Elmer

Pre-processing utilities within ElmerSolver

ElmerTeam
CSC – IT Center for Science

Alternatives for increasing mesh resolution

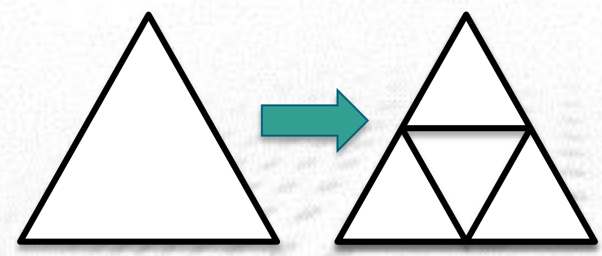
- Use of higher order nodal elements
  - Elmer supports 2nd to 4th order nodal elements
  - Unfortunately not all preprocessing steps are equally well supported for higher order elements
    - E.g. Netgen output supported only for linear elements
- Use of hierarhical p-element basis functions
  - Support up to 10th degree polynomials
  - In practice Element = p:2, or p:3
  - Not supported in all Solvers
- Mesh multiplication
  - Subdivision of elements by splitting

Note on bottle-necks in pre-processing

- After the solution pre-processing is typically the 2nd most time- and memory intensive task
- Mesh partitioning is typically less laborious than mesh generation
  - In Elmer we haven't utilized parallel graph partitioning libraries (e.g. ParMetis)
- Serial mesh generation limited to around ~10 M elements
- Finalizing the mesh in parallel level within ElmerSolver may be used to eliminate this bottle-neck

Finalizing the mesh in parallel level

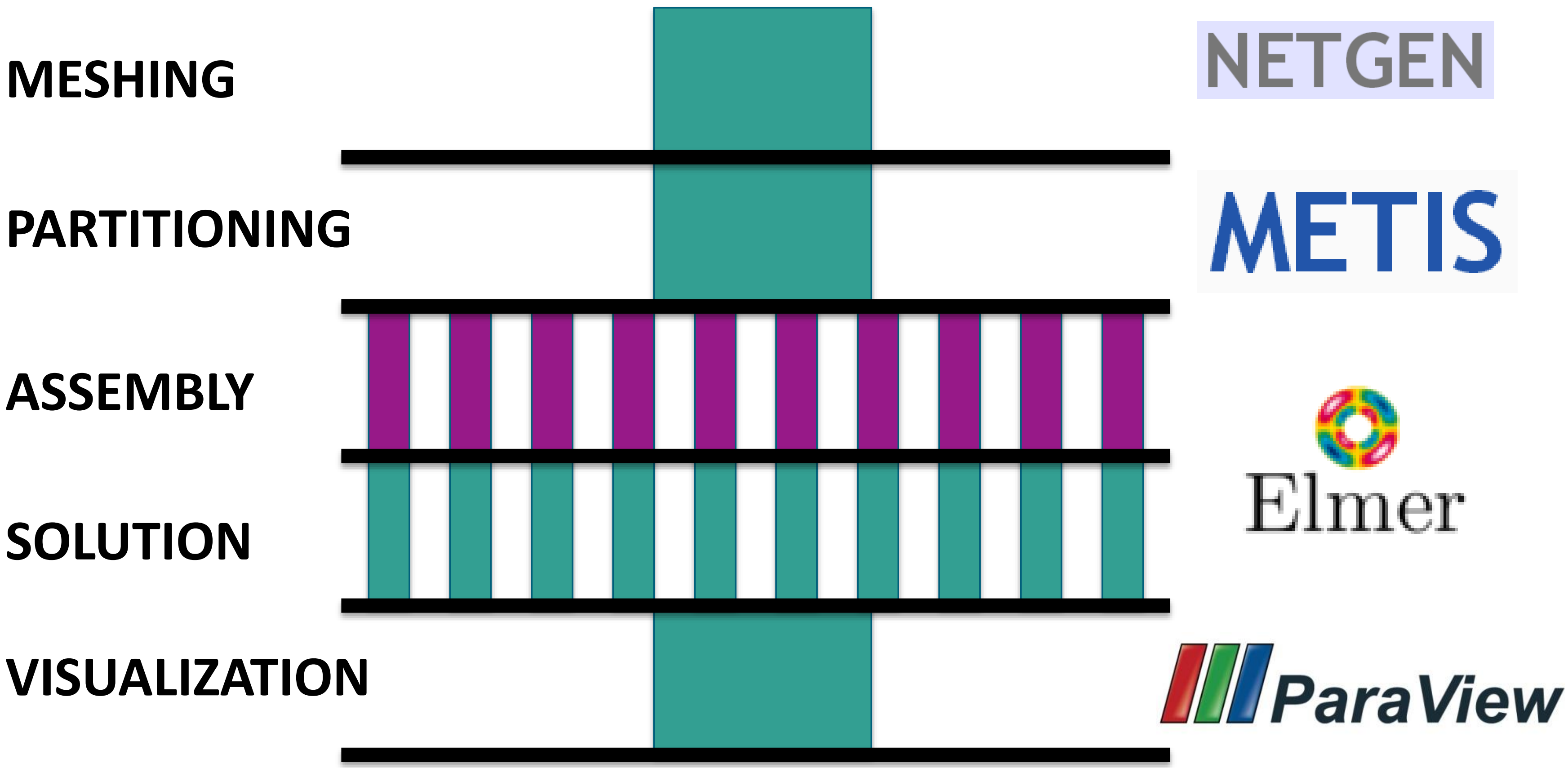
- First make a coarse mesh and partition it
- Bisection of existing elements in each direction
  - *2^DIM^n* -fold problem-size
  - Known as ”**Mesh Multiplication**”
  - Simple inheritance of mesh grading



- Increase of element order (p-elements)
  - p-hierarchy enables the use of p-multigrid
- Extrusion of 2D layer into 3D for special cases
  - Example: Greenland Ice-sheet

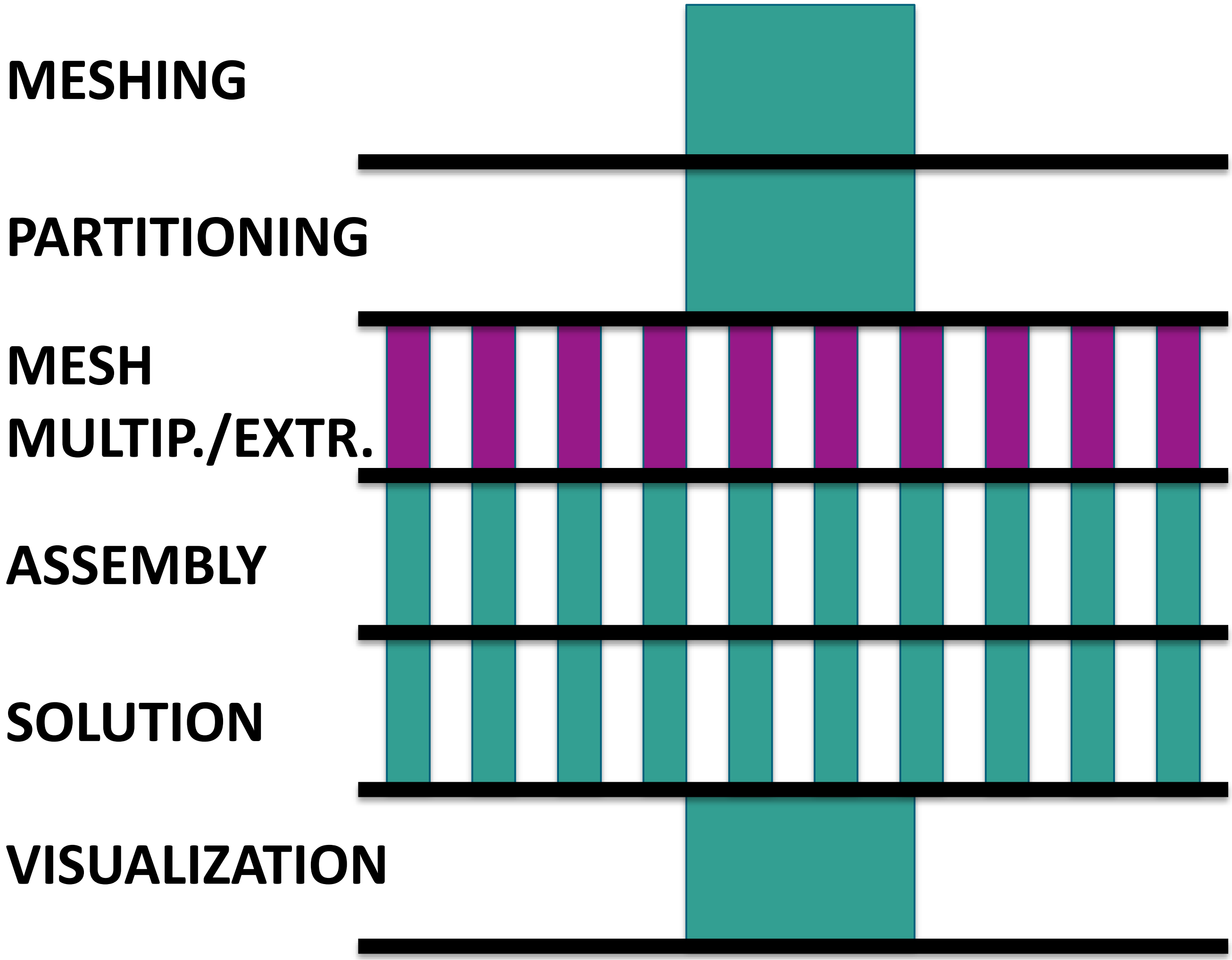
Standard parallel workflow

- Both assembly and solution is done in parallel using MPI
- Assembly is trivially parallel
- This is the basic parallel workflow used for Elmer



