard

Thomas Zwinger  
ElmerTeam

data contributions from Jack Kohler (NPI)

# Midtre Lovénbreen (MLB)



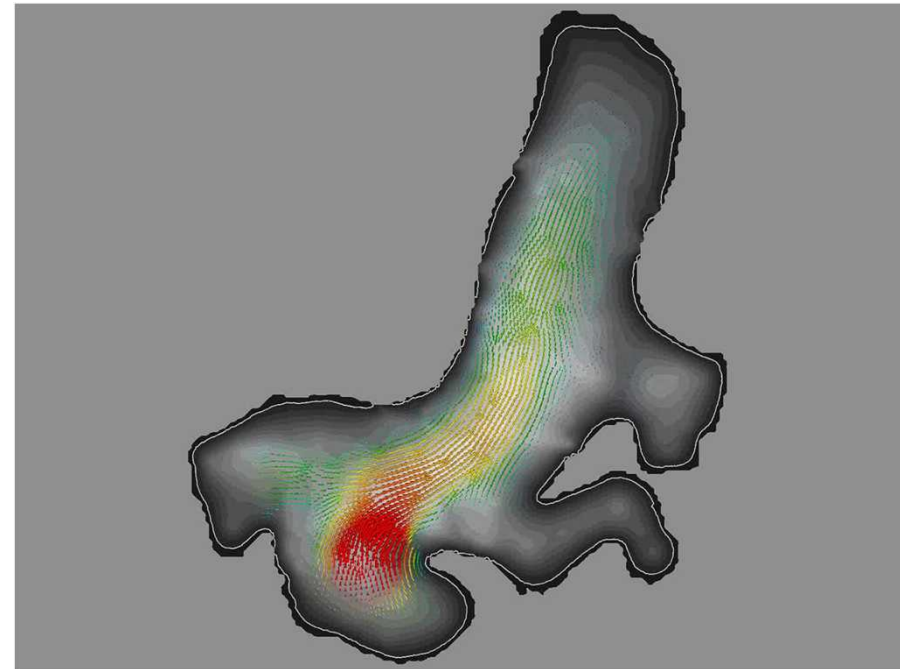
Vestfonna

Austfonna

SVALBARD

## Midtre Lovénbreen

- Midtre Lovénbreen (MLB) is a small glacier very close to the research station in Ny Ålesund ([https://www.swisseduc.ch/glaciers/svalbard/midtre\\_lovenbreen/index-en.html](https://www.swisseduc.ch/glaciers/svalbard/midtre_lovenbreen/index-en.html))
- Very well monitored glacier
  - Part of World Glacier Monitoring Service (WGMS) ([https://wgms.ch/products\\_ref\\_glaciers/midtre-lovenbreen-svalbard/](https://wgms.ch/products_ref_glaciers/midtre-lovenbreen-svalbard/))
- It is a cold-ice glacier in the lower parts, but in the upper parts temperate
- From 50 – 650 m a.s.l., ~5 km<sup>2</sup>
- Like most glaciers under constant retreat



Zwinger T. and J.C. Moore, 2009. *Diagnostic and prognostic simulations with a full Stokes model accounting for superimposed ice of Midtre Lovénbreen, Svalbard*, *The Cryosphere*, 3, 217-229, doi:10.5194/tc-3-217-2009

## Reconstructing SMB: Midtre Lovénbreen, Svalbard

Pictures and data provided by Jack Kohler, NPI, NOR (2005 DEM from NERC)

- DEM's obtained at different times

- Using 2 consecutive time-levels

- Obtaining averaged DEM

- $h_{2000} = (h_{2005} - h_{1995})/2$

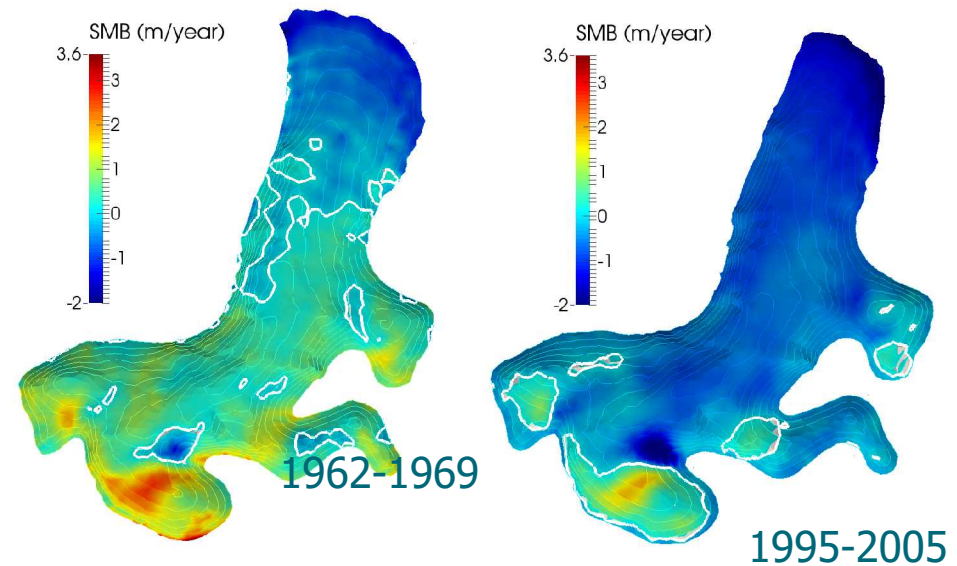
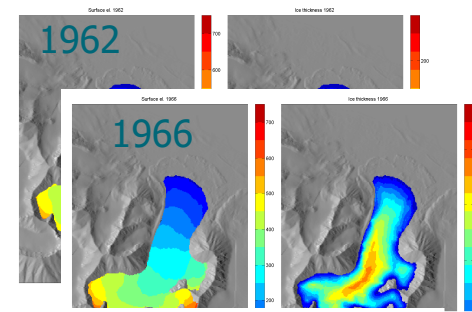
- and local elevation change

$$\frac{\partial h}{\partial t} \Big|_{2000} = (h_{2005} - h_{1995})/11$$

- Elmer/Ice FS diagnostic simulations  $\rightarrow \mathbf{u} = (u, v, w)^T$

- Spatial distribution of SMB:

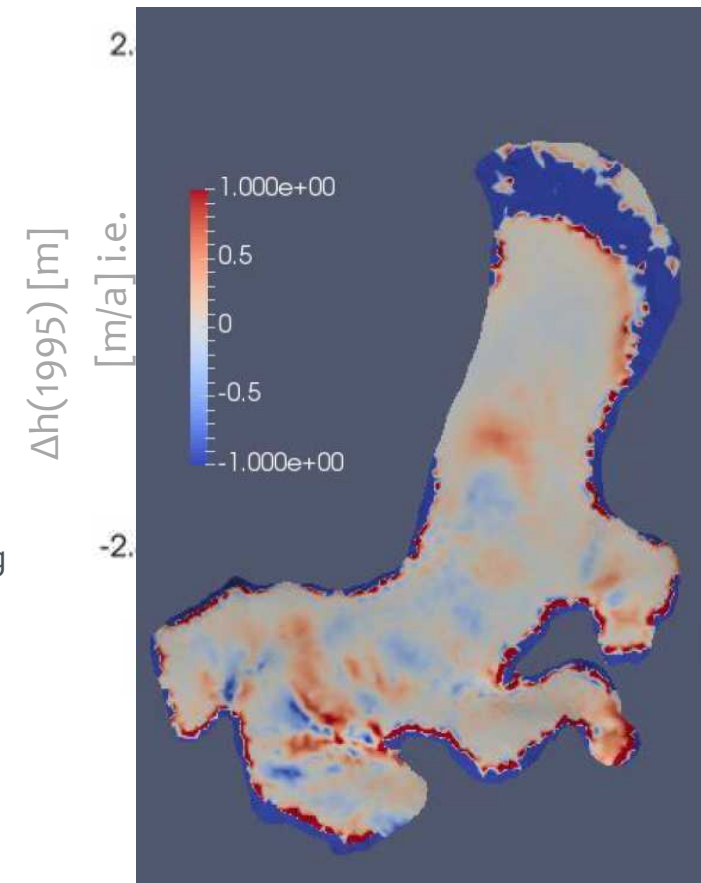
$$\text{SMB} = \left( \frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} + v \frac{\partial h}{\partial y} - w \right)$$



Välisuo, I., T. Zwinger and J. Kohler (2017): *Inverse solution of surface mass balance of Midtre Lovénbreen, Svalbard*, Journal of Glaciology, 1-10, doi:10.1017/jog.2017.26.

