Elmer/Ice Course - MLB


50
350
650 m
3
1
2
1
Midtre Lovénbreen (MLB)
Vestfonna
Austfonna
Spitzbergen
SVALBARD
https://wgms.ch/products_ref_glaciers/midtre
-lovenbreen-svalbard/

Reconstructing Climate: 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

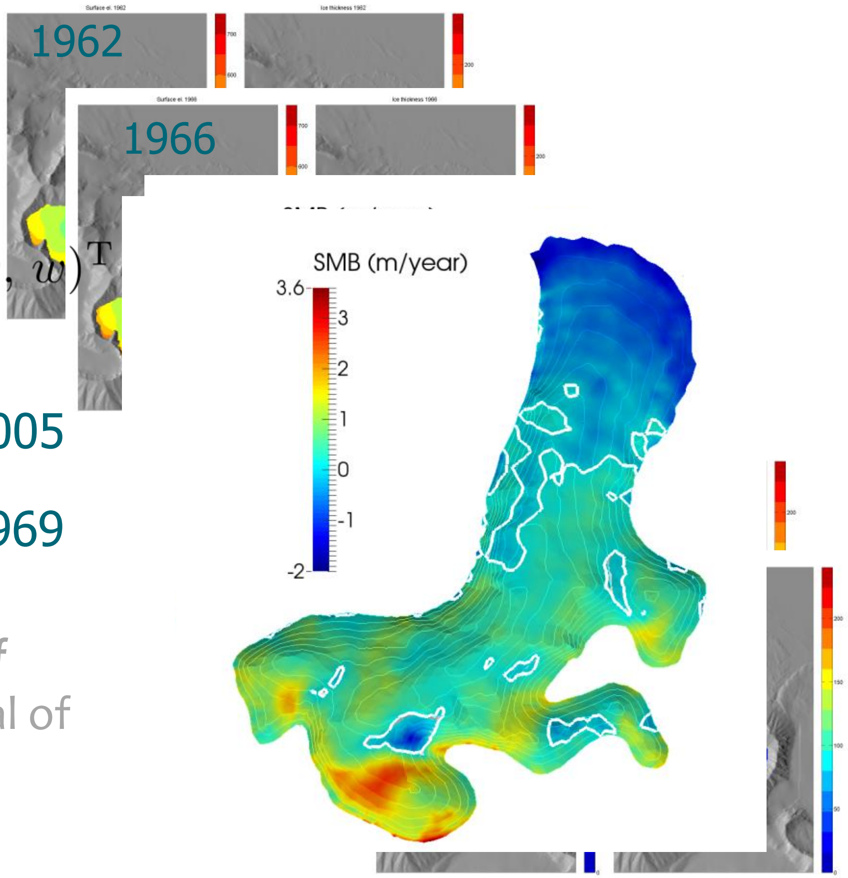
$$\frac{\partial h}{\partial t}|_{2000} = (h_{2005} - h_{1995})/11$$

- Elmer/Ice full Stokes diagnostic simulations → $\boldsymbol{u} = (u, v, w)^{\mathrm{T}}$
- Spatial distribution of SMB:

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

1995-2005

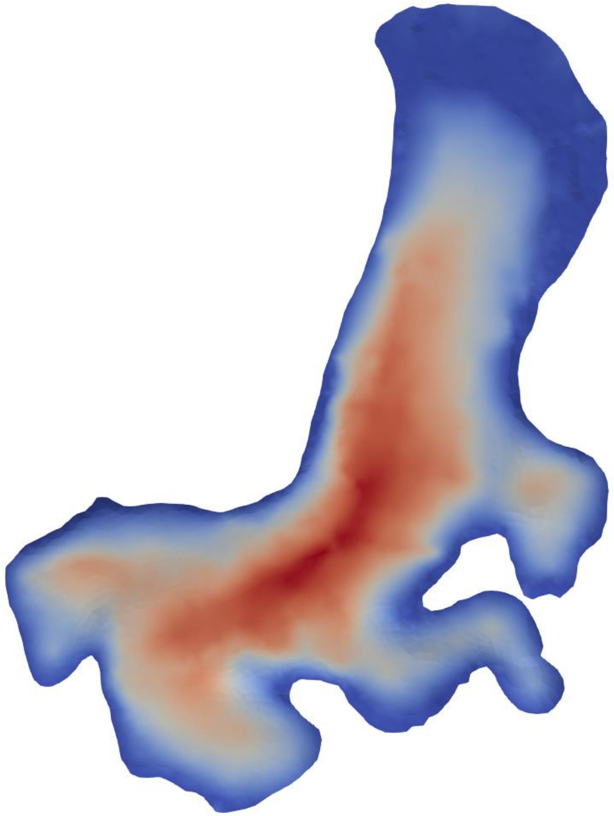
1962-1969

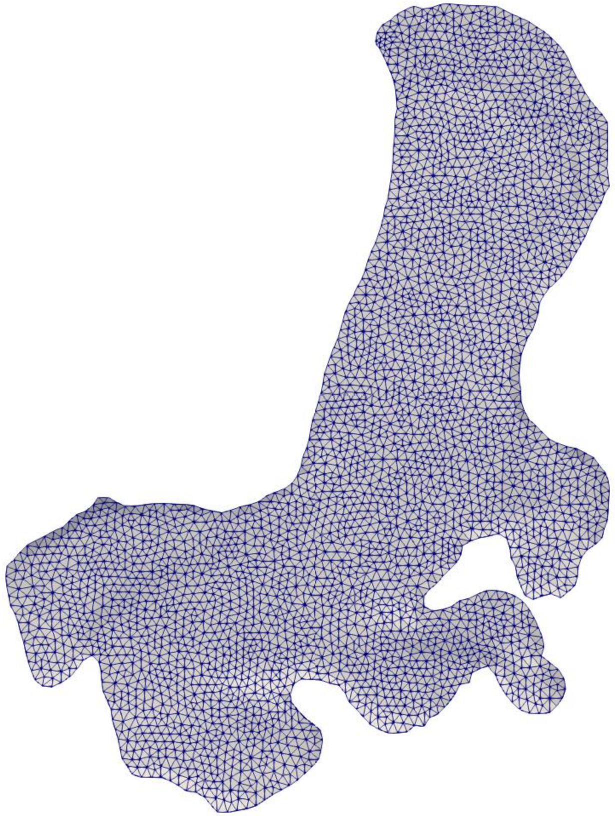


Välisuo, I., T. Zwinger and J. Kohler (2017): ***Inverse solution of surface mass balance of Midtre Lovénbreen, Svalbard***, Journal of Glaciology, 1-10, doi:10.1017/jog.2017.26.

This exercise

- We take the DEM of 1995
- We will 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
  - Semilagrangian solver for purely convective transport i.e. of age
- Users are free to try out different things
  - Solution strategies
  - Parallel runs
  - …