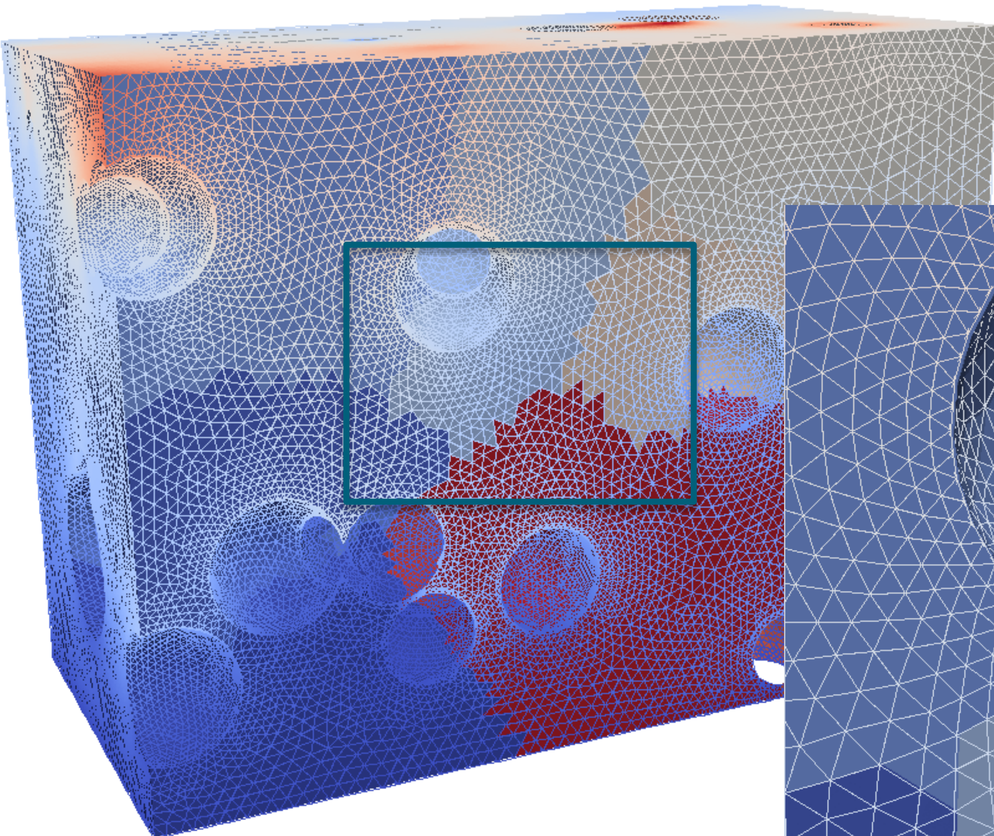
Parallel workflow

- Large meshes may be finilized at the parallel level

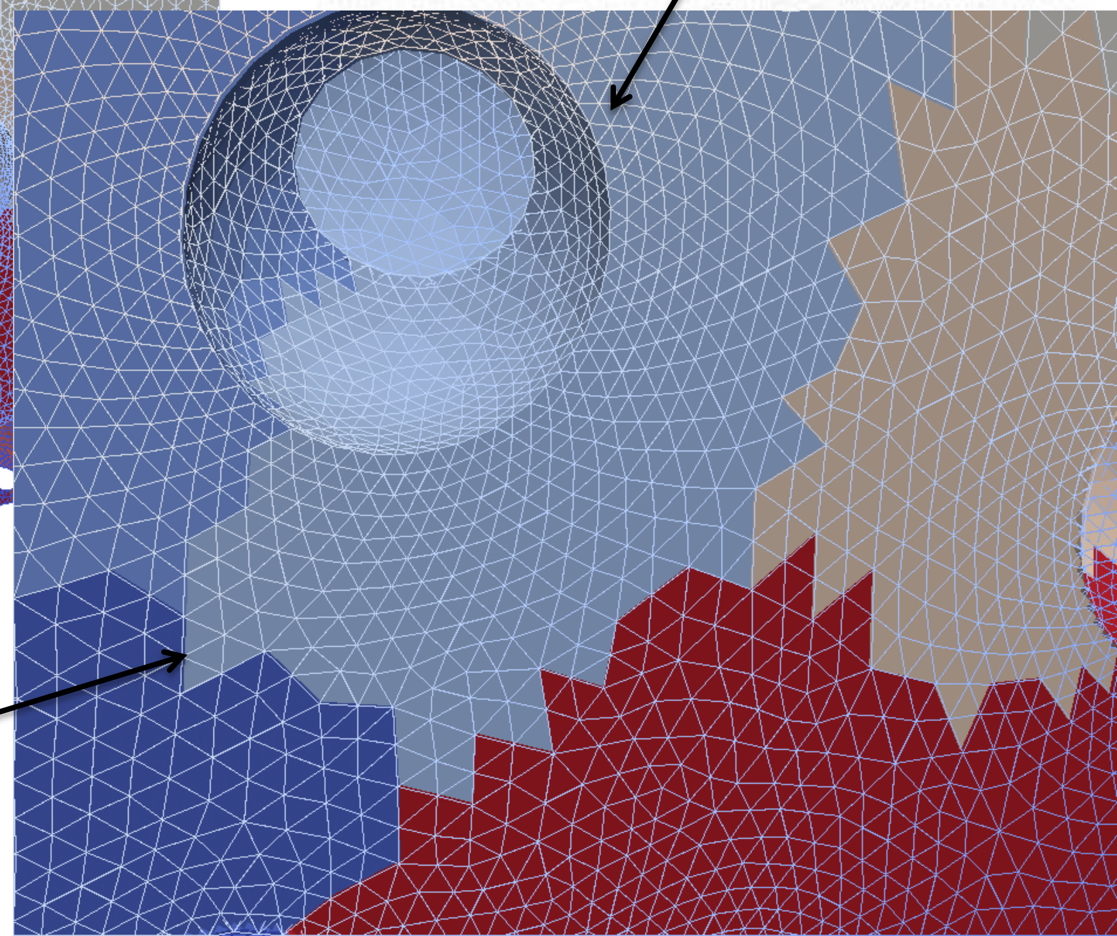


Mesh multiplication, example

`Mesh Levels = 2`



Mesh grading nicely preserved



Splitting effects visible
in partition interfaces

Mesh Multiplication, example

- Implemented in Elmer as internal strategy ~2005
- Mesh multiplication was applied to two meshes
  - Mesh A: structured, 62500 hexahedrons
  - Mesh B: unstructured, 65689 tetrahedrons
- The CPU time used is negligible

Mesh	#splits	#elems	#procs	T_center (s)	T_graded (s)
A	2	4 M	12	0.469	0.769
	2	4 M	128	0.039	0.069
	3	32 M	128	0.310	0.549
B	2	4.20 M	12	0.369	
	2	4.20 M	128	0.019	
	3	33.63 M	128	0.201	

Limitations of mesh multiplication

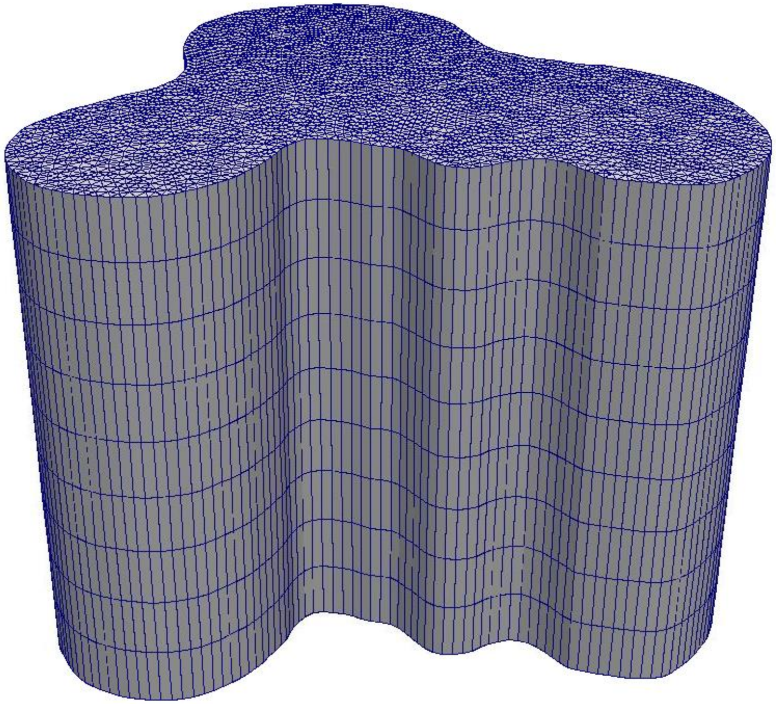
- Standard mesh multiplication does not increase geometric accuracy
  - Polygons retain their shape
  - Mesh multiplication could be made to honor boundary shapes but this is not currently done
- Optimal mesh grading difficult to achieve
  - The coarsest mesh level does not usually have sufficient information to implement fine level grading