## Reconstructing Climate: Midtre Lovénbreen, Svalbard

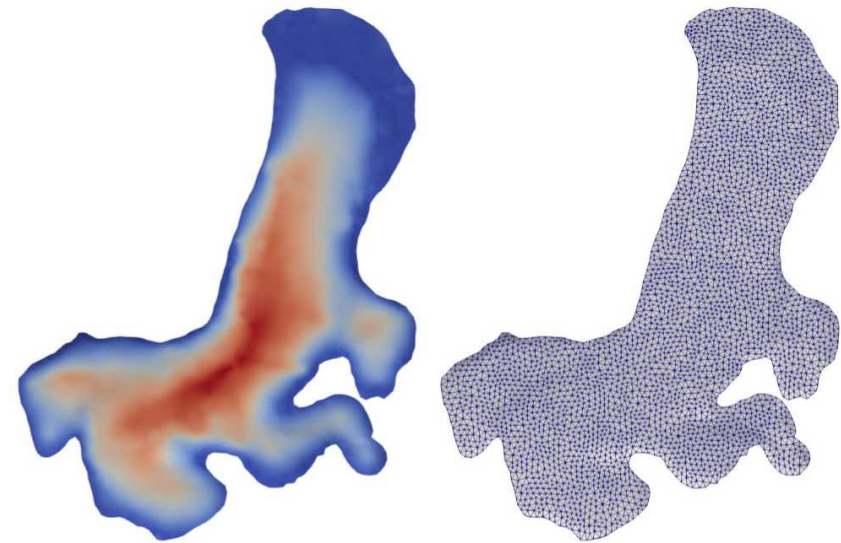
- This works nicely on MLB, because it is a relatively slow flowing glacier, dominated by SMB
  - Quite simple method, as it only needs a single diagnostic run for one time-interval
- **BUT:** It will fail on any ice-mass that shows significant amount of basal sliding
- Test run: using the 1977-1995 time interval
  - Using  $SMB_{77-95}$  obtained with DEM method (to the right)
  - Starting with 77 DEM, integrating for 18 years, obtaining surface change by solving kinematic free surface equation
  - Comparison of computed 1977-95 result and 1995 DEM



Result produced by 2017 student class  
at Univ. Helsinki

## This exercise

- We take the DEM of 1995
  - If running standard virtual machine settings, use the 75m DEM
- We will first run a diagnostic simulation on the given geometry
- Emphasis on some special features
  - 3D mesh generation using extrusion
  - Restart from 2D data
  - Utilizing extruded structure in mesh deformation
  - Vectorized & threaded version of Navier-Stokes
  - Block preconditioning
  - Semi-Lagrangian solver for purely advective transport i.e. of age
- Users are free to try out different things
  - Solution strategies
  - Parallel runs
  - ...

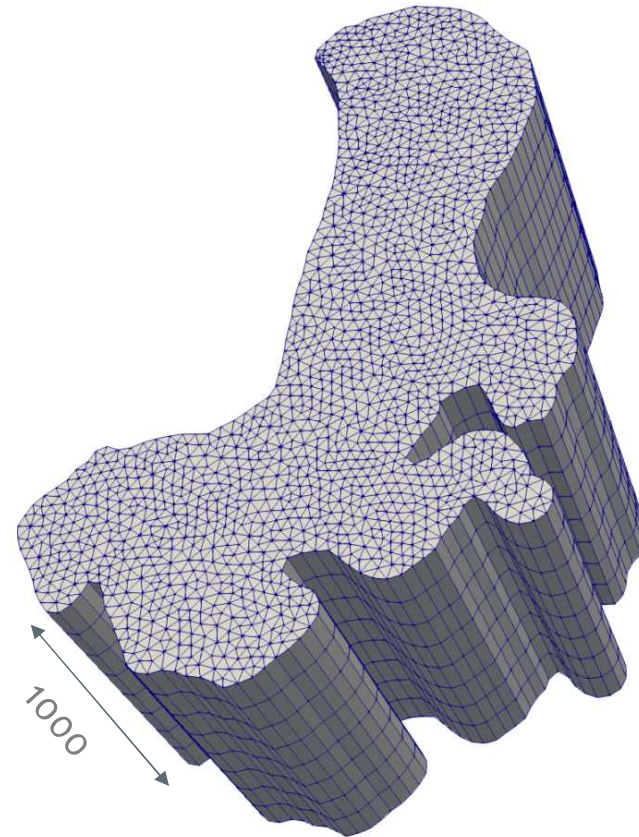


## Finalizing mesh using internal extrusion

- The mesh is finalized in memory starting from 2D footprint
- Mind! Here the extruded height does not play any role
  - Mesh is further adapted to follow true bottom and top DEM

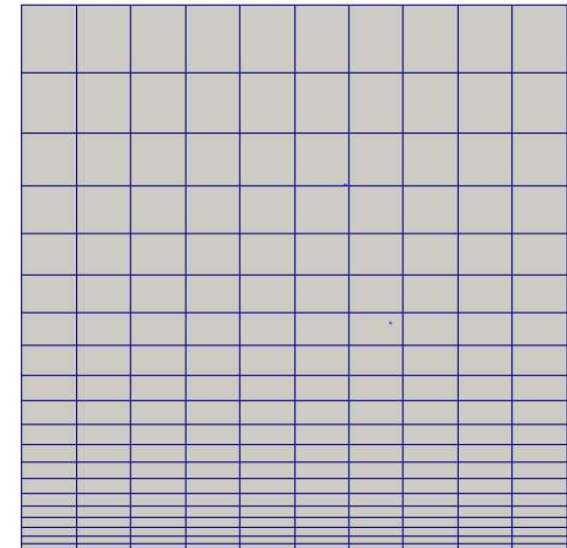
### Simulation

```
Extruded Mesh Levels = Integer 9  
Extruded Max Coordinate = Real 1000
```



## Internal mesh extrusion

- Start from an initial 2D (1D) mesh and then extrude into 3D (2D)
  - Mesh density may be given a geometric ratio and even an arbitrary function
- Implemented also for partitioned meshes
  - Extruded lines belong to the same partition by construction!
- Effectively eliminates meshing bottle-necks
- Side boundaries get a BC constraint so that
  - 2D constraint BC = 1D constraint BC + offset
  - offset is set if the baseline BCs are preserved
- Top and bottom boundaries get the next free BC constraint indexes
  - Note that the BCs refer directly to the "Boundary Condition"
  - "Target Boundaries" is used only when reading in the mesh in the 1st place and they are not available any more at this stage



```
Extruded Mesh Levels = 21
Extruded Mesh Density = Variable Coordinate 1
Real MATC "1+10*tx"
```



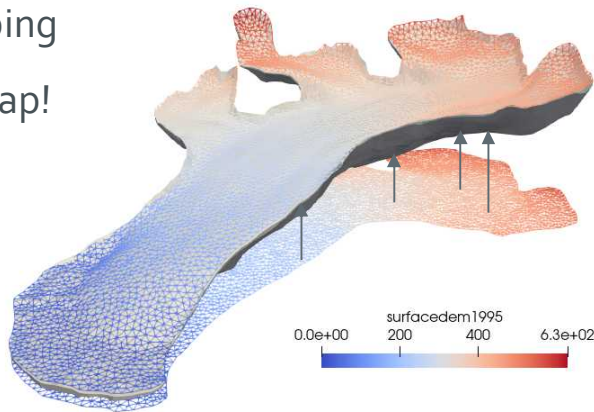
## Restart from 2D data: Mesh2MeshSolver

- We can take 2D data and interpolate it to top/bottom layers of 3D mesh
  - 2D interpolation task with z-coordinate neglected
- Makes workflow easier since the data needs to be interpolated only once to an Elmer mesh
- 2D file is read in full to all processes
  - Same restart file can be used for any number of cores!
- **We have precomputed restart files for you!**

```
Solver 1
  Exec Solver = "before all"
  Equation = "InterpSolver"
  Procedure = "Mesh2MeshSolver" "Mesh2MeshSolver"
  ! Restart is here always from a serial mesh
  Mesh = -single $restartdir
  Restart File = $restartfile
  ! We use the primary 2D mesh with local copy
  Mesh Enforce Local Copy = Logical True
  ! These are the variables for restart
  Restart Position = Integer 0
  Restart Variable 1 = String "bedrockDEM"
  Restart Variable 2 = String "surfaceDEM1995"
  ! Ensures that we perform interpolation
  ! on plane
  Interpolation Passive Coordinate = Integer 3
End
```

## Utilizing extruded structure: StructuredMeshMapper

- The shape of the mesh needs to be accommodated
  - Bottom of ice follows bedrock
  - Top of ice follows ice surface
- This could be done using generic 3D techniques
  - MeshSolve (version of linear elasticity equation)
  - Expensive and unnecessary!
- We can apply to each vertically extruded node 1D mapping
- Very cheap!



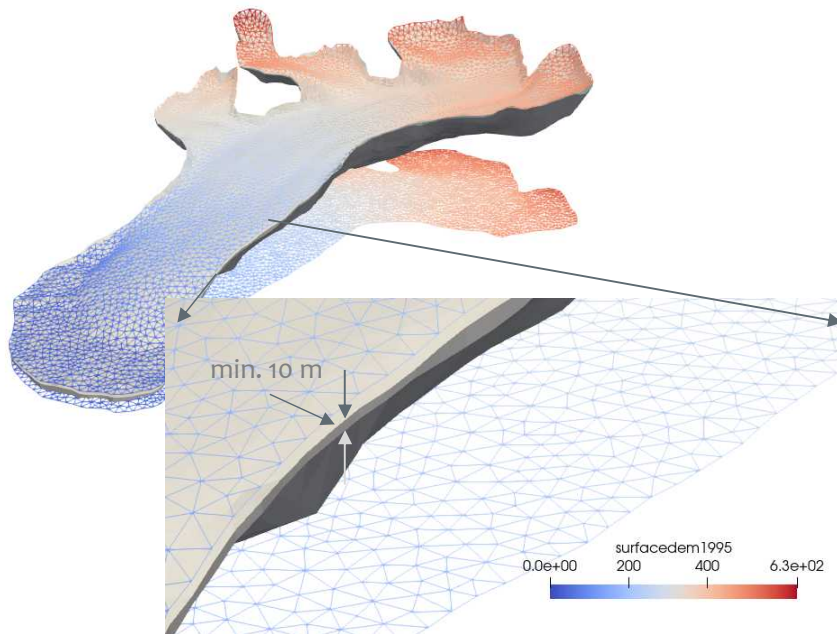
```

! Maps the constant-thickness mesh
! between given bedrock and surface topology
Solver 2
  Exec Solver = "before simulation"
  Equation = "MapCoordinate"
  Procedure = "StructuredMeshMapper" "StructuredMeshMapper"
  Active Coordinate = Integer 3
  Displacement Mode = Logical False
  Correct Surface = Logical True
  Minimum Height = Real 10.0
  Correct Surface Mask = String "Glaciated"
  Dot Product Tolerance = 1.0e-3
  ! Allocate some fields here
  Variable = MeshUpdate
  Exported Variable 1 = "bedrockDEM"
  Exported Variable 1 Mask = String "BedRock"
  Exported Variable 2 = "surfaceDEM1995"
  Exported Variable 2 Mask = String "Surface"
End