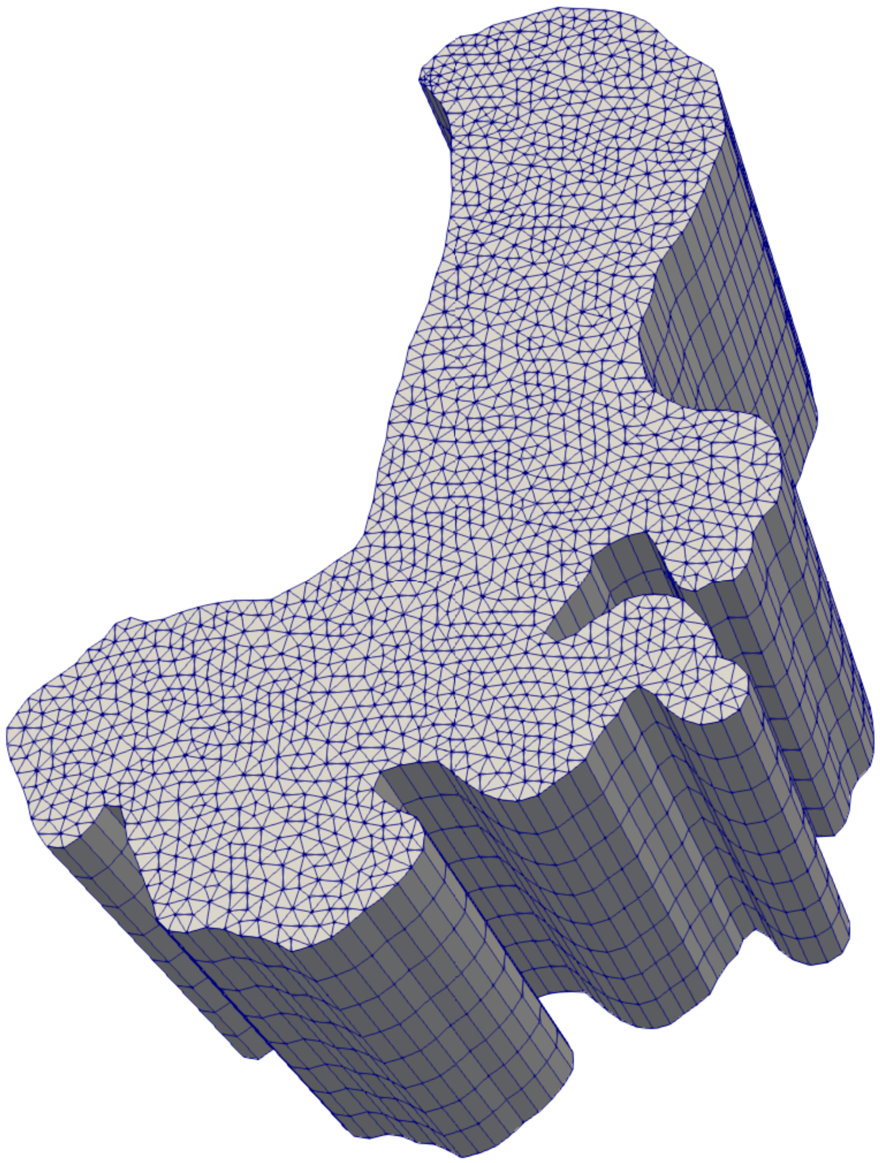




Finalizing mesh using internal extrusion

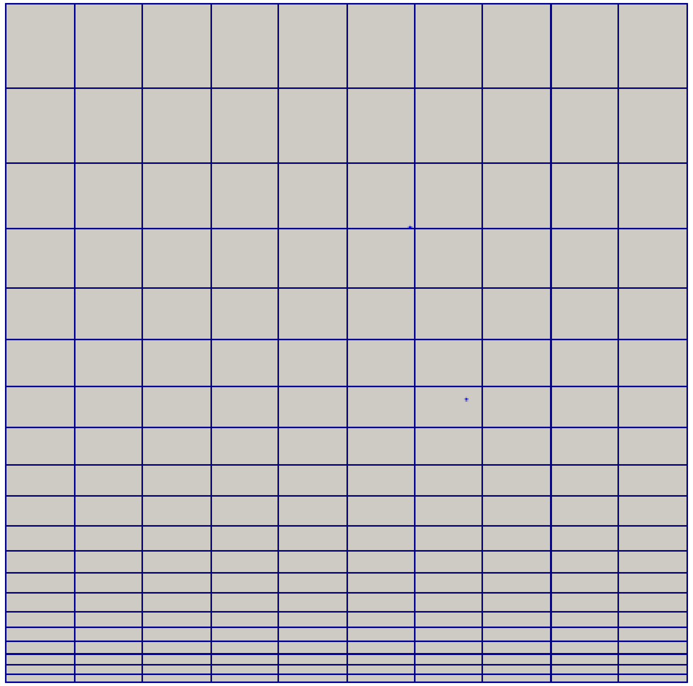
- The mesh is finalized in memory starting from 2D footprint
- Here the extruded height does not play any role
  - Mesh is further adapted to follow true bottom and top

```
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 contraint BC + offset
  - offset is set if the baseline BCs are preserved
- Top and bottom boundaries get the next free BC constraint indexes
  - Note thet 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"
```

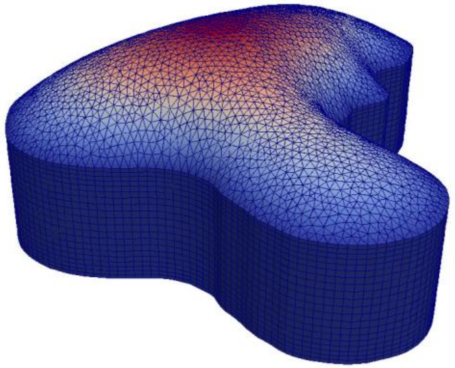
Restart from 2D data: Mesh2MeshSolver

- We can take 2D data and interpolate it to top/bottom layers of 3D mesh
  - 2D interpolatio 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 in mesh deformation: StructuredMeshMapper

- The shape of the mesh needs to be accomodated
  - 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 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" "StructuredMeshMappe

  Active Coordinate = Integer 3
  Displacement Mode = Logical False
  Correct Surface = Logical True
  Minimum Height = Real 5.0
  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
```

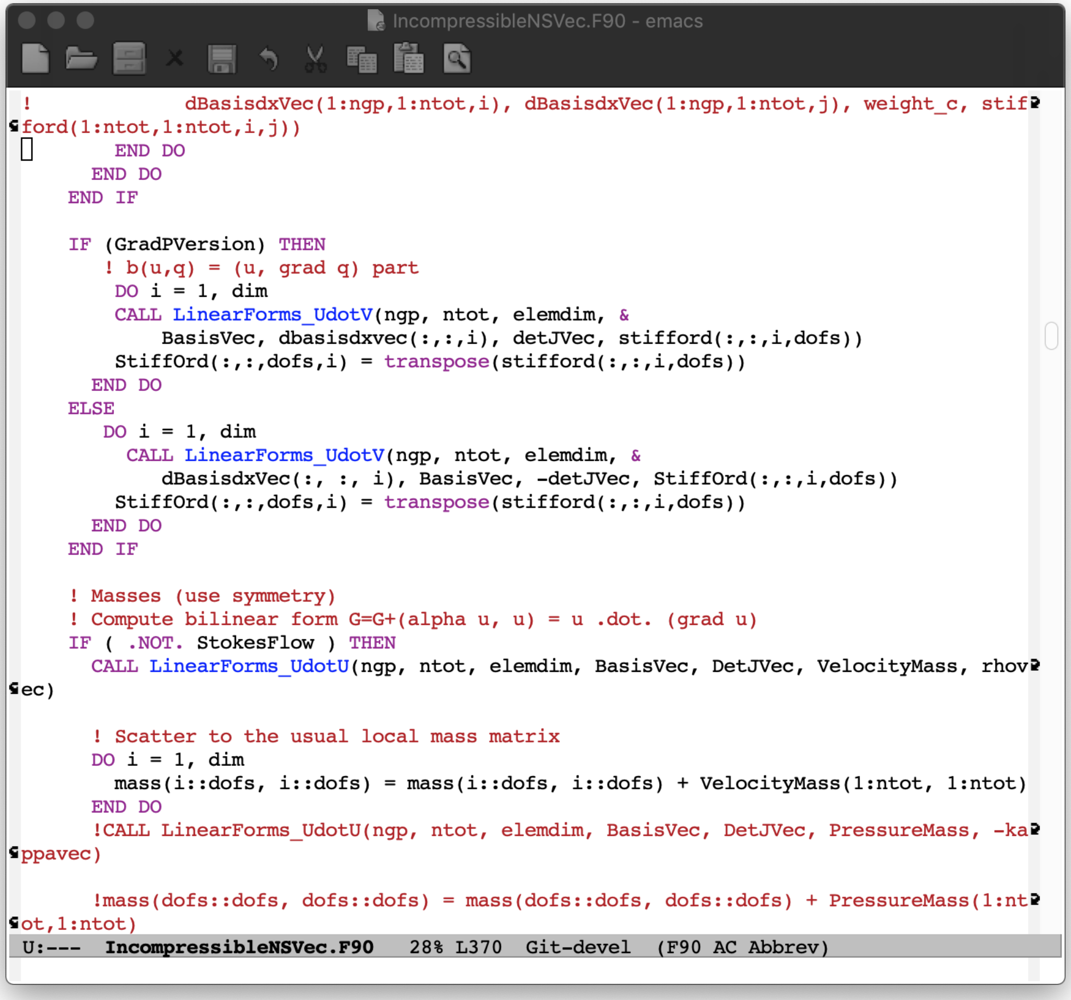
Using extruded structure for mapping: StructuredProjectToPlane

- We may perform various operations along the extruded 1D 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
```