Extrusion of partitioned meshes

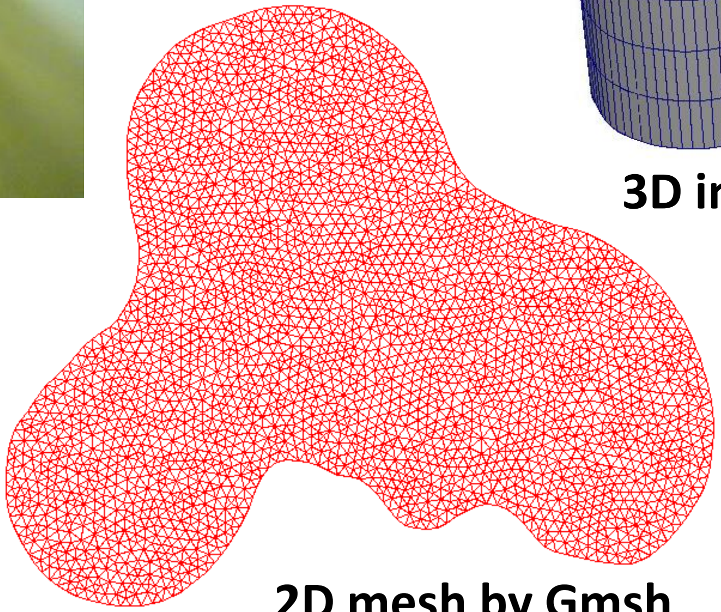
- Implemented as an internal strategy in ElmerSolver
- Star from an initial 2D mesh and then extrude into 3D
- Implemented also for partitioned meshes
  – Extruded lines belong to the same partition by construction!
- Deterministic, i.e. element and node numbering determined by the 2D mesh
  – Complexity: O(N)
- There are many problems of practical problems where the mesh extrusion of a initial 2D mesh provides a good solution
  – One such field is glasiology where glaciers are thin, yet the 2D approach is not always sufficient in accurary

Internal extrusion example: Aalto Vase

Design Alvar Aalto, 1936



3D internally extruded mesh



2D mesh by Gmsh

Deforming meshes

- Meshes may be internally deformed
- **`MeshUpdate`** solver uses linear elasticity to deform the mesh
- **`RigidMeshMapper`** uses rigid deformations and their smooth transitions to deform the mesh
- Deforming meshes have number of uses
  - Deforming structures in multiphysics simultion
    - E.g. fluid-structure interaction
  - Rotating & sliding structures
  - Geometry optimization
    - Mesh topology remains unchanged

Conclusions on internal meshing features

- There are number of ways to increase the resolution of solution within ElmerSolver that eliminate meshing bottle-necks
  - For complex cases these may still be unsatisfactory
- Internal mesh deformation may be used to solve complex problems without a need for remeshing
  - Large deformations may be problematic and topological changes impossible

Elmer

Post-processing utilities within ElmerSolver

ElmerTeam
CSC – IT Center for Science

Postprocessing utilities in ElmerSolver

- Apart from saving distributed data there is a larger number of capabilities within ElmerSolver to treat data within ElmerSolver
  - Data reduction
    - nD -> 1D, 0D
  - Data averaging and filtering over time (FilterTimeSeries)
  - Derived fields (gradient, curl, divecgence,...)
  - Creating fields of material properties
- This functionality is often achieved by use of atomic auxiliary solvers

Exporting 2D/3D data: ResultOutputSolve

- Apart from saving the results in .ep format it is possible to use other postprocessing tools
- ResultOutputSolve offers several formats
  - vtk: Visualization tookit legacy format
  - vtu: Visualization tookit XML format
  - Gid: GiD software from CIMNE: http://gid.cimne.upc.es
  - Gmsh: Gmsh software: http://www.geuz.org/gmsh
  - Dx: OpenDx software
- **Vtu** is the recommended format!
  - offers parallel data handling capabilities
  - Has binary and single precision formats for saving disk space
  - Suffix **.vtu** in Post File does this automatically

Exporting 2D/3D data: ResultOutputSolve

An example shows how to save data in unstructured XML VTK (.vtu) files to directory ”results” in single precision binary format.

```
Solver n
  Exec Solver = after timestep
  Equation = "result output"
  Procedure = "ResultOutputSolve" "ResultOutputSolver"
  Output File Name = "case"
  Output Format = String ”vtu”
  Binary Output = True
  Single Precision = True
End
```

Derived fields

- Many solvers have internal options for computing derived fields (fluxes, heating powers,...)
- Elmer offers several auxiliary solvers
  - SaveMaterials: makes a material parameter into field variable
  - Streamlines: computes the streamlines of 2D flow
  - FluxComputation: given potential, computes the flux $q = - c \nabla\phi$
  - VorticitySolver: computes the vorticity of flow, $w = \nabla\times\phi$
  - PotentialSolver: given flux, compute the potential $- c \nabla\phi = q$
  - Filtered Data: compute filtered data from time series (mean, fourier coefficients,...)
  - …
- Usually auxiliary data need to be computed only after the iterative solution is ready
  - Exec Solver = after timestep
  - Exec Solver = after all
  - Exec Solver = before saving

Derived nodal data

- By default Elmer operates on distributed fields but sometimes nodal values are of interest
  - Multiphysics coupling may also be performed alternatively using nodal values for computing and setting loads
- Elmer computes the nodal loads from *Ax-b* where *A*, and *b* are saved before boundary conditions are applied
  - **`Calculate Loads = True`**
- This is the most consistant way of obtaining boundary loads
- Note: the nodal data is really pointwise
  - expressed in units N, C, W etc.
    (rather than N/m^2, C/m^2, W/m^2 etc.)
  - For comparison with distributed data divided by the ~size of the surface elements

Derived lower dimensional data

- Derived boundary data
  - SaveLine: Computes fluxes on-the-fly
- Derived lumped (or 0D) data
  - SaveScalars: Computes a large number of different quantities on-the-fly
  - FluidicForce: compute the fluidic force acting on a surface
  - ElectricForce: compute the electrostatic froce using the Maxwell stress tensor
  - Many solvers compute lumped quantities internally for later use
    (Capacitance, Lumped spring,...)

Saving 1D data: SaveLine

- Lines of interest may be defined on-the-fly
- Data can either be saved in uniform 1D grid, or where element faces and lines intersect
- Flux computation using integration points on the boundary – not the most accurate
- By default saves all existing field variables

Saving 1D data: SaveLine…

```
Solver n
 Equation = "SaveLine"
 Procedure = File "SaveData" "SaveLine"
 Filename =  "g.dat"
 File Append = Logical True
 Polyline Coordinates(2,2) = Real 0.0 1.0 0.0 2.0
End

Boundary Condition m
  Save Line = Logical True
End
```

Saving 0D data: SaveScalars

Operators on bodies

- Statistical operators
  - Min, max, min abs, max abs, mean, variance, deviation
- Integral operators (quadratures on bodies)
  - volume, int mean, int variance
  - Diffusive energy, convective energy, potential energy

Operators on boundaries

- Statistical operators
  - Boundary min, boundary max, boundary min abs, max abs, mean, boundary variance, boundary deviation, boundary sum
  - Min, max, minabs, maxabs, mean
- Integral operators (quadratures on boundary)
  - area
  - Diffusive flux, convective flux

Other operators

- nonlinear change, steady state change, time, timestep size,…

Saving 0D data: SaveScalars...

```
Solver n
  Exec Solver = after timestep
  Equation = String SaveScalars
  Procedure = File "SaveData" "SaveScalars"
  Filename = File "f.dat"
  Variable 1 = String Temperature
  Operator 1 = String max
  Variable 2 = String Temperature
  Operator 2 = String min
  Variable 3 = String Temperature
  Operator 3 = String mean
End

Boundary Condition m
  Save Scalars = Logical True
End
```