```

## Utilizing extruded structure: StructuredMeshMapper

- Imposing a minimum extrusion depth



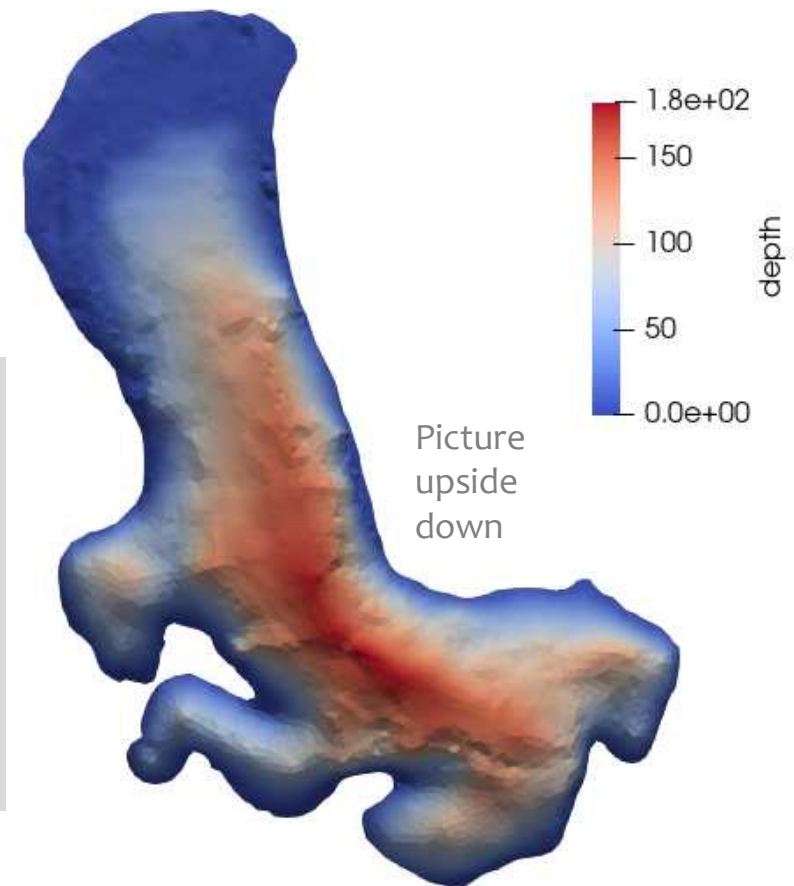
```

! Maps the constant-thickness mesh
! between given bedrock and surface topology
Solver 2
  Exec Solver = "before simulation"
  Equation = "MapCoordinate"
  Procedure = "StructuredMeshMapper" "StructuredMeshMapper"
  Active Coordinate = Integer 3
  Displacement Mode = Logical False
  Correct Surface = Logical True
  Minimum Height = Real 10.0
  Correct Surface Mask = String "Glaciated"
  Dot Product Tolerance = 1.0e-3
  ! Allocate some fields here
  Variable = MeshUpdate
  Exported Variable 1 = "bedrockDEM"
  Exported Variable 1 Mask = String "BedRock"
  Exported Variable 2 = "surfaceDEM1995"
  Exported Variable 2 Mask = String "Surface"
End
  
```

## Using extruded structure for mapping: StructuredProjectToPlane

- We may perform various operations
- Along the extruded 1D (vertical) lines
  - Computation of height & depth
  - Computation of integrals over the depth etc.

```
! Computes height and depth assuming an
! extruded mesh.
Solver 3
  Exec Solver = "before simulation"
  Equation = "HeightDepth"
  Procedure = "StructuredProjectToPlane" \ ↓
           "StructuredProjectToPlane" ←
  Active Coordinate = Integer 3
  Operator 1 = depth
  Operator 2 = height
End
```



## Optimized Stokes solver: IncompressibleNSVec

- Legacy module `FlowSolve` is one of the oldest in Elmer
  - Has a lot of extra baggage
  - Cannot ideally utilize modern CPU architectures
- `IncompressibleNSVec`:
  - Includes vectorization and threading
  - Takes use of code modernization in many places
  - Unfortunately vectorization and threading make the code less readable
- Performance boost depends heavily on the length of the vectors = Number of Gaussian integration points

```

!
      dBasisdxVec(1:ngp,1:ntot,i), dBasisdxVec(1:ngp,1:ntot,j), weight_c, stif
do(1:ntot,1:ntot,i,j)
  END DO
END DO
END IF

IF (GradPVersion) THEN
! b(u,q) = (u, grad q) part
DO i = 1, dim
  CALL LinearForms_UdotV(ngp, ntot, elemdim, &
    BasisVec, dBasisdxvec(:, :, i), detJVec, stifford(:, :, i, dofs))
  StiffOrd(:, :, dofs, i) = transpose(stifford(:, :, i, dofs))
END DO
ELSE
DO i = 1, dim
  CALL LinearForms_UdotV(ngp, ntot, elemdim, &
    dBasisdxVec(:, :, i), BasisVec, -detJVec, StiffOrd(:, :, i, dofs))
  StiffOrd(:, :, dofs, i) = transpose(stifford(:, :, i, dofs))
END DO
END IF

! Masses (use symmetry)
! Compute bilinear form G=G+(alpha u, u) = u .dot. (grad u)
IF ( .NOT. StokesFlow ) THEN
  CALL LinearForms_UdotU(ngp, ntot, elemdim, BasisVec, DetJVec, VelocityMass, rho
sec)

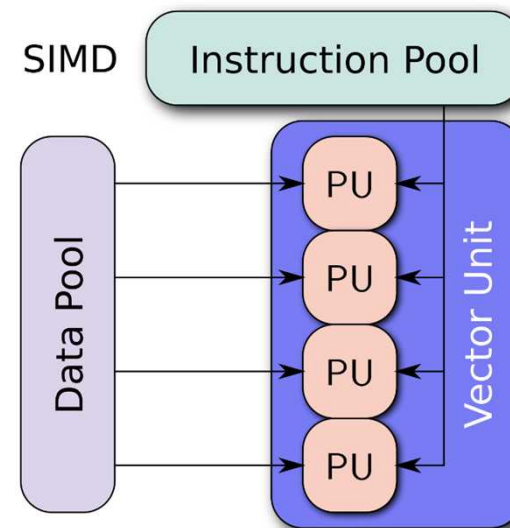
! Scatter to the usual local mass matrix
DO i = 1, dim
  mass(i::dofs, i::dofs) = mass(i::dofs, i::dofs) + VelocityMass(1:ntot, 1:ntot)
END DO
!CALL LinearForms_UdotU(ngp, ntot, elemdim, BasisVec, DetJVec, PressureMass, -ka
ppavec)

!mass(dofs::dofs, dofs::dofs) = mass(dofs::dofs, dofs::dofs) + PressureMass(1:nt
ot, 1:ntot)
U:--- IncompressibleNSVec.F90 28% L370 Git-devel (F90 AC Abbrev)

```

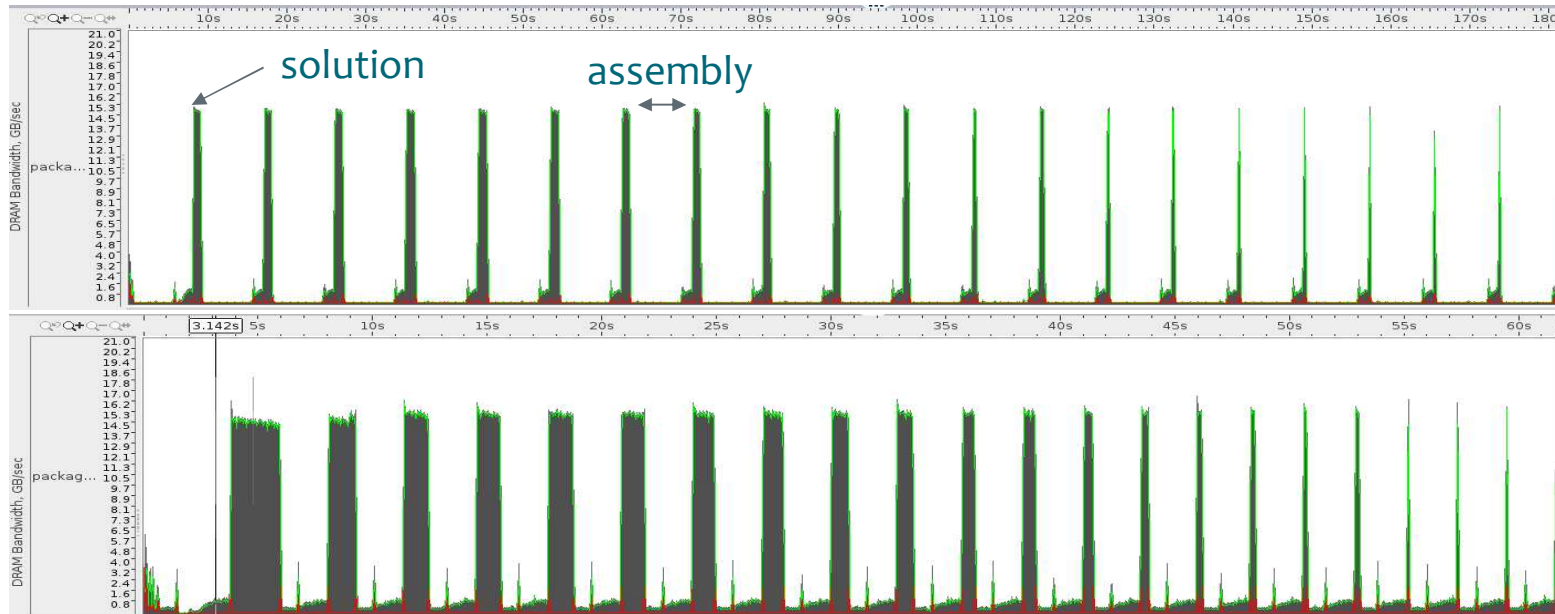
## Optimized Stokes solver: `IncompressibleNSVec`

- New CPU architectures use vector units (SIMD) to do fast computations (AVX2/AVX512)
- Cache misses (=memory performance) governed by data-layout
  - If you (on an Intel chip) ignore this, you easily loose  $\frac{3}{4}$  of your performance by not utilizing whole cache line
- Until quite recently, assembly procedures in Elmer did not utilize SIMD and did not have a cache-friendly data layout, neither where they threaded (OMP)
- `IncompressibleNSVec` does!
- Interface to block-preconditioner functionality to increase solution efficiency
  - Or to allow for Krylov methods at all
- SIMD first step to enable code for accelerators



By Vadikus - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=39715273>

# Optimized Stokes solver: IncompressibleNSVec



Mind the different timescales!