New Stokes solver: IncompressibleNSVec

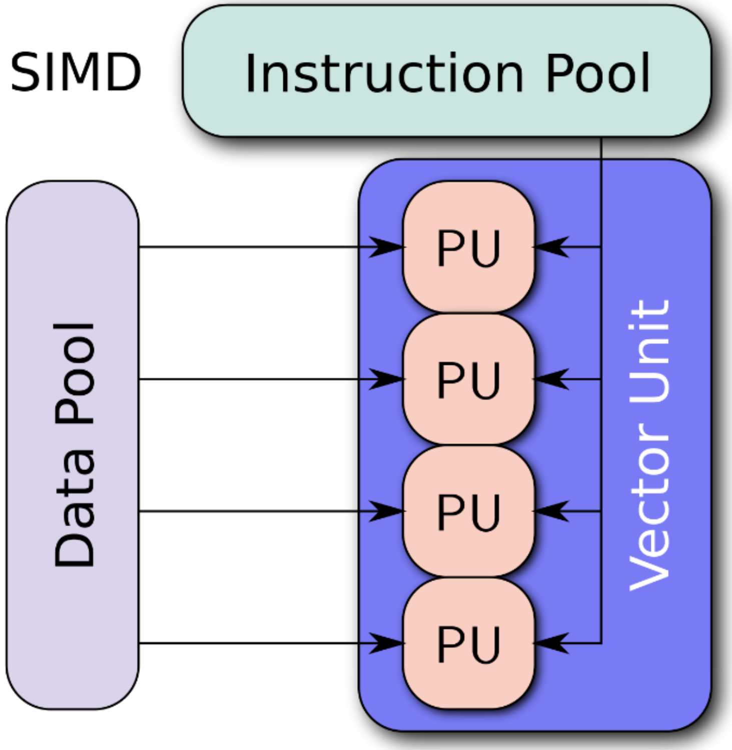
- FlowSolve is one of the oldest modules in Elmer
  - Has a lot of extra baggage
  - Cannot ideally utilize modern CPU architectures
- IncompressibleNSVec is fresh out of the oven
  - Includes vectorization and threading
  - Takes use of code modernination in many places
  - Unfortunately vectorization and threading do not make the modules prettier
- Performance boost depends heavily on the length of the vectors
  - Number of Gaussian integration points