Case: TwelveSolvers

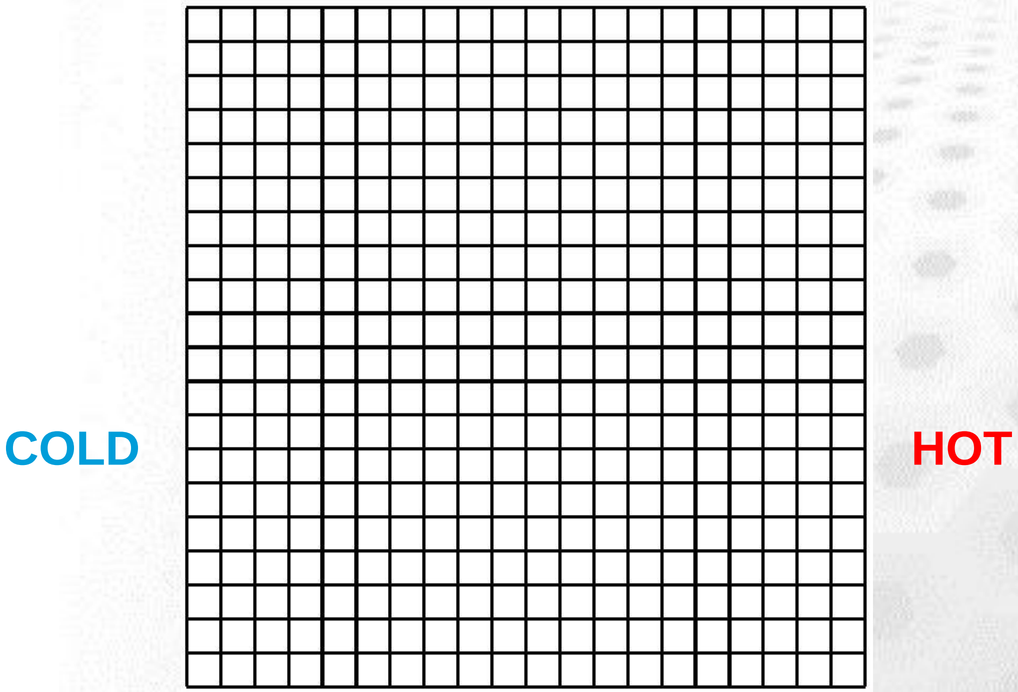
Natural convection with ten auxialiary solvers

Case: Motivation

- The purpose of the example is to show the flexibility of the modular structure
- The users should not be afraid to add new atomistic solvers to perform specific tasks
- A case of 12 solvers is rather rare, yet not totally unrealitistic

Case: preliminaries

- Square with hot wall on right and cold wall on left
- Filled with viscous fluid
- Bouyancy modeled with Boussinesq approximation
- Temperature difference initiates a convection roll



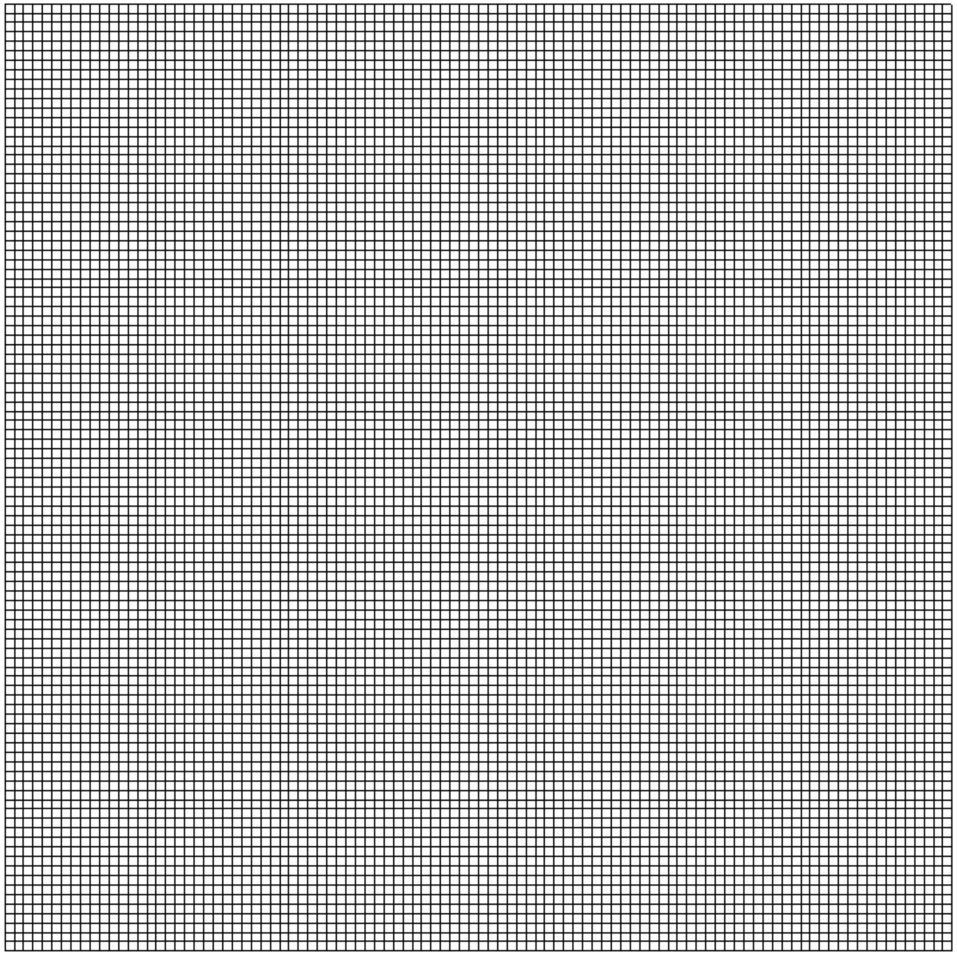
Case: 12 solvers

1. **HeatSolver**
2. **FlowSolver**

---

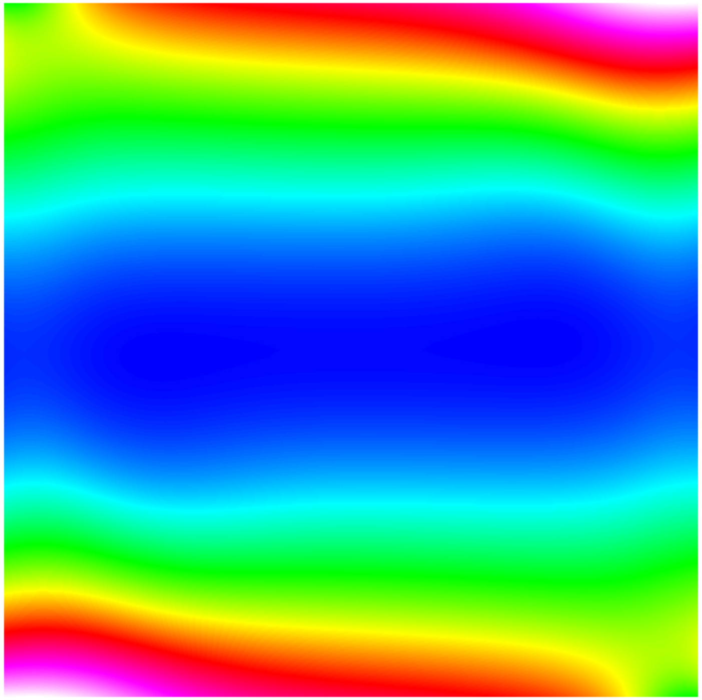
3. **FluxSolver**: solve the heat flux
4. **StreamSolver**: solve the stream function
5. **VorticitySolver**: solve the vorticity field (curl of vector field)
6. **DivergenceSolver**: solve the divergence
7. **ShearrateSolver**: calculate the shearrate
8. **IsosurfaceSolver**: generate an isosurface at given value
9. **ResultOutputSolver**: write data
10. **SaveGridData**: save data on uniform grid
11. **SaveLine**: save data on given lines
12. **SaveScalars**: save various reductions

