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
  - Derived fields
  - Creating fields of material properties
- This functionality is often achieved by use of atomic auxiliary solvers

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 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
- 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

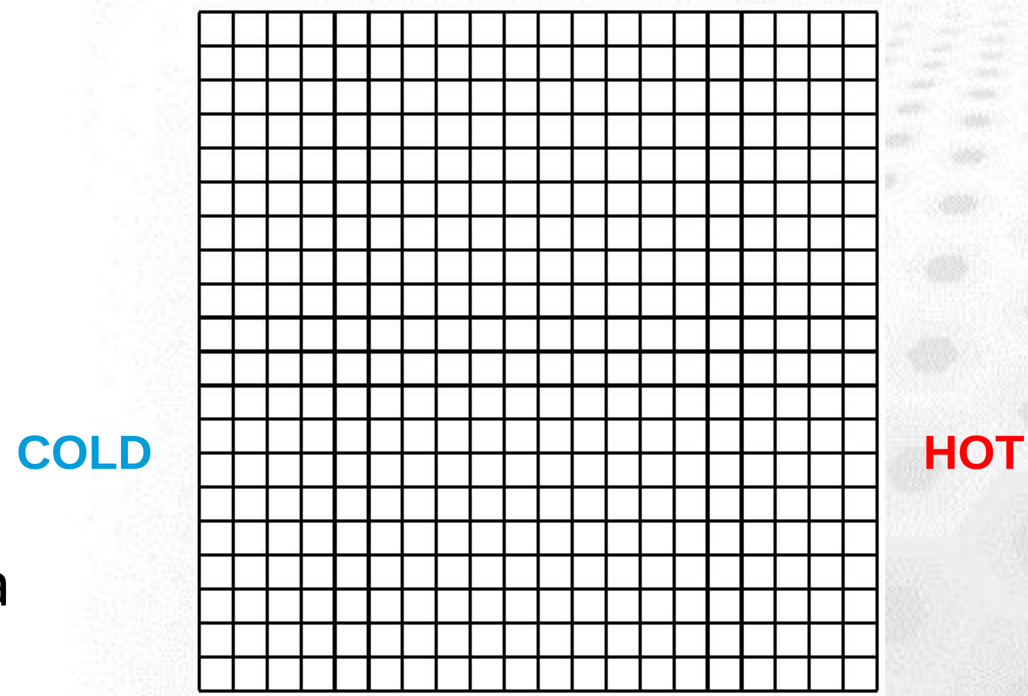
Natural convection with ten auxialiary solvers