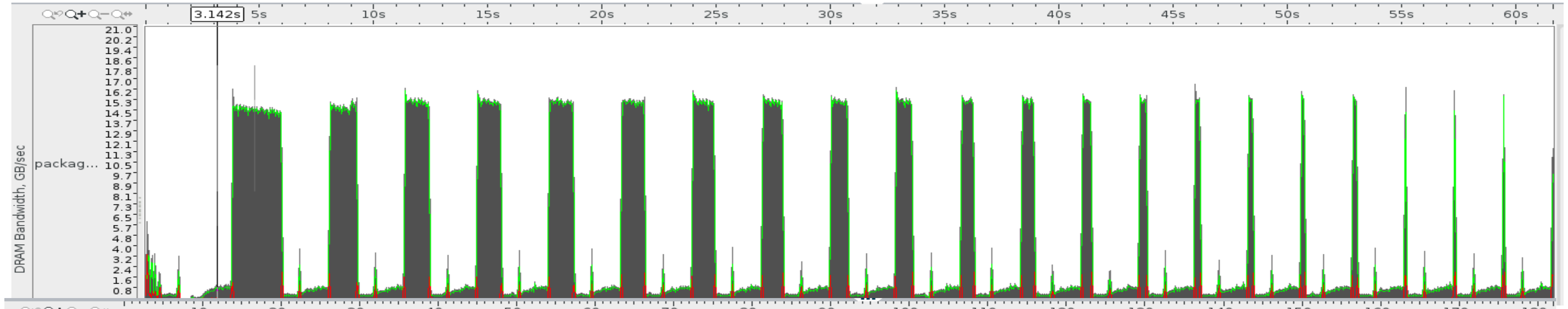
Motivation for new Stokes Solver

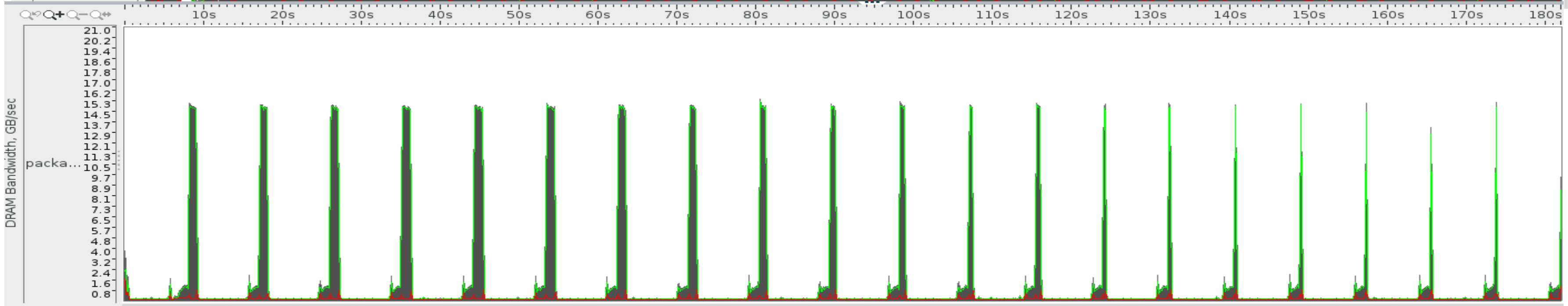
- New computer architectures use SIMD (=vector) units to do fast computations
- If you (on an Intel chip) don't utilize this, you a priori loose ¾ of your performance
- FEM: assembly = creating the matrix

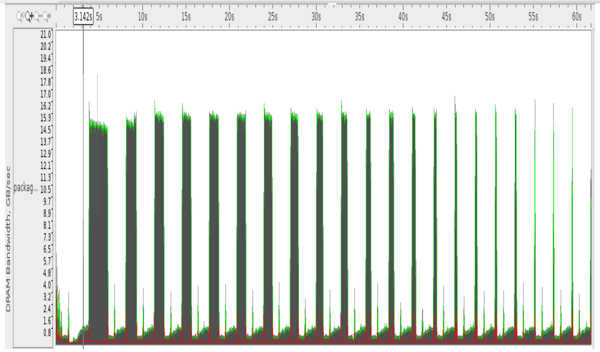
  solution = solving it
- Until recently, assembly procedures in Elmer did not utilize SIMD
- New Stokes solver does!
- Gains depend on the number of integration points



Comparison vectorised/legacy Solver using Intel VTune







Using the new Stokes Solver

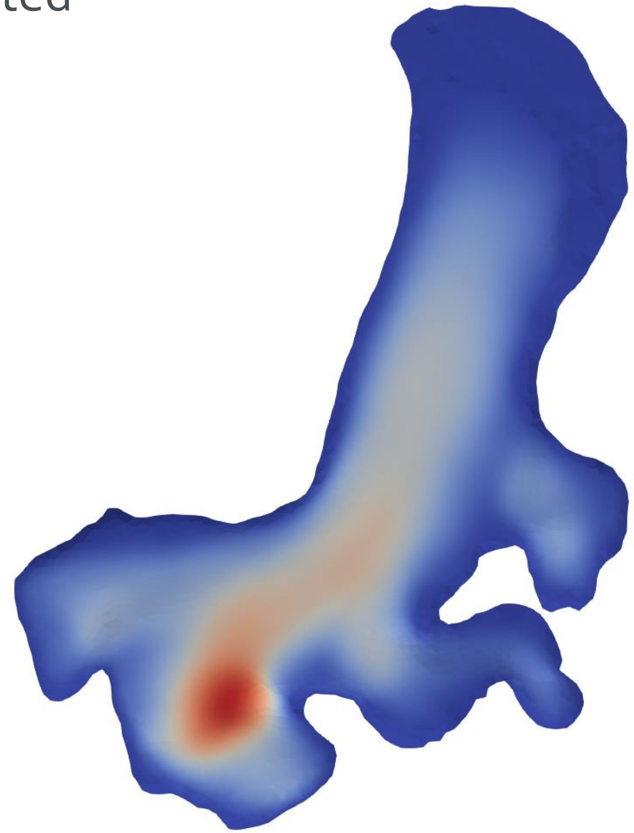
- We have to specify that this is a Stokes model
  - Inertia terms neglected
- We can start with constant viscosity
  - Eliminates need for initial guess
- Number of integration points affects the accuracy of discretization
  - Has significant effect on performance!
- We may use different solution techniques for linear solver
  - Iterative method
  - Direct metod
  - Block preconditioning (next topic)
- Nonlinear solver takes use of Newton's linearization
  - Started 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 Abort Not Converged = False
  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
```

csc

Material law for Ice

- Ice is a shear-thinning fluid and requires a complicated viscosity model
- Plain viscosity is used in the 1st solution if requested