Case: Computational mesh

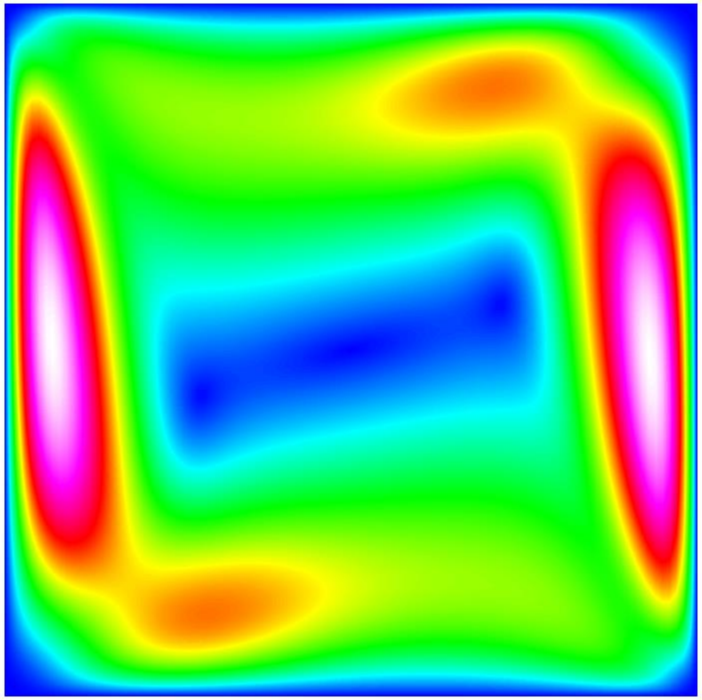


10000 bilinear elements

Case: Navier-Stokes, primary fields

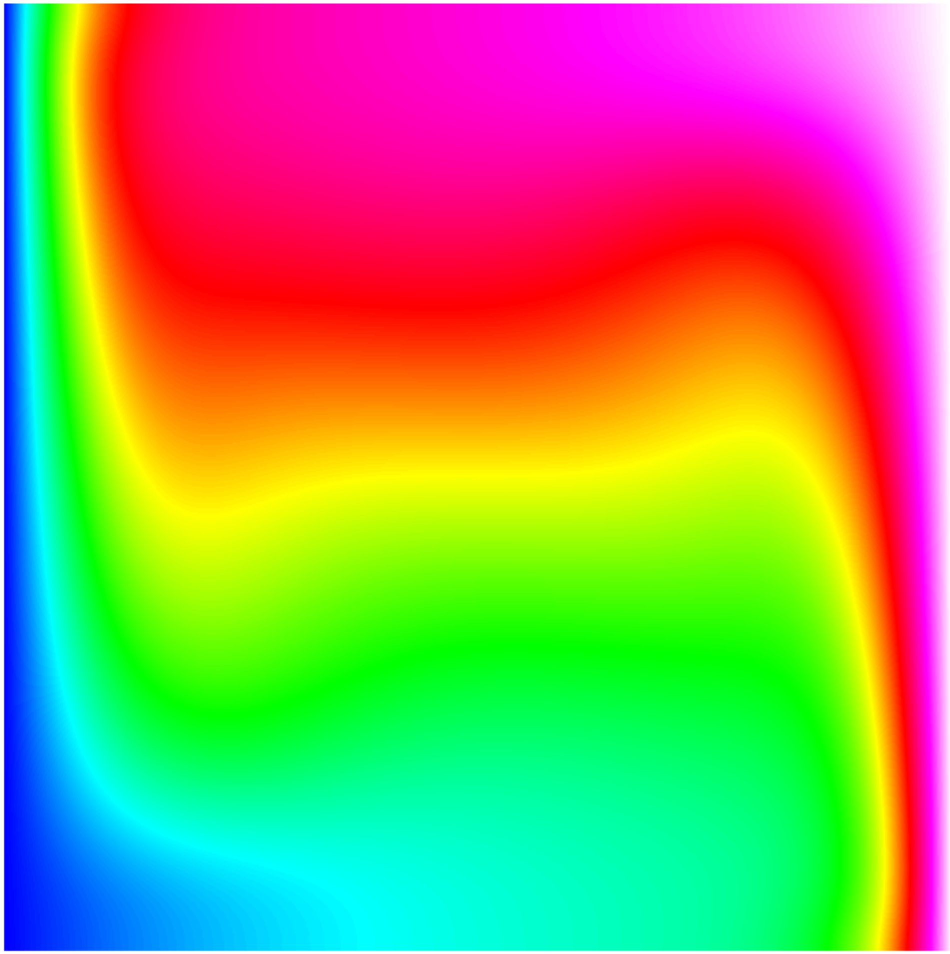


Pressure

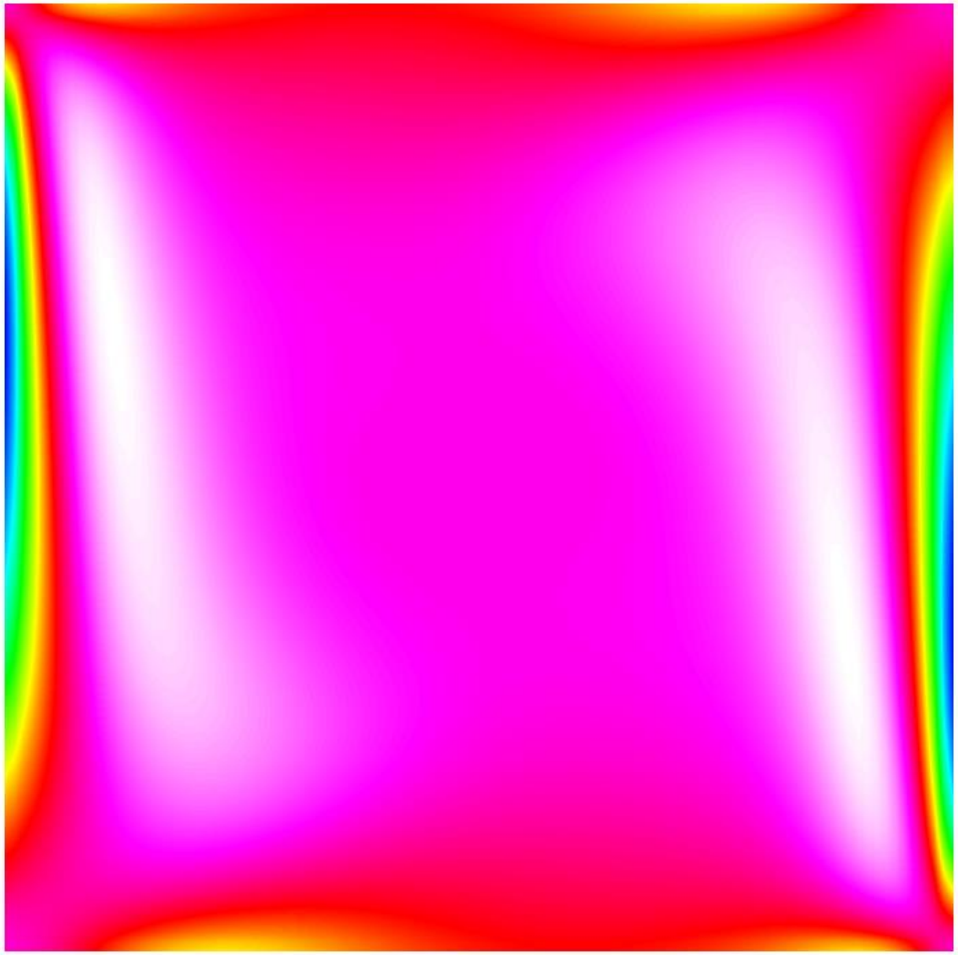


Velocity

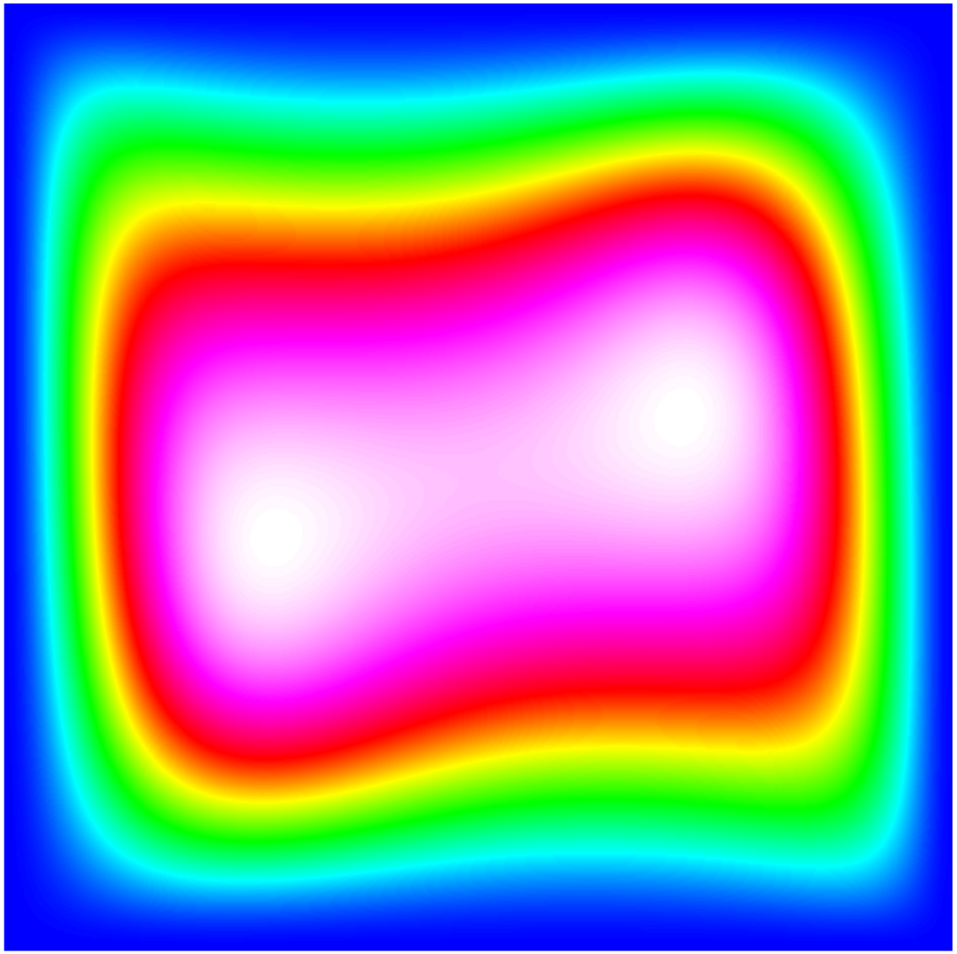
Case: Heat equation, primary field



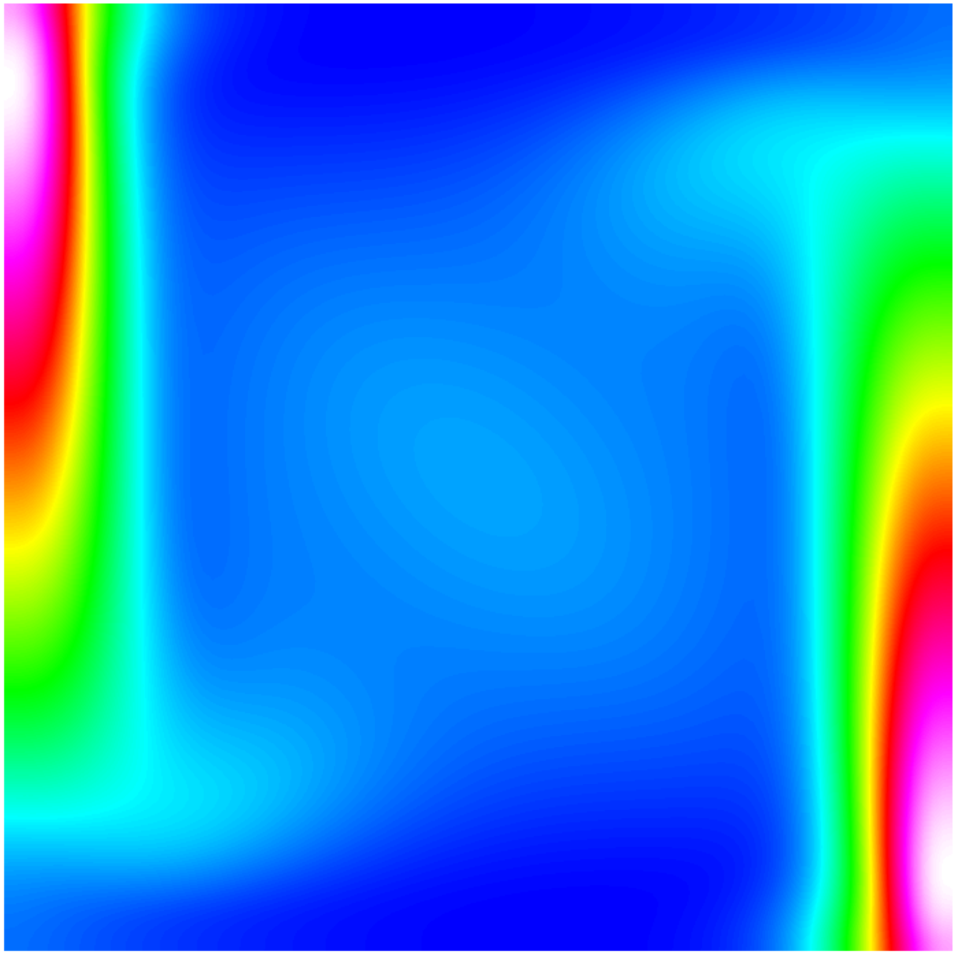
Case: Derived field, vorticity