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



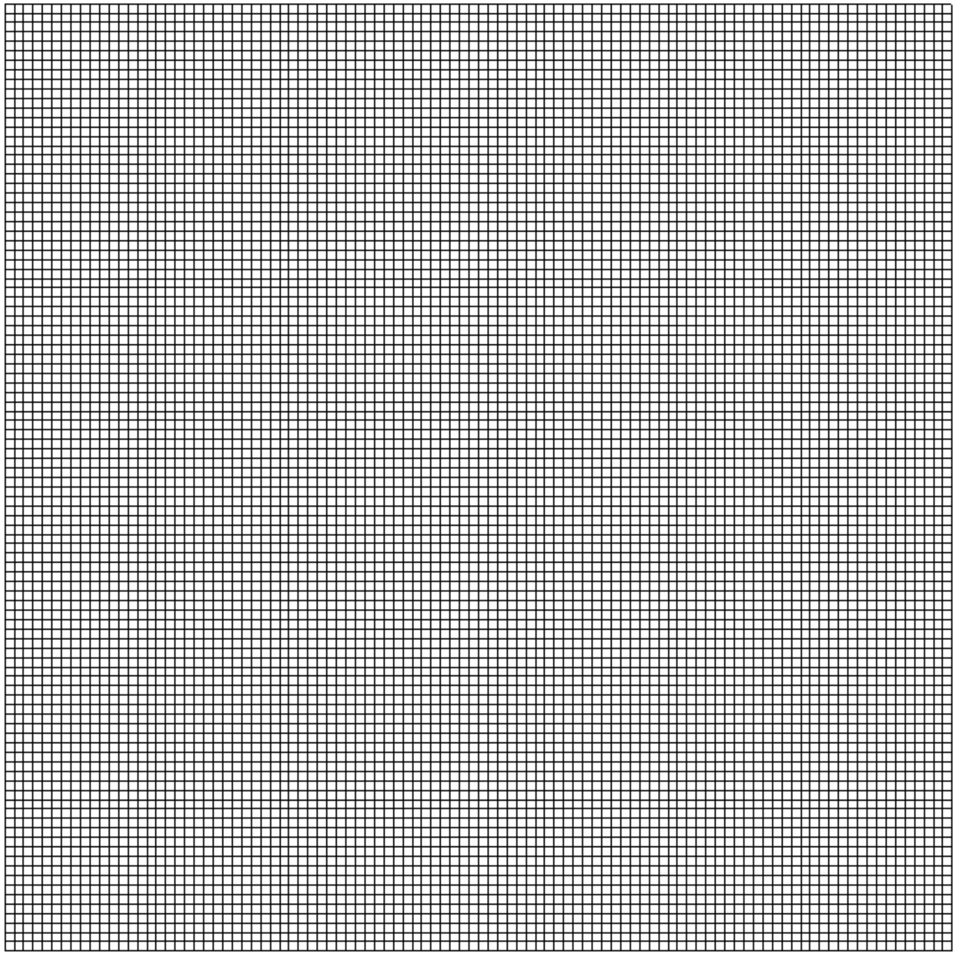
Case: 12 solvers

1. Heat Equation
2. Navier-Stokes

---

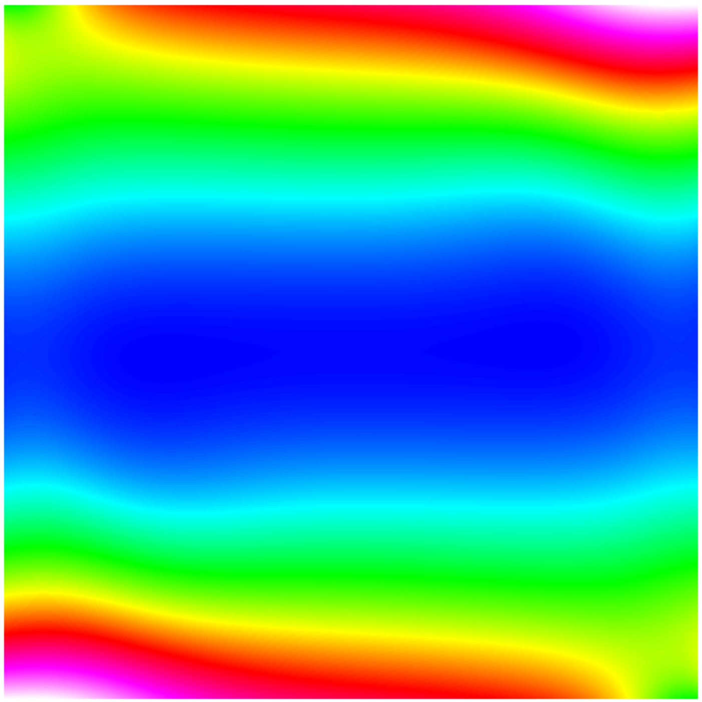
3. FluxSolver: solve the heat flux
4. StreamSolver
5. VorticitySolver
6. DivergenceSolver
7. ShearrateSolver
8. IsosurfaceSolver
9. ResultOutputSolver
10. SaveGridData
11. SaveLine
12. SaveScalars