```
Material 1
  Name = "Ice"
  Density = Real $rhoi

  ! First viscosity with newtonian fluid
  ! happens to give velocities of proper size
  Viscosity = Real 1.0

  ! Nonnewtonian viscosity
  Viscosity Model = String Glen
  Glen Exponent = Real 3.0
  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
```

Block preconditioning

- In parallel runs a central challenge is to have good **parallel preconditioners**
- This problem is increasingly difficult for PDEs with vector fields
  - Navier-Stokes, elasticity, acoustics,…
  - Strongly coupled multiphysics problems
- Preconditioner need not to be just a matrix, it can be a procedure!
- **Idea**: Use as 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
  - Multilevel methods

Preconditioner (from right):

Instead of solving

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

Identify a preconditioner P which makes solution of

$$\mathbf{K}\mathbf{P}^{-1}\mathbf{z} = \mathbf{b},$$

with z=Px easier than the original problem.

Block precontioning

- Given a block system

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

- Block Gauss-Seidel

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

Block Jacobi

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

- Preconditioner is the operator which produces the new search direction $s^{(k)}$
- Use GCR to minimize the residual over the space

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

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

GCR with general search directions to solve Ku = f

$$
\begin{aligned}
& k = 0 \\
& \mathbf{r}^{(k)} = \mathbf{f} - \mathbf{K}\mathbf{u}^{(k)} \\
& \texttt{while } (\|\mathbf{r}^{(k)}\| < TOL\|\mathbf{f}\| \text{ and } k < m) \\
& \qquad \text{Generate the search direction } \mathbf{s}^{(k+1)} \\
& \qquad \mathbf{v}^{(k+1)} = \mathbf{K}\mathbf{s}^{(k+1)} \\
& \qquad \text{do } j = 1, k \\
& \qquad\quad \mathbf{v}^{(k+1)} = \mathbf{v}^{(k+1)} - \langle \mathbf{v}^{(j)}, \mathbf{v}^{(k+1)} \rangle \mathbf{v}^{(j)} \\
& \qquad\quad \mathbf{s}^{(k+1)} = \mathbf{s}^{(k+1)} - \langle \mathbf{v}^{(j)}, \mathbf{v}^{(k+1)} \rangle \mathbf{s}^{(j)} \\
& \qquad \texttt{end do} \\
& \qquad \mathbf{v}^{(k+1)} = \mathbf{v}^{(k+1)} / \|\mathbf{v}^{(k+1)}\| \\
& \qquad \mathbf{s}^{(k+1)} = \mathbf{s}^{(k+1)} / \|\mathbf{v}^{(k+1)}\| \\