## Comparison vectorised/legacy Solver using Intel VTune

## Optimized Stokes solver: IncompressibleNSVec

- We have to specify that this is a Stokes model
  - Inertia terms neglected
- Number of integration points affects the accuracy of discretization
  - Has significant effect on performance!
- We may use different solution techniques for linear solver
  - Iterative method
  - Direct method
  - Block preconditioning (next topic)

```
Solver 4
Equation = "Stokes-Vec"
Procedure = "IncompressibleNSVec" "IncompressibleNSSolver"
Flow Model = Stokes
! 1st iteration viscosity is constant
Constant-Viscosity Start = Logical True
! Accuracy of numerical integration (on wedges)
Number of Integration Points = Integer 44 ! 21, 28, 44, 64
! Iterative approach:
Linear System Solver = Iterative
Linear System Iterative Method = "GCR"
Linear System Max Iterations = 500
Linear System Convergence Tolerance = 1.0E-08
Linear System Preconditioning = "ILU1"
Linear System Residual Output = 10
! Direct approach (as alternative to above):
!Linear System Solver = Direct
!Linear System Direct Method = MUMPS
!Non-linear iteration settings:
Nonlinear System Max Iterations = 50
Nonlinear System Convergence Tolerance = 1.0e-5
Nonlinear System Newton After Iterations = 10
Nonlinear System Newton After Tolerance = 1.0e-1
Nonlinear System Consistent Norm = True
! Nonlinear System Relaxation Factor = 1.00
End
```



## Optimized Stokes solver: IncompressibleNSVec

- We can start with constant viscosity
  - Eliminates need for initial guess – takes value of Viscosity in Material
- Nonlinear solver takes use of Newton linearization
- Starts with Picard iteration that has larger radius of convergence

```
Solver 4
Equation = "Stokes-Vec"
Procedure = "IncompressibleNSVec" "IncompressibleNSSolver"
Flow Model = Stokes
! 1st iteration viscosity is constant
Constant-Viscosity Start = Logical True
! Accuracy of numerical integration (on wedges)
Number of Integration Points = Integer 44 ! 21, 28, 44, 64
! Iterative approach:
Linear System Solver = Iterative
Linear System Iterative Method = "GCR"
Linear System Max Iterations = 500
Linear System Convergence Tolerance = 1.0E-08
Linear System Preconditioning = "ILU1"
Linear System Residual Output = 10
! Direct approach (as alternative to above):
!Linear System Solver = Direct
!Linear System Direct Method = MUMPS
!Non-linear iteration settings:
Nonlinear System Max Iterations = 50
Nonlinear System Convergence Tolerance = 1.0e-5
Nonlinear System Newton After Iterations = 10
Nonlinear System Newton After Tolerance = 1.0e-1
Nonlinear System Consistent Norm = True
! Nonlinear System Relaxation Factor = 1.00
End
```

## Optimized Stokes solver: IncompressibleNSVec

- **Material** section stays quite the same as in legacy solver
- Remember from previous slide: start from constant, Newtonian Viscosity (here  $1 \text{ MPa a}^{-1}$ ) using keyword

**Constant-Viscosity Start = Logical True**

```
Material 1
  Name = "Ice"
  Density = Real $rhoi
  ! First round viscosity with Newtonian fluid
  ! happens to give velocities of proper size
  Viscosity = Real 1.0
  ! Non-Newtonian viscosity
  Viscosity Model = String Glen
  Glen Exponent = Real 3.0 !(yes, you may also try 4.0)
  Critical Shear Rate = Real 1.0E-10
  ! Paterson value in MPa^-3a^-1
  Limit Temperature = Real -10.0
  Rate Factor 1 = Real $A1
  Rate Factor 2 = Real $A2
  Activation Energy 1 = Real $Q1
  Activation Energy 2 = Real $Q2
  Glen Enhancement Factor = Real 1.0
  Relative Temperature = Real $Tc
End
```

## Advecting scalars with ice: ParticleAdvector

- Uses ability to follow particles in the mesh
  - Initially implemented for true physical particles
- Particles are made to travel backward in time along the flowlines
- Values may be integrated along the path or registered at the initial location



```

Solver 5
Equation = ParticleAdvector
Procedure = "ParticleAdvector" "ParticleAdvector"
! Initialize particles at center of elements
Advect Elemental = Logical True
! Timestepping strategy
Particle Dt Constant = Logical False
Max Timestep Intervals = Integer 1000
Timestep Unisotropic Courant Number = 0.25
Max Timestep Size = 1.0e3
Max Integration Time = Real 1.0e4
! Integration forward in time
Runge Kutta = Logical False
Velocity Gradient Correction = Logical True
Velocity Variable Name = String "Flow Solution"
! The internal variables for this solver
Variable 1 = String "Particle Distance"
Variable 2 = String "Particle Time Integral"
! The field variables being advected
Variable 3 = String "Coordinate 1"
Result Variable 3 = String "Advected Z"
End
  
```

## Running initial case

- In serial:

```
ElmerSolver mlb.sif
```

- In parallel, here with 2 processes:

```
ElmerGrid 2 2 outline62_lc50 -partdual -metiskway 2
```

```
mpirun -np 2 ElmerSolver_mpi mlb.sif
```

- Try both of the above and check the timings at the end of the run

## Postprocessing

- Load the case into ParaView
- Display the advected properties
  - which also provide a nice way to determine which tributary the ice comes from
  - This structure is also reflected in the surface of the glacier (compare to picture)



Picture: Midtre Lovénbreen in 1999 (taken by Michael Hambrey)  
Source: [https://wgms.ch/products\\_ref\\_glaciers/midtre-lovenbreen-svalbard/](https://wgms.ch/products_ref_glaciers/midtre-lovenbreen-svalbard/)

## Block-preconditioner in IncompressibleNSVec

- In parallel runs a central challenge is to have good **preconditioners** that work in parallel
- This problem is increasingly difficult for PDEs with vector fields
  - Navier-Stokes, elasticity, acoustics,...
  - Strongly coupled multi-physics problems
- **Preconditioner** need not to be just a matrix, it **can be a procedure!**
- Use as **block-preconditioner** a procedure where the components are solved one-by-one and the solution is used as a search direction in an outer Krylov method
- Number of outer iterations may be shown to be bounded
- Individual blocks may be solved with optimally scaling methods
  - E.g. multilevel methods

### What is a preconditioner?:

Instead of solving

$$\mathbf{K} \cdot \mathbf{x} = \mathbf{b}$$

Identify a preconditioner,  $\mathbf{P}$ ,

which makes solution of

$$\mathbf{K}\mathbf{P}^{-1} \cdot \mathbf{x}' = \mathbf{b}$$

$$\text{with } \mathbf{x}' = \mathbf{P} \cdot \mathbf{x}$$

(much) easier than the original problem.

## Block-preconditioner in IncompressibleNSVec

- Utilizing block-preconditioner

```
Solver 4
Equation = "Stokes-Vec"
Procedure = "IncompressibleNSVec" "IncompressibleNSSolver"
Flow Model = Stokes
! 1st iteration viscosity is constant
Constant-Viscosity Start = Logical True
! Accuracy of numerical integration (on wedges)
Number of Integration Points = Integer 44 ! 21, 28, 44, 64
! Iterative approach using BPC:
include "linsys/block4_gcr.sif"
!Non-linear iteration settings:
Nonlinear System Max Iterations = 50
Nonlinear System Convergence Tolerance = 1.0e-5
Nonlinear System Newton After Iterations = 10
Nonlinear System Newton After Tolerance = 1.0e-1
Nonlinear System Consistent Norm = True
! Nonlinear System Relaxation Factor = 1.00
End
```

- Given a block system:

$$\begin{bmatrix} \mathbf{K}_{11} & \cdots & \mathbf{K}_{1N} \\ & \ddots & \\ \mathbf{K}_{N1} & \cdots & \mathbf{K}_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}$$

- Preconditioner is operator which produces new search directions,  $\mathbf{s}$ , for  $\mathbf{x}$
- Block-Gauss-Seidel or Block Jacobi

$$P = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{K}_{21} & \mathbf{K}_{22} & \mathbf{0} & \cdots \\ \cdots & & & \end{bmatrix} \quad P = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{K}_{22} & \mathbf{0} & \cdots \\ \cdots & & & \end{bmatrix}$$