Case: Computational mesh

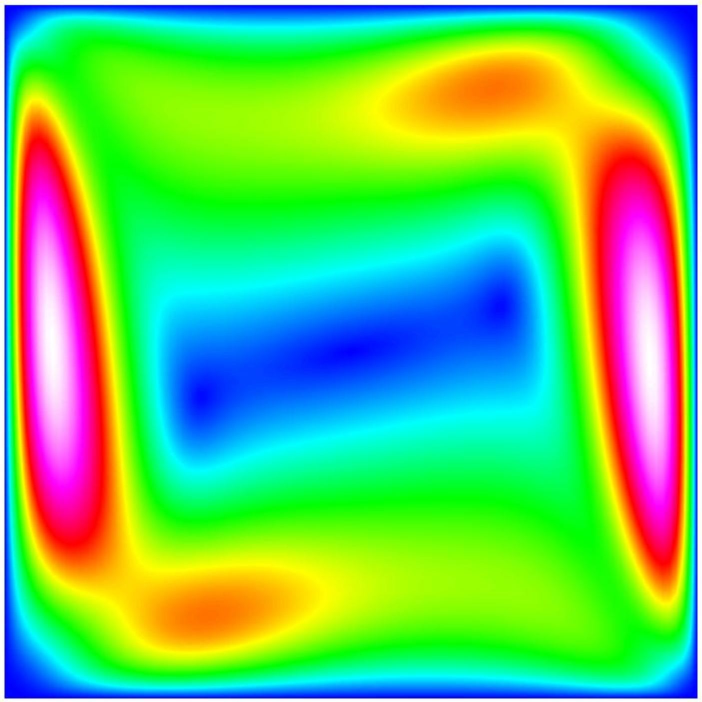


10000 bilinear elements

Case: Navier-Stokes, primary fields