& \qquad \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \langle \mathbf{v}^{(k+1)}, \mathbf{r}^{(k)} \rangle \mathbf{s}^{(k+1)} \\
& \qquad \mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \langle \mathbf{v}^{(k+1)}, \mathbf{r}^{(k)} \rangle \mathbf{v}^{(k+1)} \\
& \qquad k = k + 1 \\
& \texttt{end while}
\end{aligned}
$$

Block preconditioner for the Stokes problem

- Each nonlinear step requires solving the Stokes problem

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

- Note that here C is result of stabilization, with suitable choice of basis vectors it can also be zero. The preconditioner is of the form

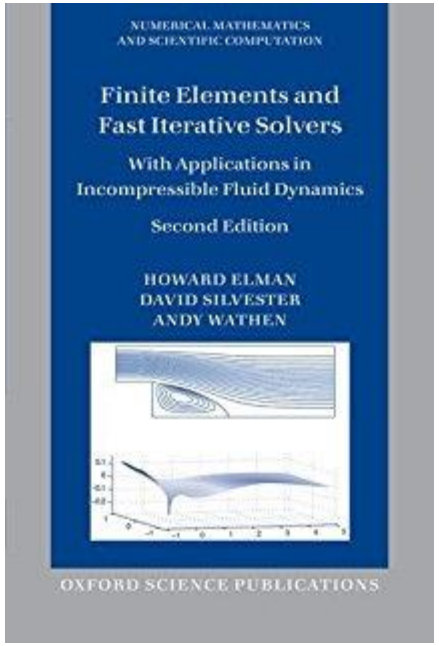
$$\mathrm{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$$

- An optimal choice of Q corresponds to the Schur complement. Usual choice is

$$\mathbf{Q} = \varepsilon^{-1}\mathbf{M},$$

where M is the mass matrix and ε is the viscosity from previous iteration.

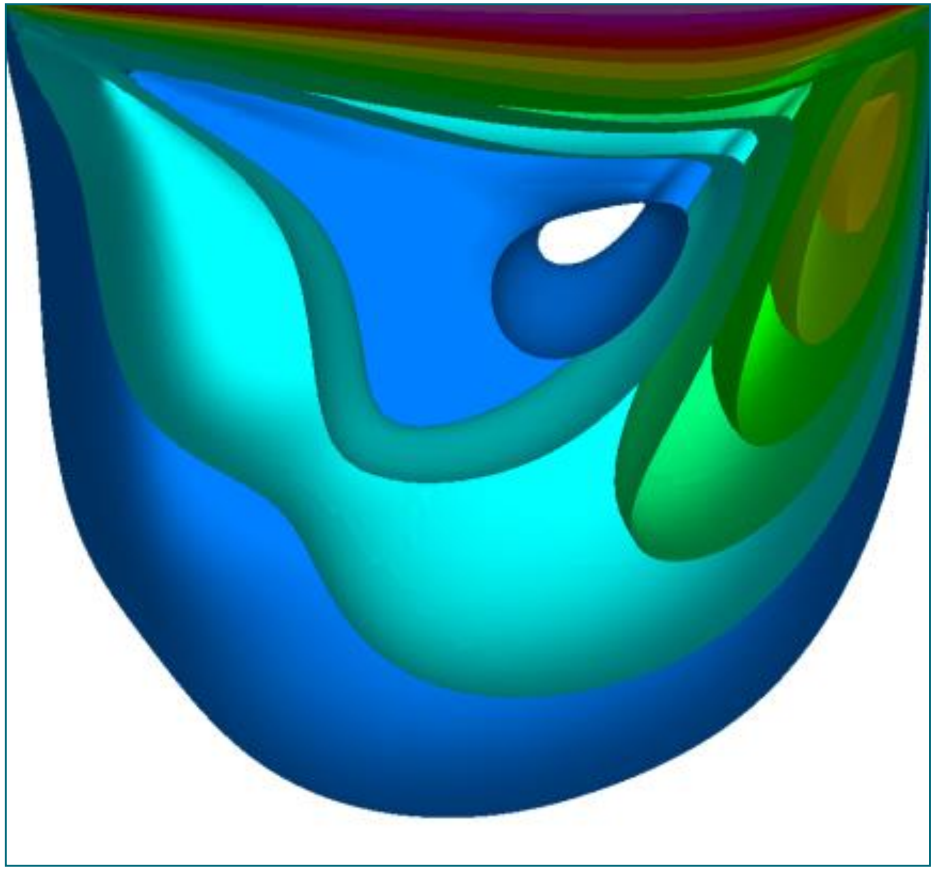
- We may split A to 3x3 submatrix also, or not



H. Elman, D. Silvester, A. Wathen,
Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics,
OUP Oxford, 2005.

Block preconditioner: Weak scaling of 3D driven-cavity

Elems	Dofs	#procs	Time (s)
34^3	171,500	16	44.2
43^3	340,736	32	60.3
54^3	665,500	64	66.7
68^3	1,314,036	128	73.6
86^3	2,634,012	256	83.5
108^3	5,180,116	512	102.0
132^3	9,410,548	1024	106.8



Velocity solves with Hypre: CG + BoomerAMG preconditioner for the 3D driven-cavity case (Re=100) on Cray XC (Sisu).
Simulation Mika Malinen, CSC, 2013.

O(~1.14)

Using block solver strategy with new Stokes module

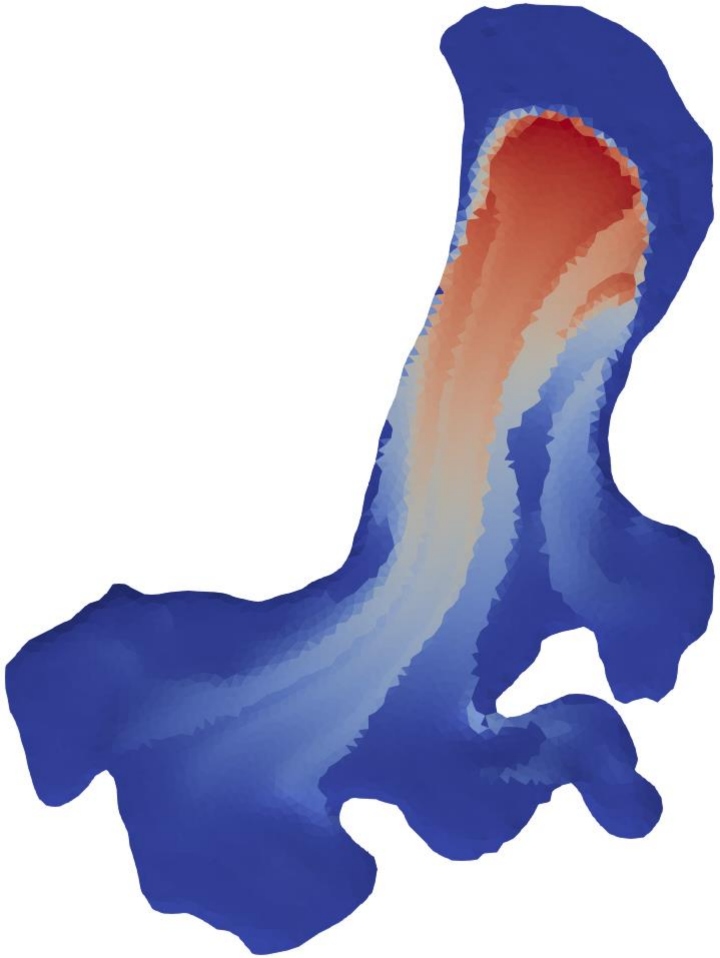
- We choose overall block splitting and strategy
- **GCR** is recommended for **outer** level
  - Does not require preconditioner to be exact!
- Different strategies may basically be used for different blocks for each **inner** system
  - Blocks 1,2,3 here associated with velocity components 1,2,3
  - Block 4 associated with pressure (preconditioned with scaled mass matrix is suggested by Elman)
- The strategy recides nowadays almost completely in the library functionality of Elmer
  - Makes dedicated block-preconditioned ParStokes obsolite
  - This strategy is not available in FlowSolver
- Note: the benefits of optimal scaling become obvious when the size of the problem grows