- Minimization of residual

$$r = \|\mathbf{b} - \mathbf{K} \cdot \mathbf{x}^{(k)}\|$$

with outer GCR loop over space

$$\mathcal{V}_k = \mathbf{x}^{(0)} + \text{span}\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \dots, \mathbf{s}^{(k)}\}$$

## Block-preconditioner in IncompressibleNSVec

- Recommended "natural" outer iterative method is GCR

```

k = 0
r(k) = f - Ku(k)
while (||r(k)|| < TOL||f|| and k < m)
  Generate the search direction s(k+1)
  v(k+1) = Ks(k+1)
  do j = 1, k
    v(k+1) = v(k+1) - ⟨v(j), v(k+1)⟩v(j)
    s(k+1) = s(k+1) - ⟨v(j), v(k+1)⟩s(j)
  end do
  v(k+1) = v(k+1) / ||v(k+1)||
  s(k+1) = s(k+1) / ||v(k+1)||
  u(k+1) = u(k) + ⟨v(k+1), r(k)⟩s(k+1)
  r(k+1) = r(k) - ⟨v(k+1), r(k)⟩v(k+1)
  k = k + 1
end while

```

```

Linear System Solver = "Block"
Block Gauss-Seidel = Logical True
Block Matrix Reuse = Logical False
Block Scaling = Logical False
Block Preconditioner = Logical True
! Default is [1 2 3 4]
Block Structure(4) = Integer 1 2 3 4
! Block Order(2) = Integer 2 1
! Linear System Scaling = False
! Linear system solver for outer loop
!-----
Outer: Linear System Solver = "Iterative"
Outer: Linear System Iterative Method = GCR
Outer: Linear System GCR Restart = 250
Outer: Linear System Residual Output = 1
Outer: Linear System Max Iterations = 200
Outer: Linear System Abort Not Converged = False
Outer: Linear System Convergence Tolerance = 1e-8

```



## Block-preconditioner in IncompressibleNSVec

- Stokes problem block-structure

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix}$$

from stabilization

- Optimal pre-conditioner with Pressure-Schur complement,  $\mathbf{Q}$ ,

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$$

- Either split velocity block,  $\mathbf{A}$ , into 3x3 (recommended!)

```
Block Structure(4)=Integer 1 2 3 4
```

- Or as one

```
Block Structure(4)=Integer 1 1 1 4
```

```
Linear System Solver = "Block"
Block Gauss-Seidel = Logical True
Block Matrix Reuse = Logical False
Block Scaling = Logical False
Block Preconditioner = Logical True
! Default is [1 2 3 4]
Block Structure(4) = Integer 1 2 3 4
! Block Order(2) = Integer 2 1
! Linear System Scaling = False
! Linear system solver for outer loop
!-----
Outer: Linear System Solver = "Iterative"
Outer: Linear System Iterative Method = GCR
Outer: Linear System GCR Restart = 250
Outer: Linear System Residual Output = 1
Outer: Linear System Max Iterations = 200
Outer: Linear System Abort Not Converged = False
Outer: Linear System Convergence Tolerance = 1e-8
```

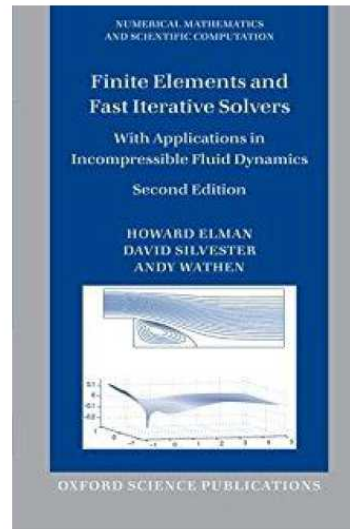
## Block-preconditioner in IncompressibleNSVec

- Inner solutions (of blocks)
- Blocks 1,2,3 here associated with velocity components 1,2,3

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{0} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_3 & \\ & \mathbf{0} & & \mathbf{Q} \end{bmatrix}$$

- Block 4 associated with pressure (preconditioned with scaled mass matrix is suggested by Elman)

$$\mathbf{Q} = \mu^{-1} \mathbf{1}$$



```

block 11: Linear System Convergence Tolerance = $blocktol
block 11: Linear System Solver = "iterative"
block 11: Linear System Scaling = false
block 11: Linear System Preconditioning = ilu
block 11: Linear System Residual Output = 100
block 11: Linear System Max Iterations = 500
block 11: Linear System Iterative Method = idrs
  
```

```

block 22: Linear System Convergence Tolerance = $blocktol
block 22: Linear System Solver = "iterative"
block 22: Linear System Scaling = false
block 22: Linear System Preconditioning = ilu
block 22: Linear System Residual Output = 100
block 22: Linear System Max Iterations = 500
block 22: Linear System Iterative Method = idrs
  
```

```

block 33: Linear System Convergence Tolerance = $blocktol
block 33: Linear System Solver = "iterative"
block 33: Linear System Scaling = false
block 33: Linear System Preconditioning = ilu
block 33: Linear System Residual Output = 100
block 33: Linear System Max Iterations = 500
block 33: Linear System Iterative Method = idrs
  
```

```

block 44: Linear System Convergence Tolerance = $blocktol
block 44: Linear System Solver = "iterative"
block 44: Linear System Scaling = true
block 44: Linear System Preconditioning = ilu
block 44: Linear System Residual Output = 100
block 44: Linear System Max Iterations = 500
block 44: Linear System Iterative Method = idrs
  
```

## Run case with BPC – compare to other methods

- Running the initial case (cl75)
  - You may try to run the larger cases (cl50, cl25) – might exceed available memory
- Altering number of integration points
  - Does it have an affect on simulation results: ...,21, 28, 44, 64,..?
- Trying out different linear system strategies
  - GCR vs. block preconditioner vs. direct solver
  - `omlb_linsys.sif` contains linear system recipes with (for GCR, MUMPS and BPC)  

```
include linsys/method.sif
```
- Trying effect of Courant number in particle advection
- NB: You may want to turn off `ParticleAdvect` if testing Stokes by adding  

```
Exec Solver = never
```

into the corresponding Solver section

## Run case with BPC – compare to others

`linsys/mumps.sif`

`linsys/gcr.sif`

`linsys/block4_idrs.sif`

```

elmeruser@elmeruser-VirtualBox: ~/Work/MLB
MAIN:
ComputeChange: NS (ITER=1) (NRM,RELC): ( 1.5394608  2.0000000  ) :: stok
es-vec
ComputeChange: NS (ITER=2) (NRM,RELC): ( 1.2546285  0.20388205  ) :: stok
es-vec
ComputeChange: NS (ITER=3) (NRM,RELC): ( 1.1157424  0.11718509  ) :: stok
es-vec
ComputeChange: NS (ITER=4) (NRM,RELC): ( 1.0382387  0.71963163E-01  ) :: stok
es-vec
ComputeChange: NS (ITER=5) (NRM,RELC): ( 0.89954382  0.14314808  ) :: stok
es-vec
ComputeChange: NS (ITER=6) (NRM,RELC): ( 0.90819142  0.95673375E-02  ) :: stok
es-vec
ComputeChange: NS (ITER=7) (NRM,RELC): ( 0.90829725  0.11652083E-03  ) :: stok
es-vec
ComputeChange: NS (ITER=8) (NRM,RELC): ( 0.90829768  0.46974973E-06  ) :: stok
es-vec
ComputeChange: SS (ITER=1) (NRM,RELC): ( 0.90829768  2.0000000  ) :: stok
es-vec
ElmerSolver: *** Elmer Solver: ALL DONE ***
ElmerSolver: The end
SOLVER TOTAL TIME(CPU,REAL):      141.95      145.15
ELMER SOLVER FINISHED AT: 2021/10/21 13:30:51
elmeruser@elmeruser-VirtualBox:~/Work/MLB

```

```

elmeruser@elmeruser-VirtualBox: ~/Work/MLB
gcr:  80  0.9399E-08  0.8017E-08
ComputeChange: NS (ITER=6) (NRM,RELC): ( 0.90819143  0.95673357E-02  ) :: stok
es-vec
gcr:  10  0.3467E-04  0.2010E-04
gcr:  20  0.9442E-05  0.3441E-05
gcr:  30  0.3254E-05  0.1983E-05
gcr:  40  0.6893E-06  0.3779E-06
gcr:  50  0.1072E-06  0.5376E-07
gcr:  60  0.1577E-07  0.1234E-07
ComputeChange: NS (ITER=7) (NRM,RELC): ( 0.90829727  0.11653190E-03  ) :: stok
es-vec
gcr:  10  0.2081E-04  0.8096E-05
gcr:  20  0.2748E-05  0.1997E-05
gcr:  30  0.1799E-06  0.1627E-06
gcr:  40  0.1431E-07  0.9471E-08
ComputeChange: NS (ITER=8) (NRM,RELC): ( 0.90829765  0.41885449E-06  ) :: stok
es-vec
ComputeChange: SS (ITER=1) (NRM,RELC): ( 0.90829765  2.0000000  ) :: stok
es-vec
ElmerSolver: *** Elmer Solver: ALL DONE ***
ElmerSolver: The end
SOLVER TOTAL TIME(CPU,REAL):      70.78      73.36
ELMER SOLVER FINISHED AT: 2021/10/21 13:34:39
elmeruser@elmeruser-VirtualBox:~/Work/MLB