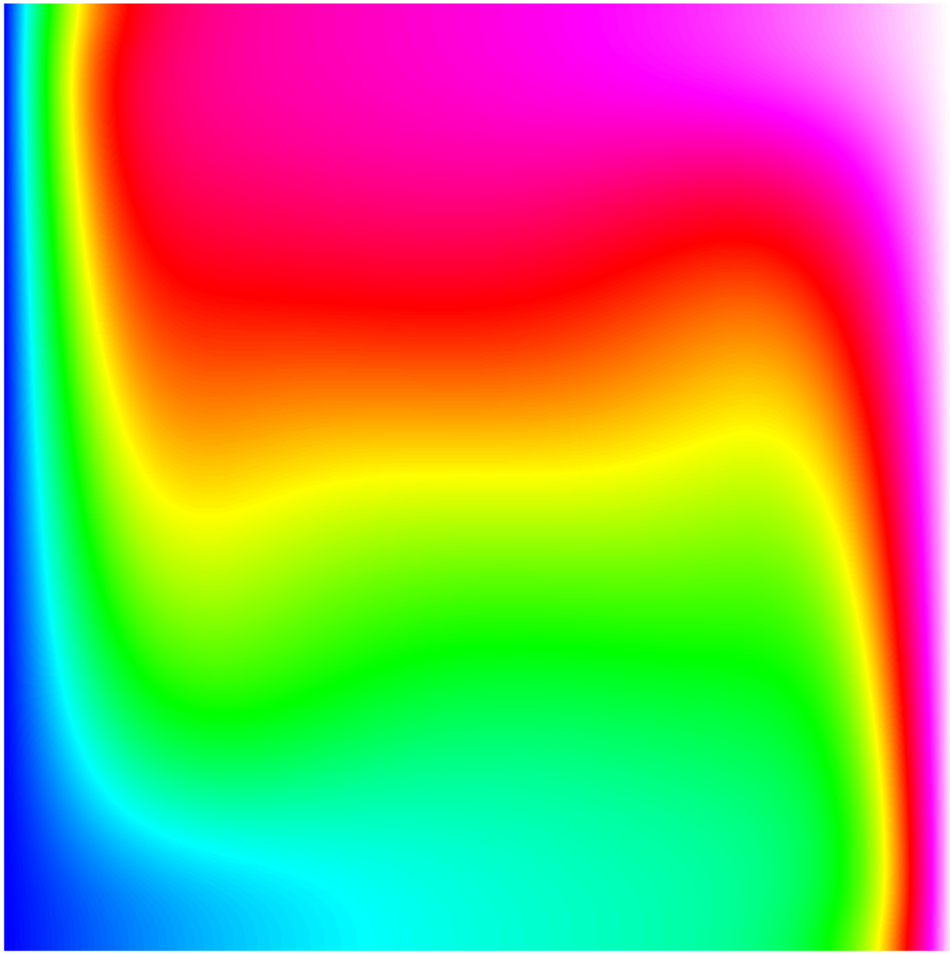


Pressure

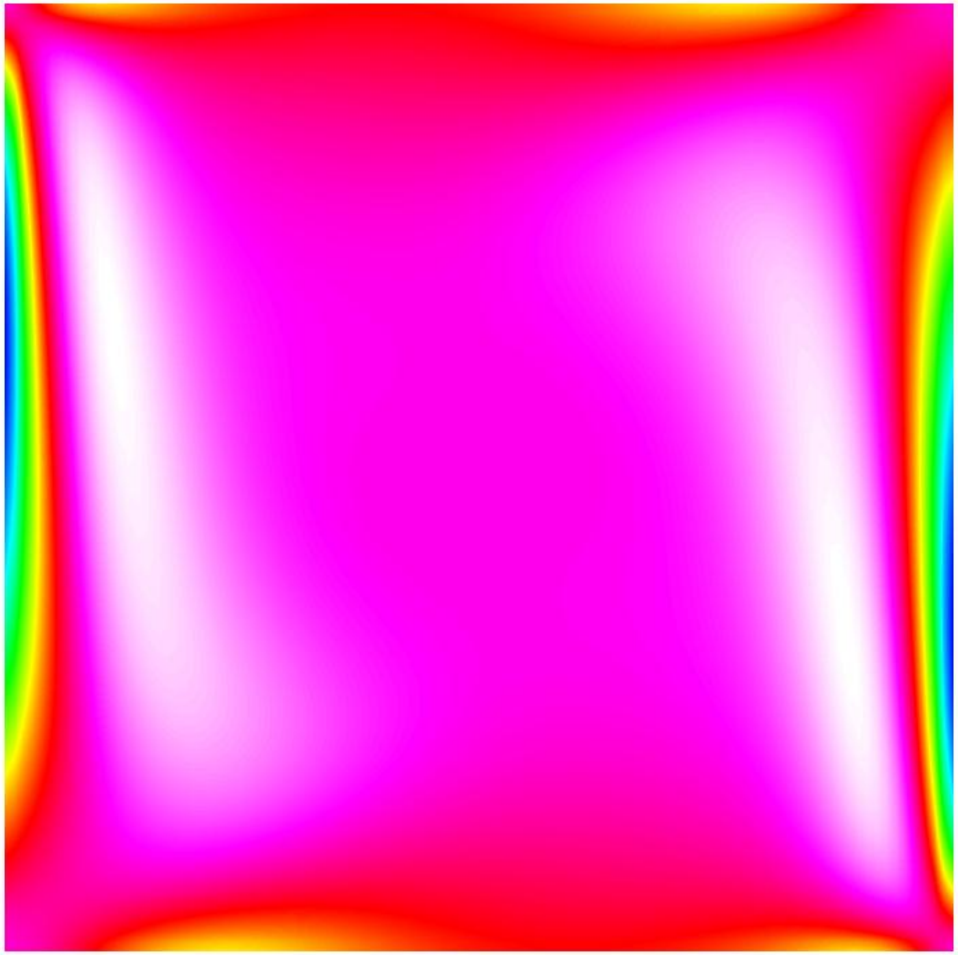


Velocity

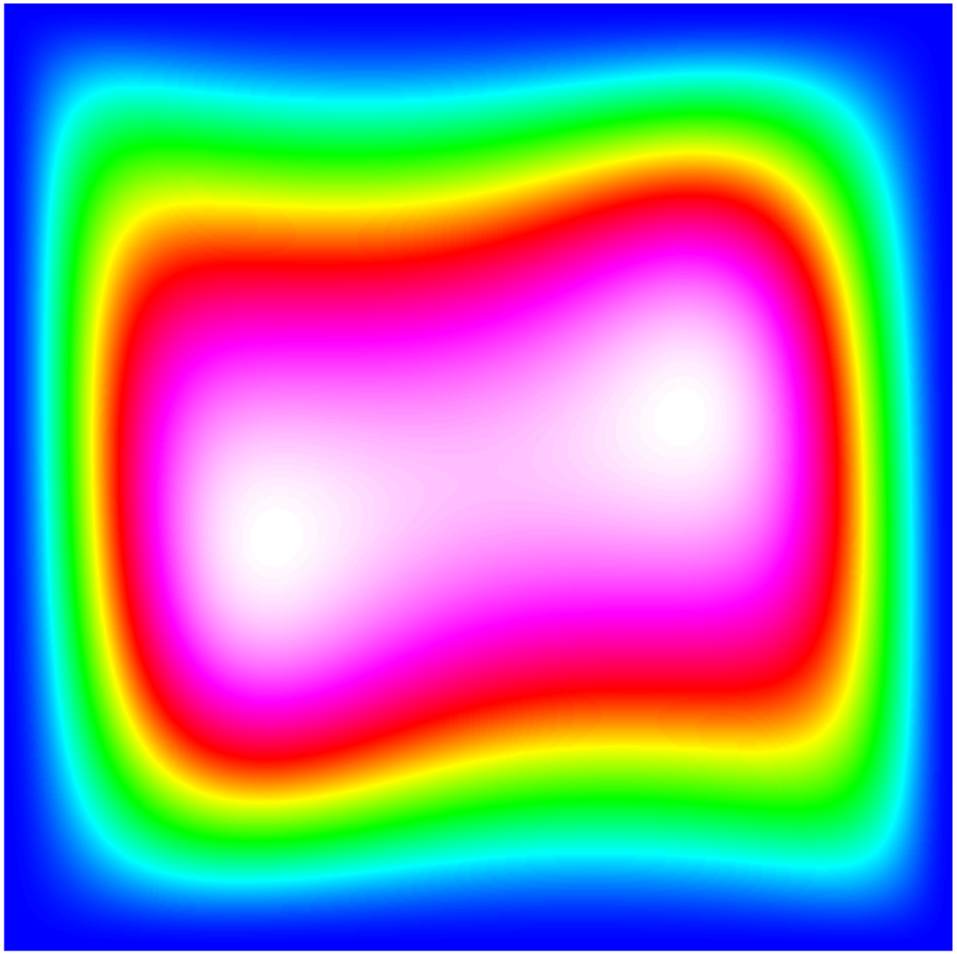