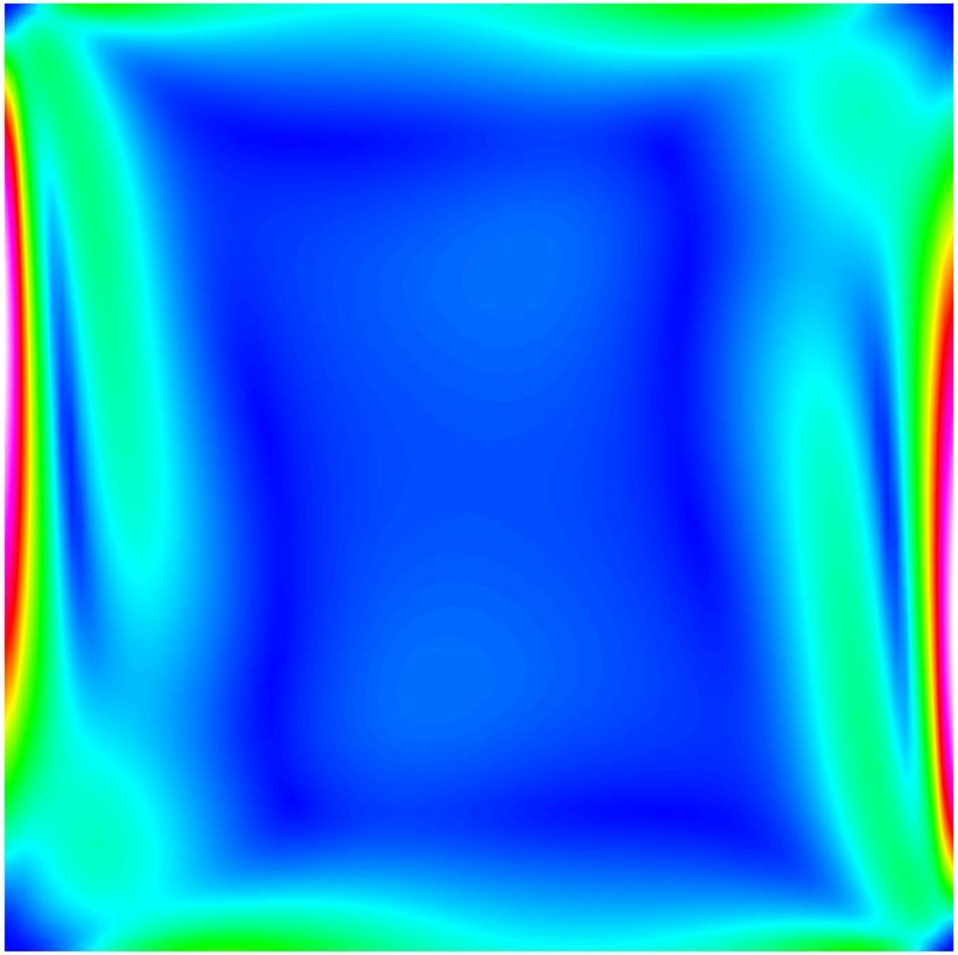
Case: Derived field, Streamlines



Case: Derived field, diffusive flux

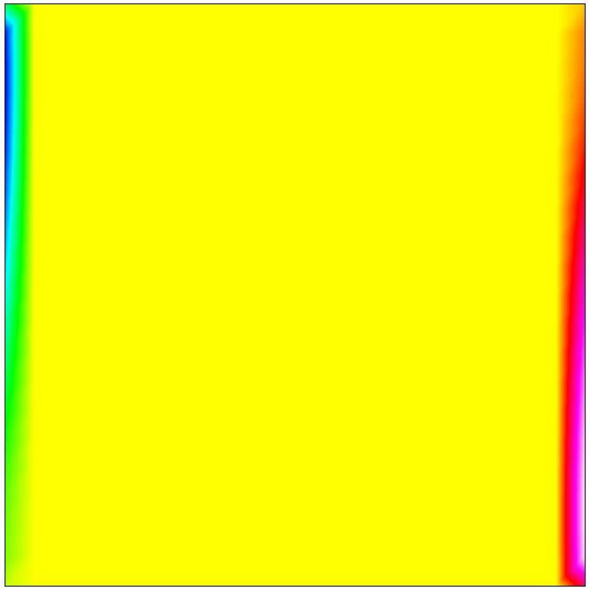


Case: Derived field, Shearrate



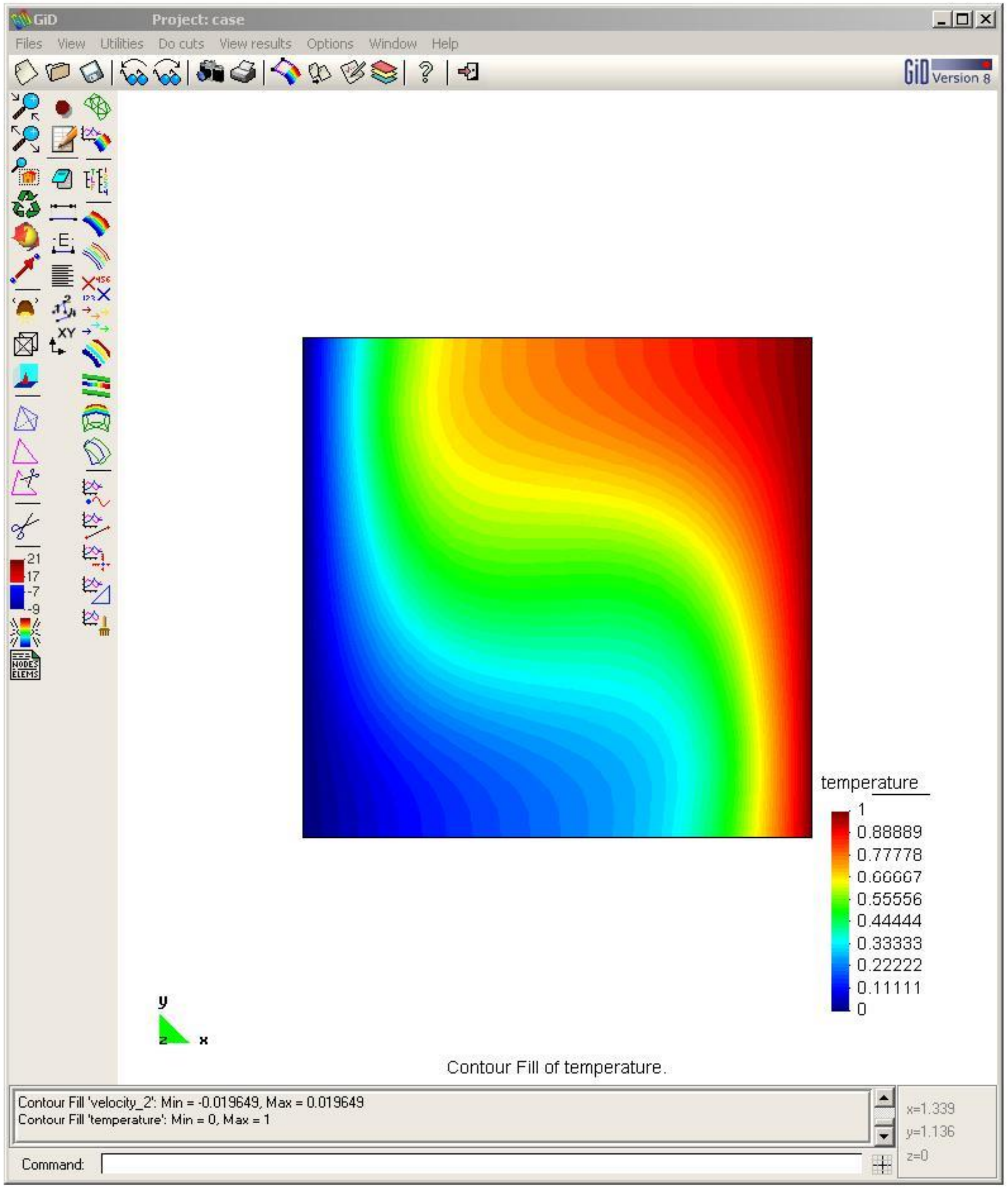
Example: nodal loads

- If equation is solved until convergence nodal loads should only occur at boundaries
- Element size h=1/20 ~weight for flux