```

```

elmeruser@elmeruser-VirtualBox: ~/Work/MLB
gcr:  10  0.3410E-04  0.1945E-04
gcr:  20  0.9500E-05  0.5621E-05
gcr:  30  0.2611E-05  0.1484E-05
gcr:  40  0.6686E-06  0.3763E-06
gcr:  50  0.2194E-06  0.9995E-07
gcr:  60  0.6400E-07  0.3815E-07
gcr:  70  0.1991E-07  0.1015E-07
ComputeChange: NS (ITER=7) (NRM,RELC): ( 0.90833252  0.11646075E-03  ) :: stok
es-vec
gcr:  10  0.1607E-04  0.8354E-05
gcr:  20  0.4514E-05  0.2901E-05
gcr:  30  0.1216E-05  0.5847E-06
gcr:  40  0.3112E-06  0.2090E-06
gcr:  50  0.8691E-07  0.4422E-07
gcr:  60  0.2457E-07  0.1275E-07
ComputeChange: NS (ITER=8) (NRM,RELC): ( 0.90833294  0.46707815E-06  ) :: stok
es-vec
ComputeChange: SS (ITER=1) (NRM,RELC): ( 0.90833294  2.0000000  ) :: stok
es-vec
ElmerSolver: *** Elmer Solver: ALL DONE ***
ElmerSolver: The end
SOLVER TOTAL TIME(CPU,REAL):      86.16      89.66
ELMER SOLVER FINISHED AT: 2021/10/21 14:13:47
elmeruser@elmeruser-VirtualBox:~/Work/MLB

```

With 21 IPs

- Mind, that BPC on small cases (here we ran the 75 m mesh) are not necessarily performing faster than the same GCR with a simple pre-conditioner (e.g., ILU)
- Nevertheless, for many cases, Krylov methods with not-optimal pre-conditioners do not work at all
- Direct methods (MUMPS, cPardiso) stop scaling beyond a few hundred cores

## Computing steady state mass balance

- Add solver for computing **emergence velocity**

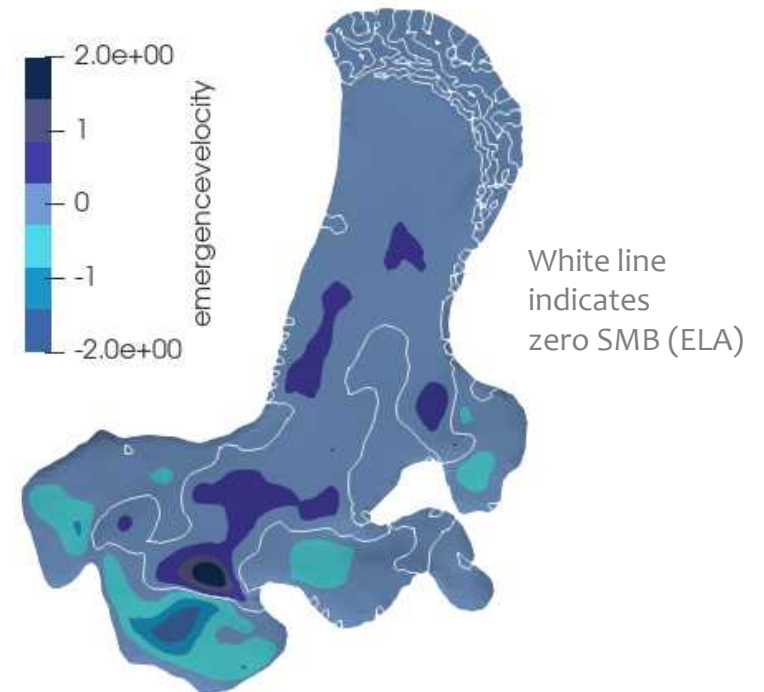
```
! Computing emergence velocity
Solver 6
  Equation = "SMB"
  Exec Solver = "After Timestep"
  Procedure = "ElmerIceSolvers" "GetEmergenceVelocity"
  Variable = -dofs 1 EmergenceVelocity
End
```

- Needs also surface normal to be computed
- Needs to be run on free-surface boundary, only

```
Body 2
  Name = "surface"
  Equation = 2
  Material = 1 ! Not used, but needed
End
Equation 2
  Name = "Surface Equations"
  Active Solvers(1) = 6
  Convection = String "Computed"
  Flow Solution Name = String "Flow Solution"
End
Boundary Condition 4 !free surface boundary
  Body ID = 2
...
End
```

Run with `mpirun -np 2 ElmerSolver mlb emergence.sif`

$$v_{em} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} + w$$



## Prognostic relaxation run

- The kinematic BC at the free surface

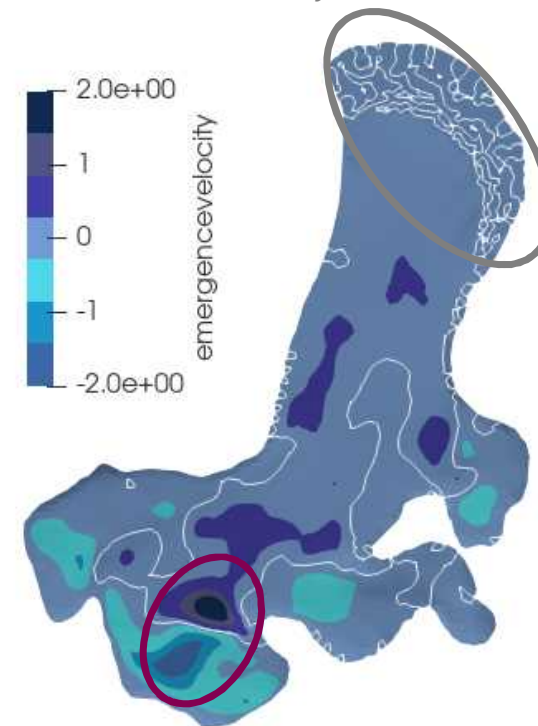
$$\frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} + v \frac{\partial h}{\partial y} - w = a_{\perp}$$

- In case of equilibrium ( $\frac{\partial h}{\partial t} \approx 0$ ) reduces to

$$u \frac{\partial h}{\partial x} + v \frac{\partial h}{\partial y} - w = -v_{em} = a_{\perp equ.}$$

- That means, if we restart a prognostic run **using the negative emergence velocity as SMB**, we can use it as a relaxation run to even out flaws in the DEMs or the interpolation of those
  - There is an obvious issue (most likely from bedrock data) with strong accumulation and ablation around the ELA (see red marker in picture)
  - Also spot the chaotic area in front of the glacier (the part kept at minimum glaciation) with partly slight positive and slight negative SMB – that will cause troubles in the long run (see grey marker in picture)

$$v_{em} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} + w$$



## Prognostic relaxation run

- Following adaptations to previous run:
  - Move from steady state to transient in Simulation-section
  - Restart from previous file in Simulation-section
  - All Exec Solver commands change from before simulation to before timestep
  
- The variable,  $Z_s$ , of the new introduced FreeSurfaceSolver (=computation of BC on surface; see next slide) is initialized to the surface DEM
- And the upper elevation in the StructuredMeshMapper is following  $Z_s$

```
Simulation
  Coordinate System = "Cartesian 3D"
  Simulation Type = "Transient"

...
  ! Time-stepping settings
  !-----
  Steady State Max Iterations = 1
  Timestepping Method = "BDF" ! implicit Euler
  BDF Order = 1
  Timestep Sizes = $1.0/52 ! 1 week
  Timestep Intervals = 52
  Output Intervals = 1

  Restart File = $restartfile
  Restart Before Initial Conditions = Logical True
  Interpolation Passive Coordinate = Integer 3
...
End
```

```
Initial Condition 1
  Zs = Equals "surfaceDEM1995"
End

Boundary Condition 4
  Body ID = 2
  Name = "surface"
  Top Surface = Equals "Zs"
...
END
```

## Prognostic relaxation run

- Free surface solver is run on Body 2 (the upper surface):
  - The keyword `Apply Dirichlet` triggers the contact problem to be evaluated. It demands
    1. `Zs Residual` to be declared as `Exported Variable`
    2. `Nonlinear System Max Iterations` to be set to a value  $>1$  (i.e., non-linear iterations are needed for the contact problem)
    3. The value `Min Zs` to be given in the `Material`-section declares the lowest possible value (here 10 m above the bedrock) for the contact problem

```
Material 1
...
  Min Zs = Variable "BedrockDem"
    Real MATC "tx + 10.0"
...
End
```