Case: Heat equation, primary field



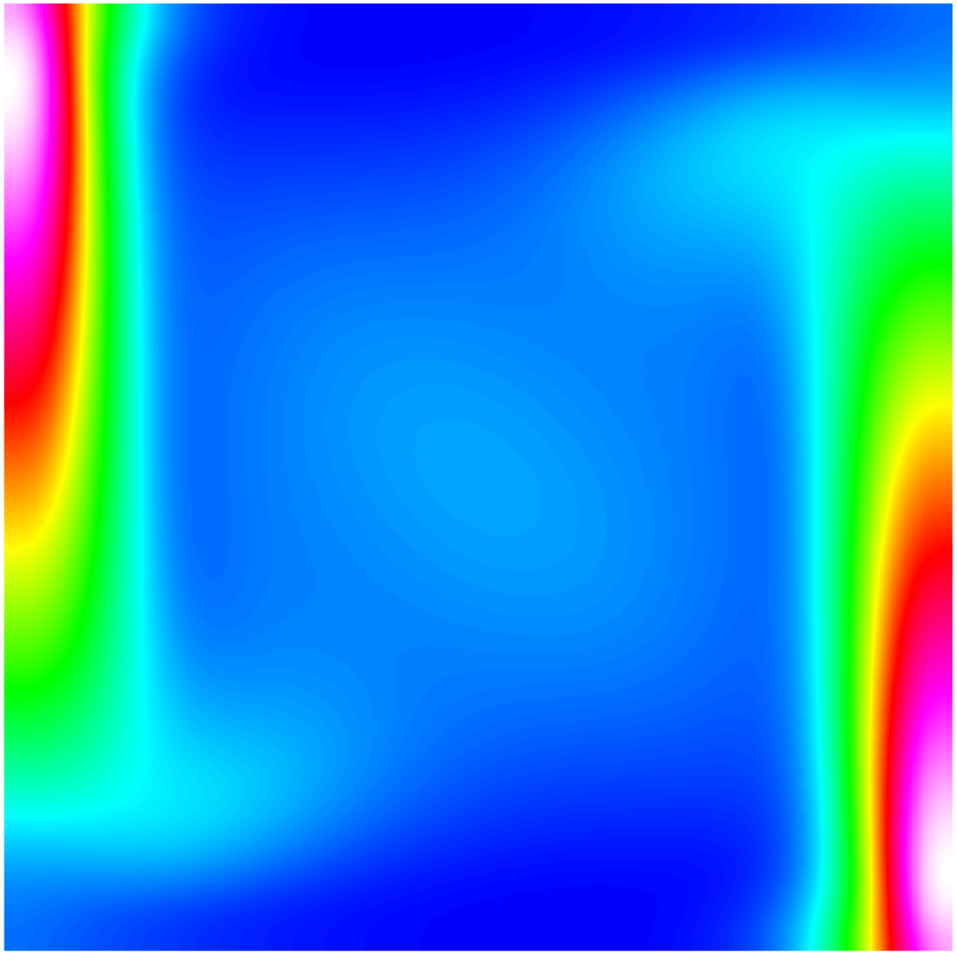
Case: Derived field, vorticity



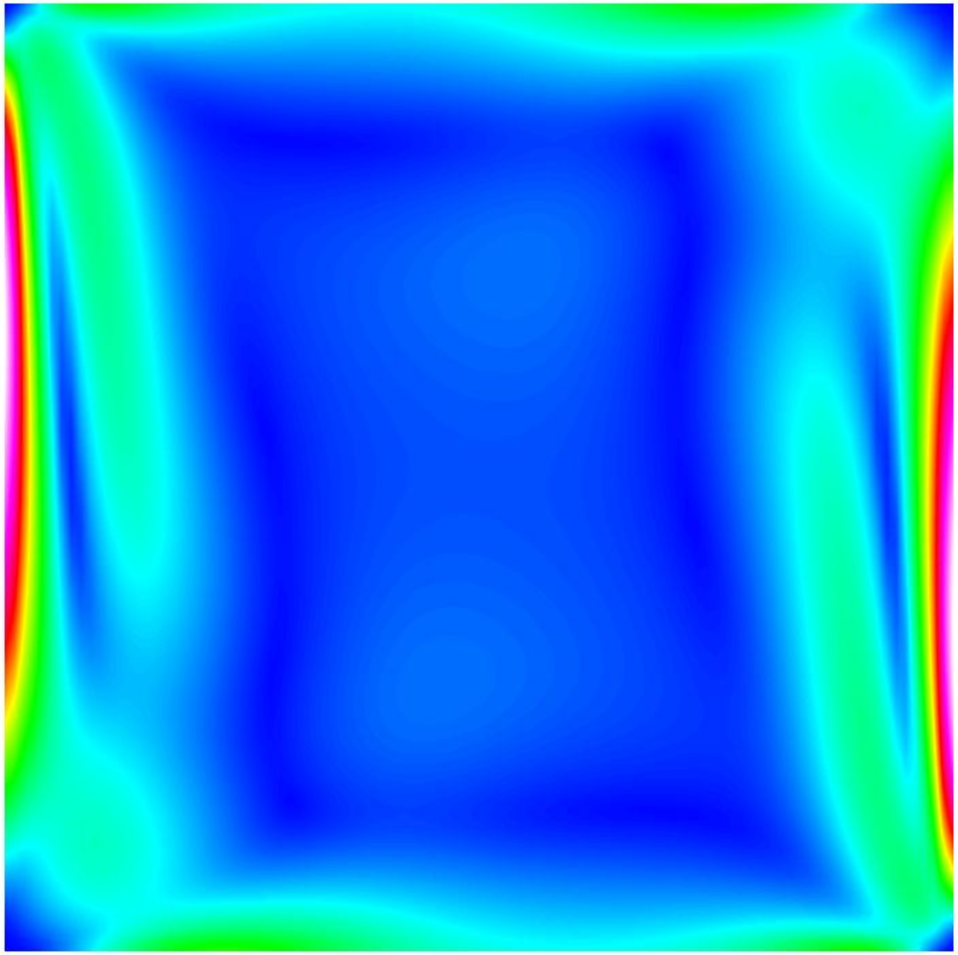