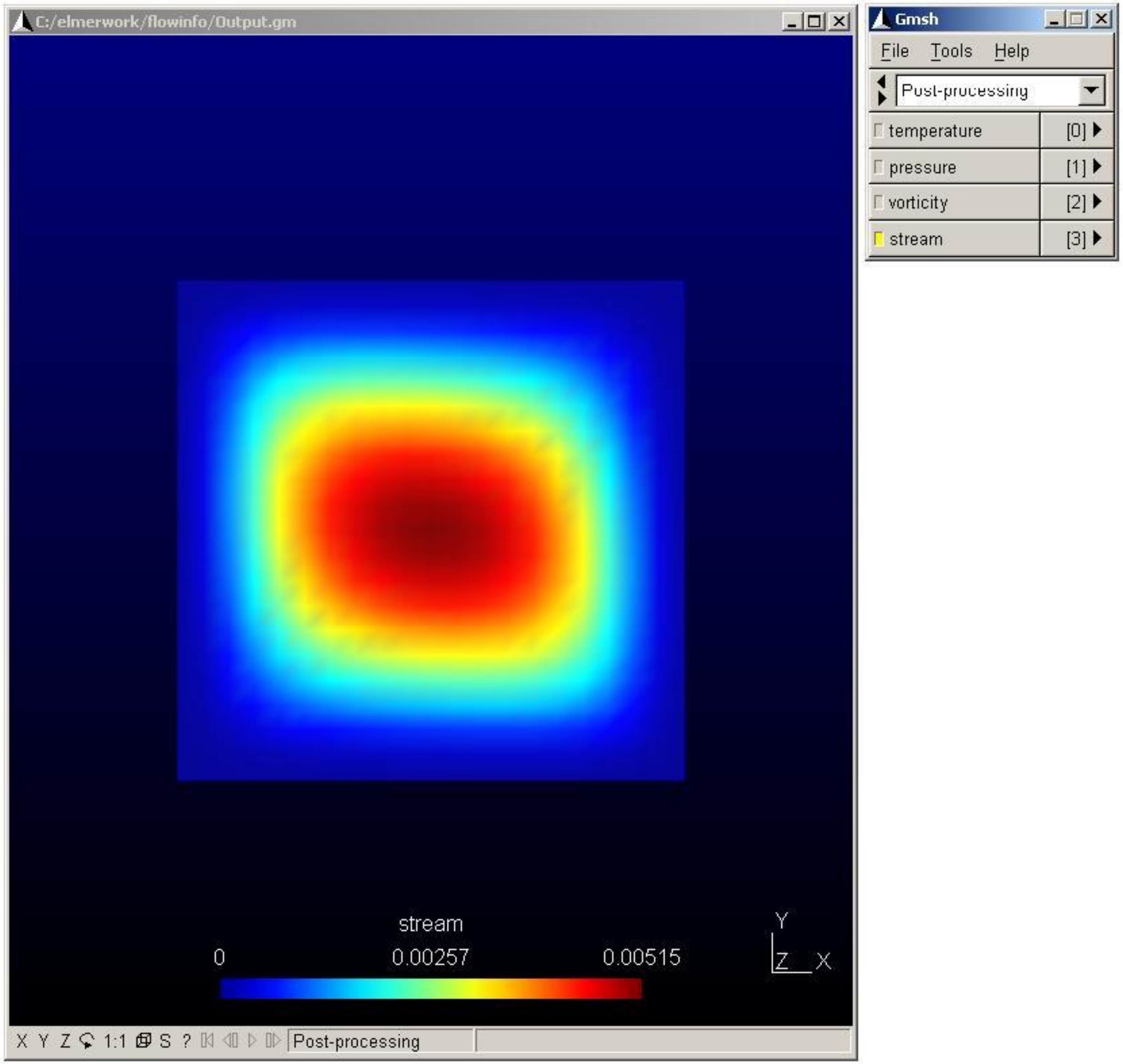


Nodal heat loads

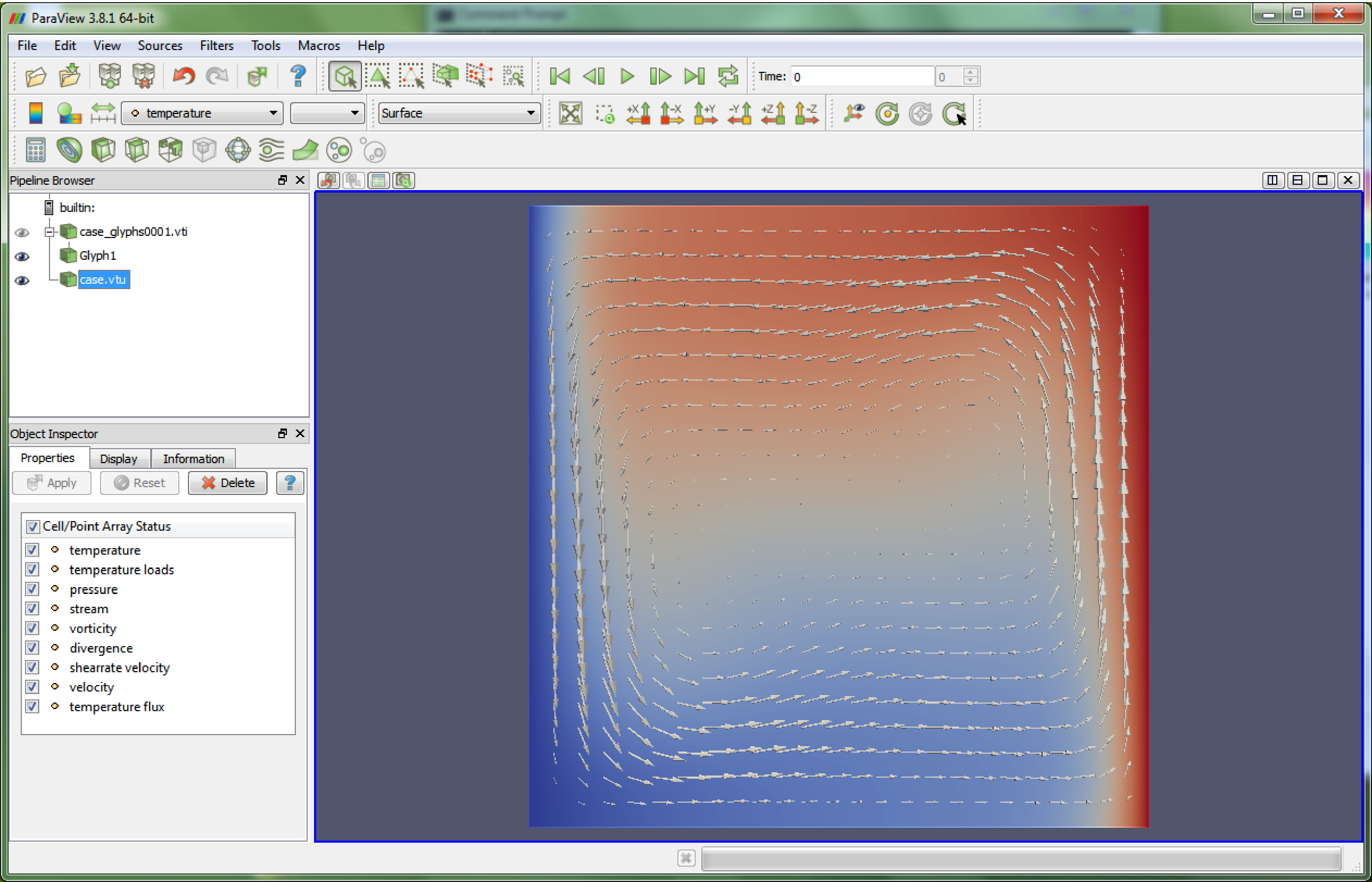
Example: view in GiD



Example: view in Gmsh

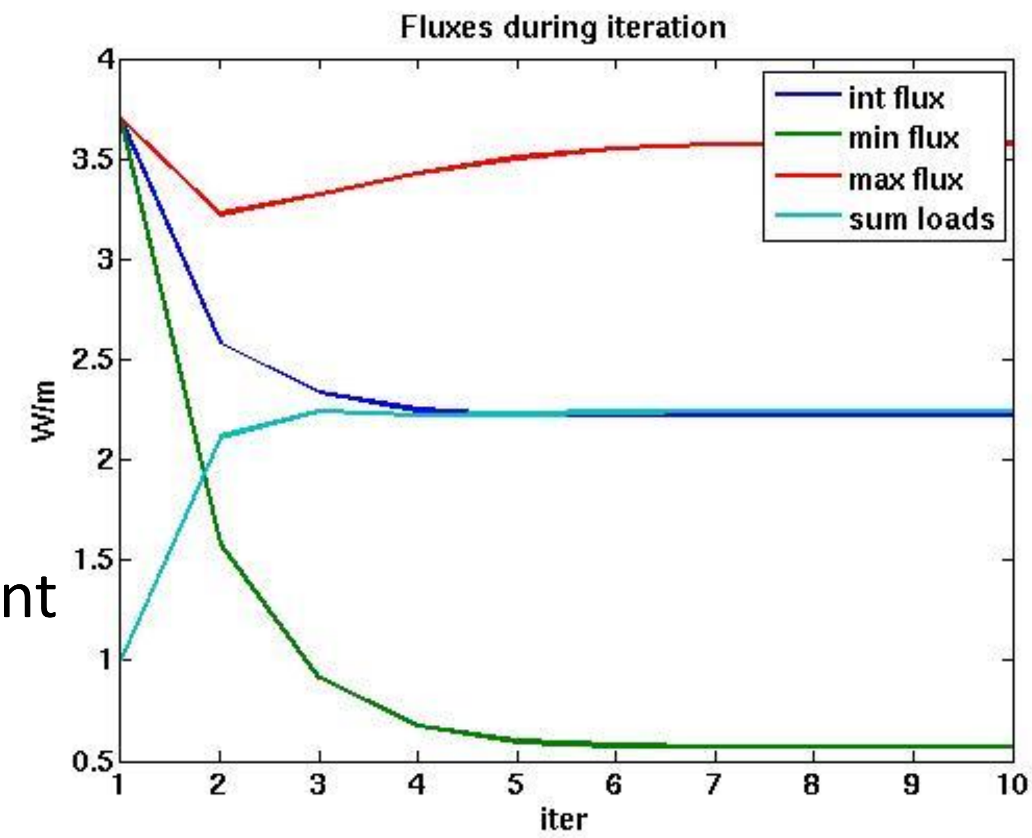


Case: View in Paraview



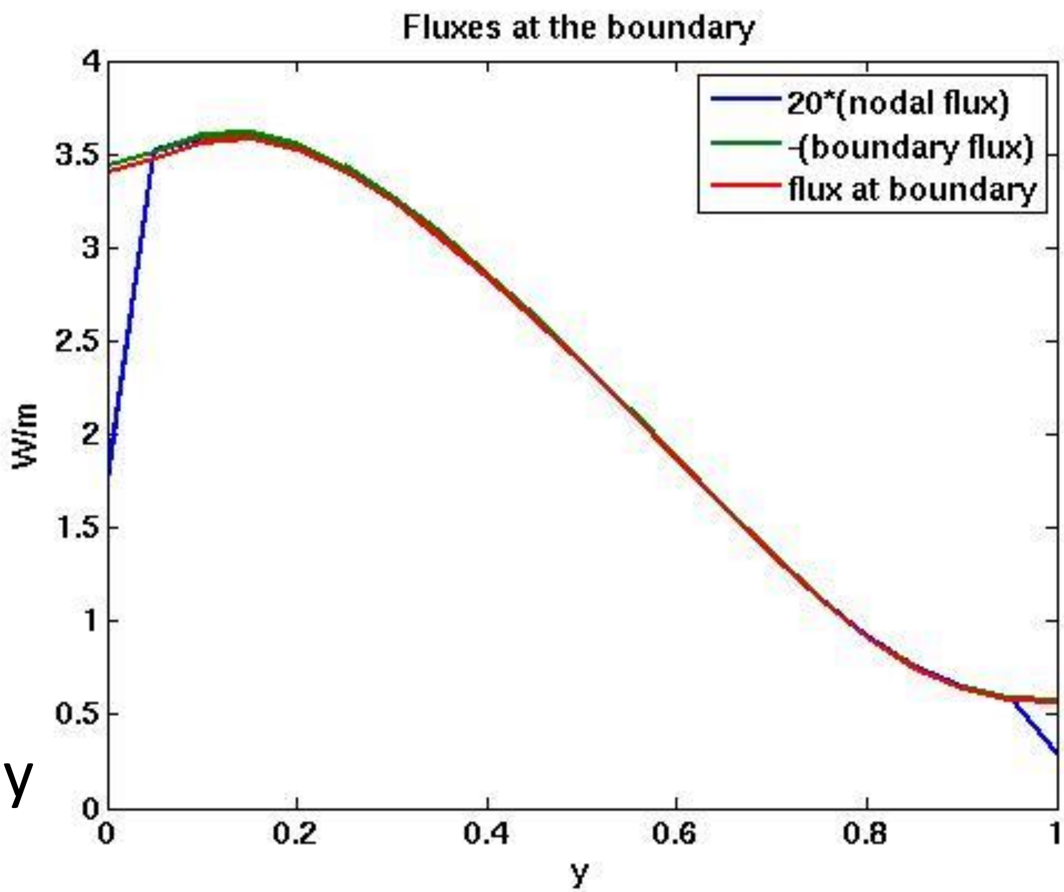
Example: total flux

- Saved by SaveScalars
- Two ways of computing the total flux give different approximations
- When convergence is reached the agreement is good



Example: boundary flux

- Saved by SaveLine
- Three ways of computing the boundary flux give different approximations
- At the corner the nodal flux should be normalized using only *h/2*



Exercise

- Study the command file with 12 solvers
- Copy-paste an appropriate solver from there to some existing case of your own
  – ResultOutputSolver for VTU output
  – StreamSolver, VorticitySolver, FluxSolver,…
- Note: Make sure that the numbering of Solvers is consistant
  – Solvers that involve finite element solution you need to activate by `Active Solvers`
- Run the modified case
- Visualize results in ElmerPost or Paraview

Conclusions

- It is good to think in advance what kind of data you need
  - 3D volume and 2D surface data
  - Derived fields
  - 1D line data
  - 0D lumped data
- Internal strategies may allow better accuracy than doing the analysis with external postprocessing software
  - Consistent use of basis functions to evaluate the data
- Often the same reduction operations may be done also at later stages but with significantly greater effort