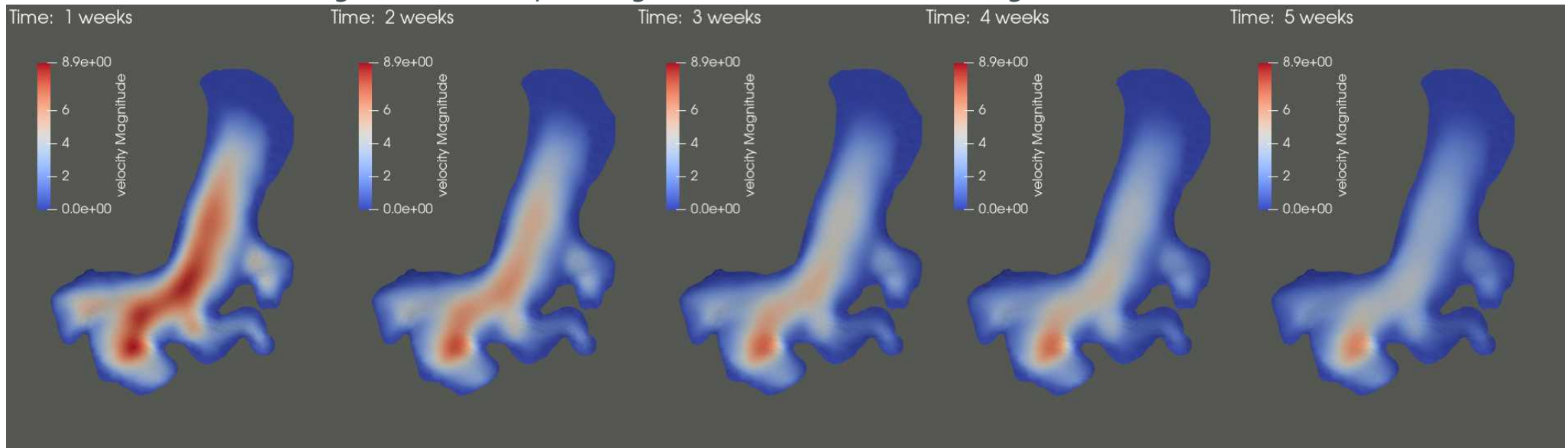
```
Solver 4
  Exec Solver = "after timestep"
  Equation = "Free Surface"
  Variable = String "Zs"
  Variable DOFs = 1
  Procedure = "FreeSurfaceSolver" "FreeSurfaceSolver"
  ! This would take the constrained points out of solution
  ! Use in serial run, only
! Before Linsolve = "EliminateDirichlet" "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-8
  Linear System Abort Not Converged = True
  Linear System Residual Output = 10
  Nonlinear System Max Iterations = 100
  Nonlinear System Convergence Tolerance = 1.0e-7
  !Nonlinear System Relaxation Factor = 0.60
  Steady State Convergence Tolerance = 1.0e-04
  ! Apply contact problem
Apply Dirichlet = Logical True
  ! needed for evaluating the contact pressure
Exported Variable 1 = -dofs 1 "Zs Residual"
  ! needed to host imported emergence velocity
  Exported Variable 2 = -dofs 1 "EmergenceVelocity"
  ! How much the free surface is relaxed (default is no
relaxation)
  ! Relaxation Factor = Real $1.0/3.0
End
```



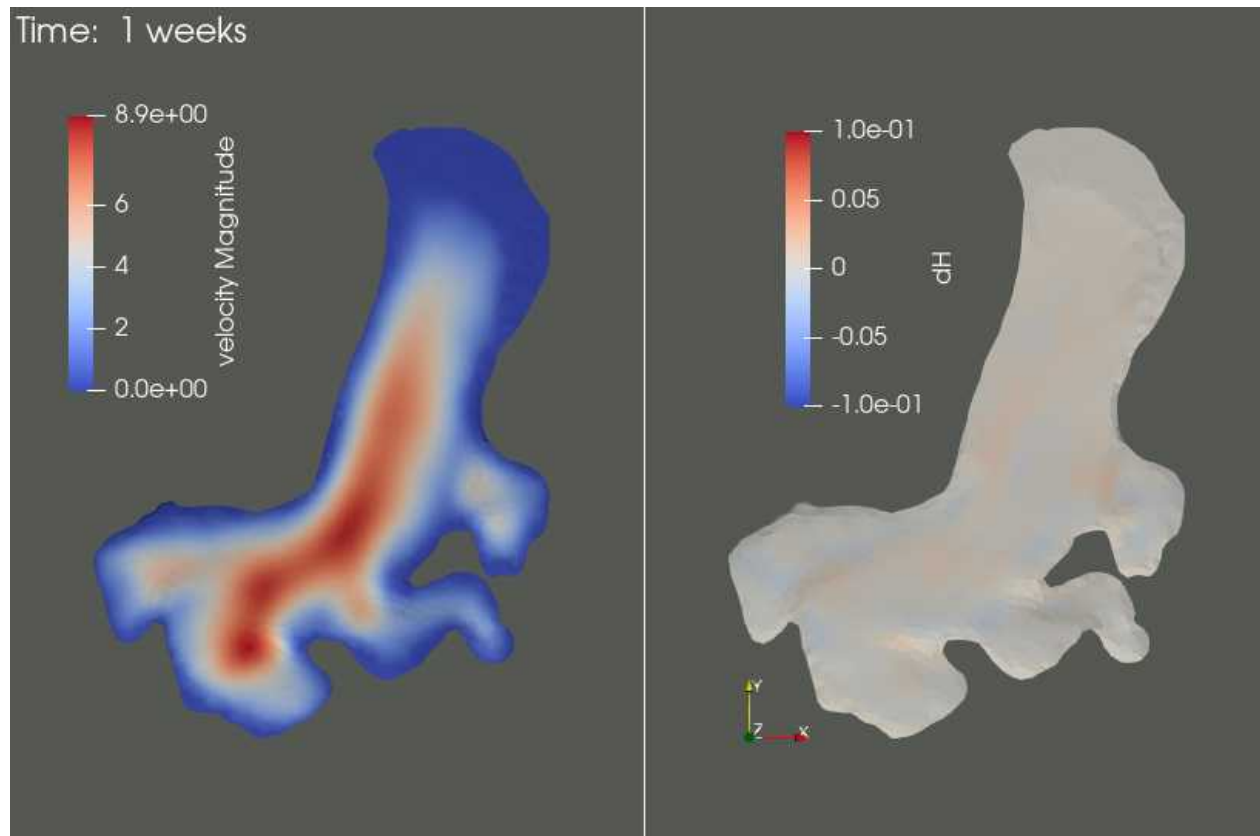
## Prognostic relaxation run

- Run with `mpirun -np 2 ElmerSolver mlb_relax.sif`

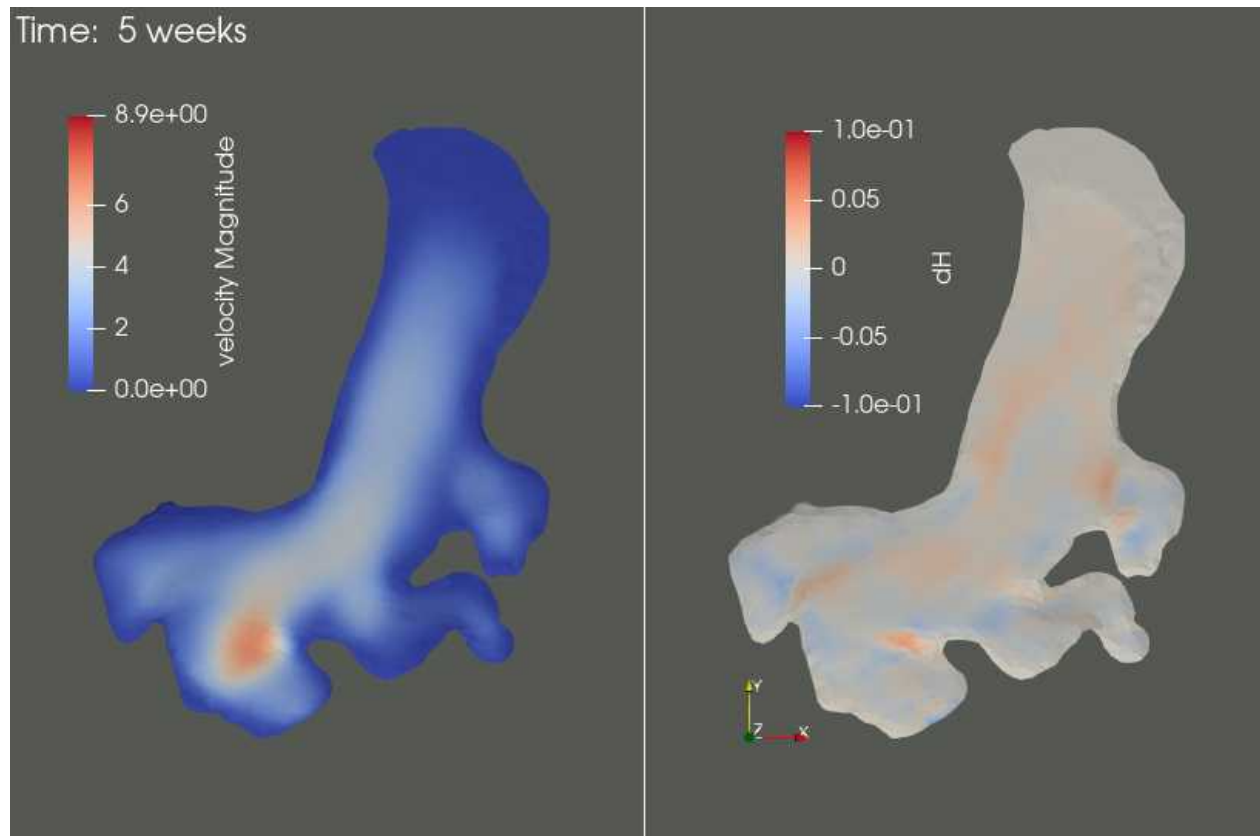
- We see a fast lowering of (the initially too high) velocities with little changes of the free surface