Case: Derived field, Streamlines



Case: Derived field, diffusive flux

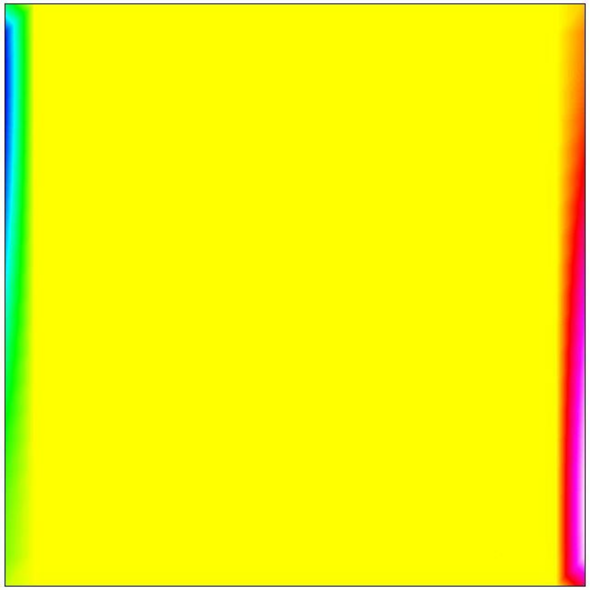