```
! Setting to choose block solver strategy
  Linear System Solver = "Block"
  Block Gauss-Seidel = Logical True
  Block Scaling = Logical False
  Block Preconditioner = Logical True
! Block Structure(4) = Integer 1 1 1 2
  Block Order(4) = Integer 1 2 3 4

! 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 =  25
  Outer: Linear System Max Iterations =  200
  Outer: Linear System Abort Not Converged = False
  Outer: Linear System Convergence Tolerance = 1e-8

  $blocktol = 0.001
  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
  …
```

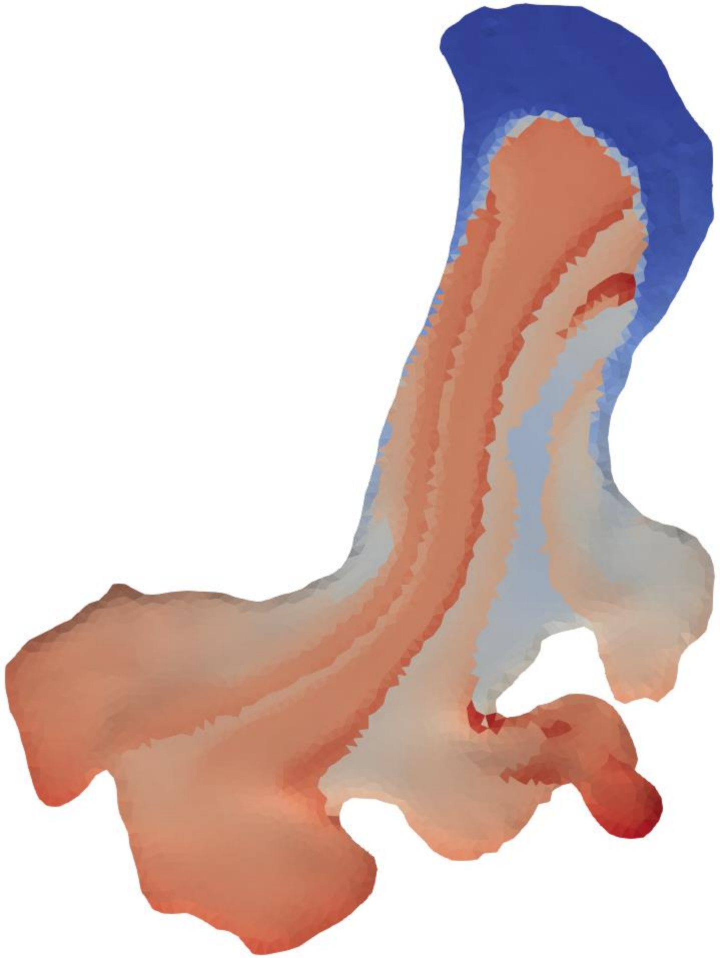
Advecting with the ice flow: ParticleAdvector

- 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
```



Distance travelled



Initial height of ice

Running the case

- In serial:

```
ElmerSolver mlb.sif
```

- In parallel, here with 4 processes:

```
ElmerGrid 2 2 outline62_lc50 -partdual -metisrec 4
mpirun -np 4 ElmerSolver_mpi mlb.sif
```

- An my laptop the basic case takes

Things to test by yourself

- Running the initial case (cl50)
- Runnig the smaller/larger cases (cl75, cl25)
- 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 precondtioner vs. direct solver
  - mlb_linsys.sif contains linear system recipes with "include linsys.sis"
- Trying effect of Courant number in particle advection
- ...
- You may turn off ParticleAdvector off if not needed as it uses a lot of time
  - Exec Solver = never