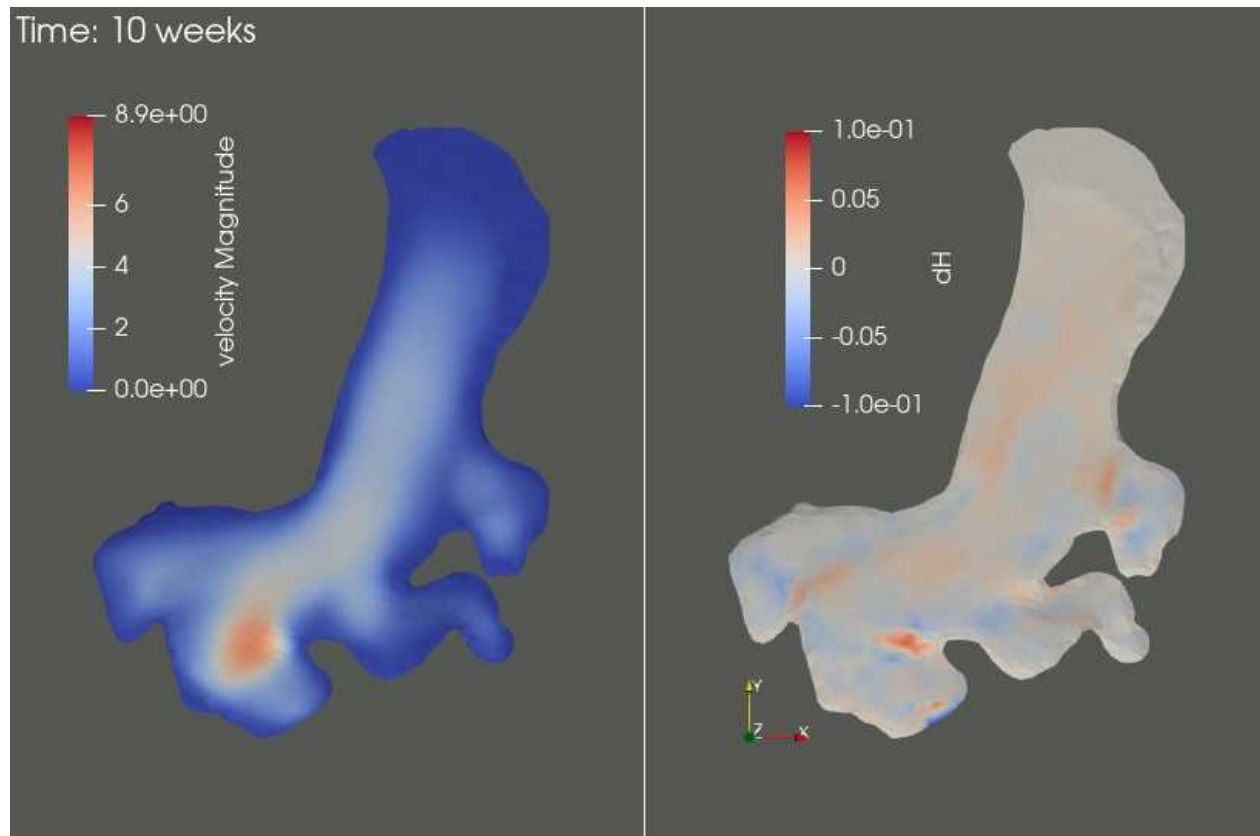
## Prognostic relaxation run



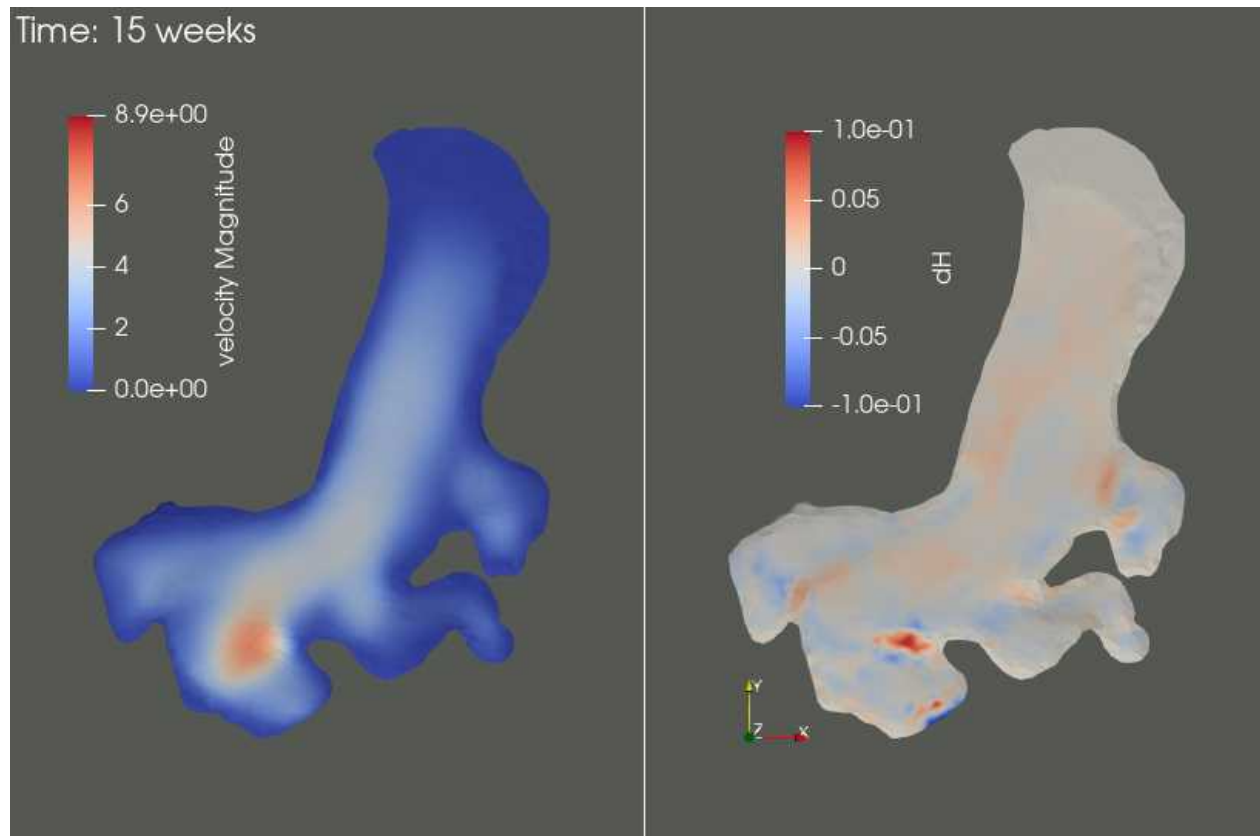
## Prognostic relaxation run



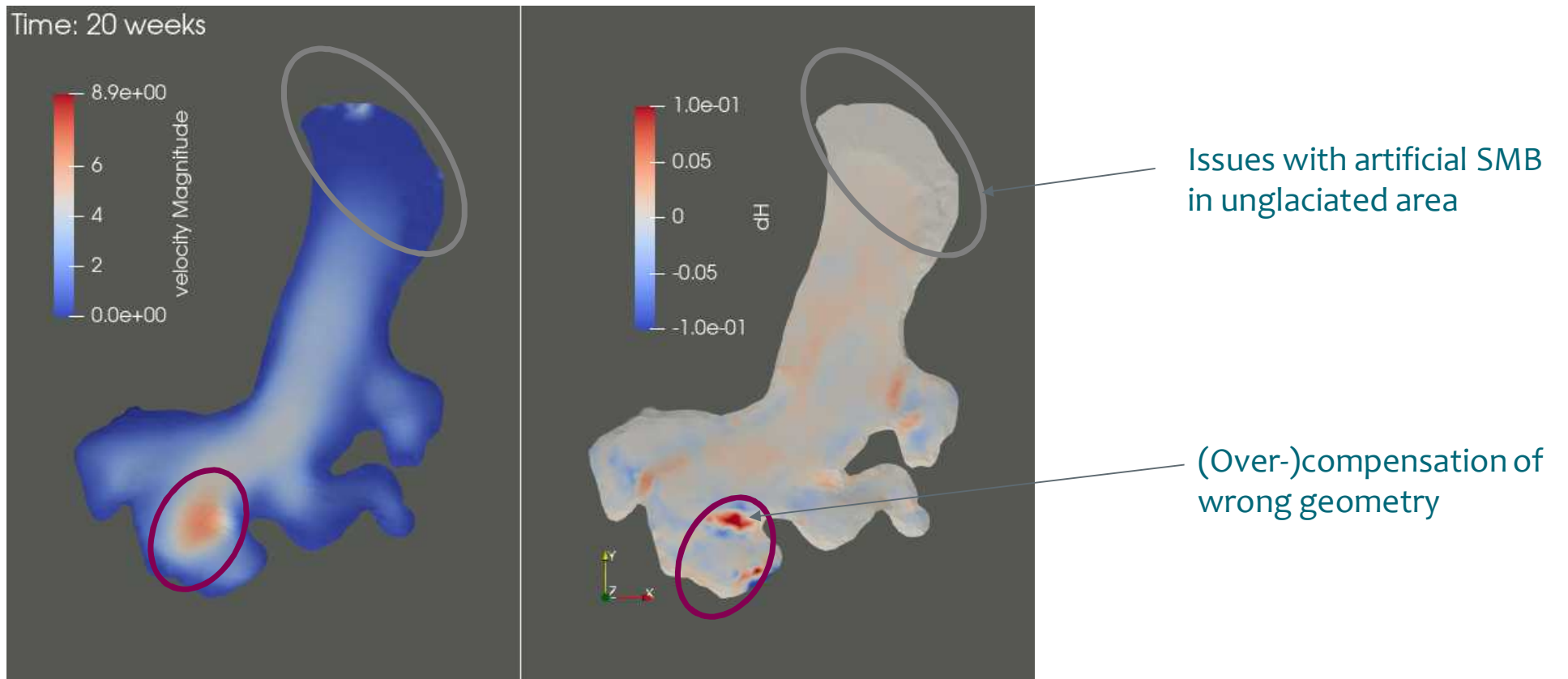
## Prognostic relaxation run



## Prognostic relaxation run



# Prognostic relaxation run



## End of session

### Things to try, if time permits

- You could take the relaxed surface and re-run the emergence velocity on it
  - Or simply include emergence velocity computation in the relaxation run and watch its change
- You could re-compute a steady state age-distribution on the relaxed geometry with the lower velocities
- You could use the equilibrium SMB (=negative emergence velocity) of the relaxed geometry and do a prognostic run with it
  - You tough might have to fix the issue with the artificial accumulation over the deglaciaded area at the front (perhaps use a function that cuts off)
  
- Thanks to **Jack Kohler** (NPI) for providing the bedrock and surface DEM