Case: Derived field, Shearrate



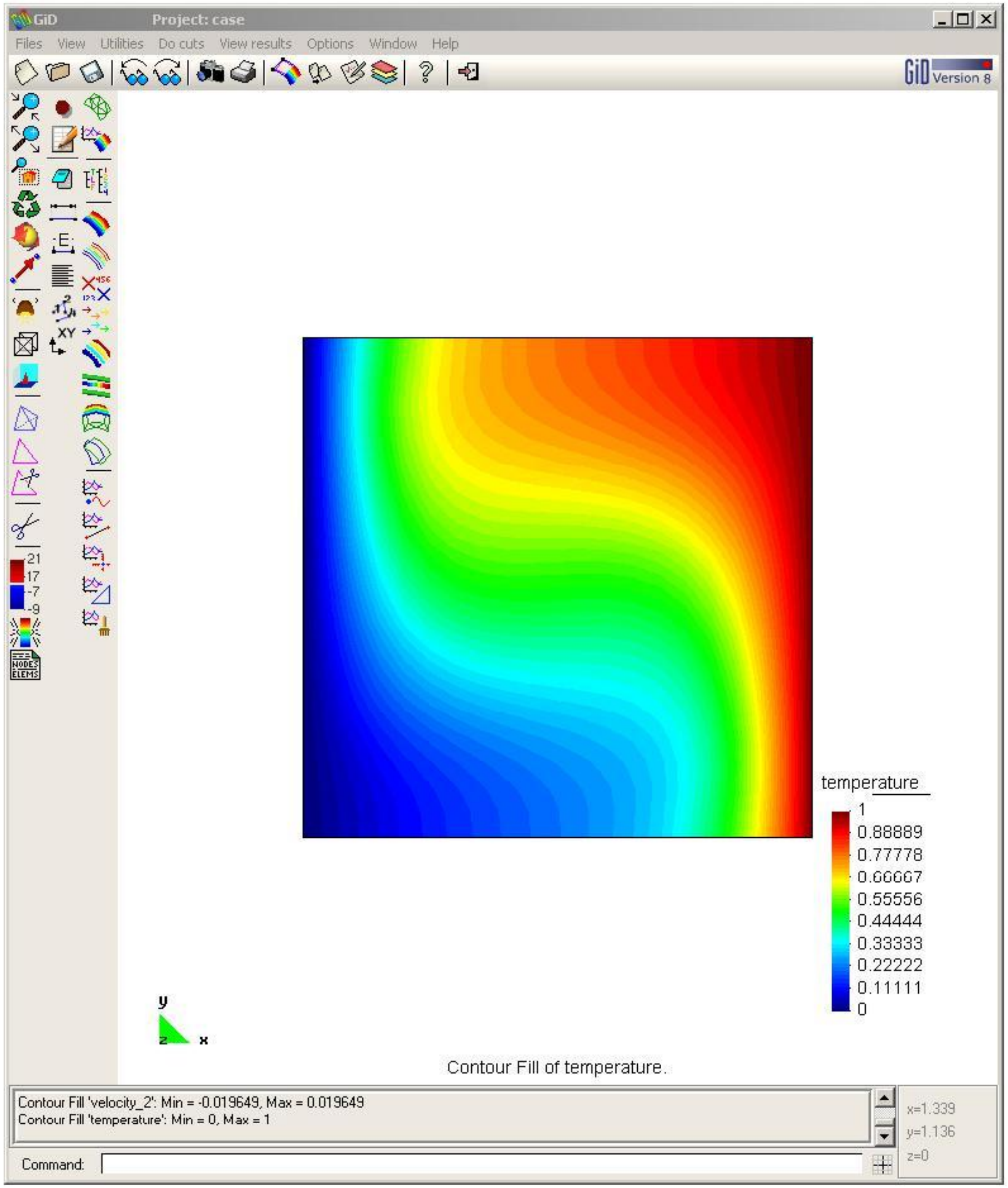
Example: nodal loads

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

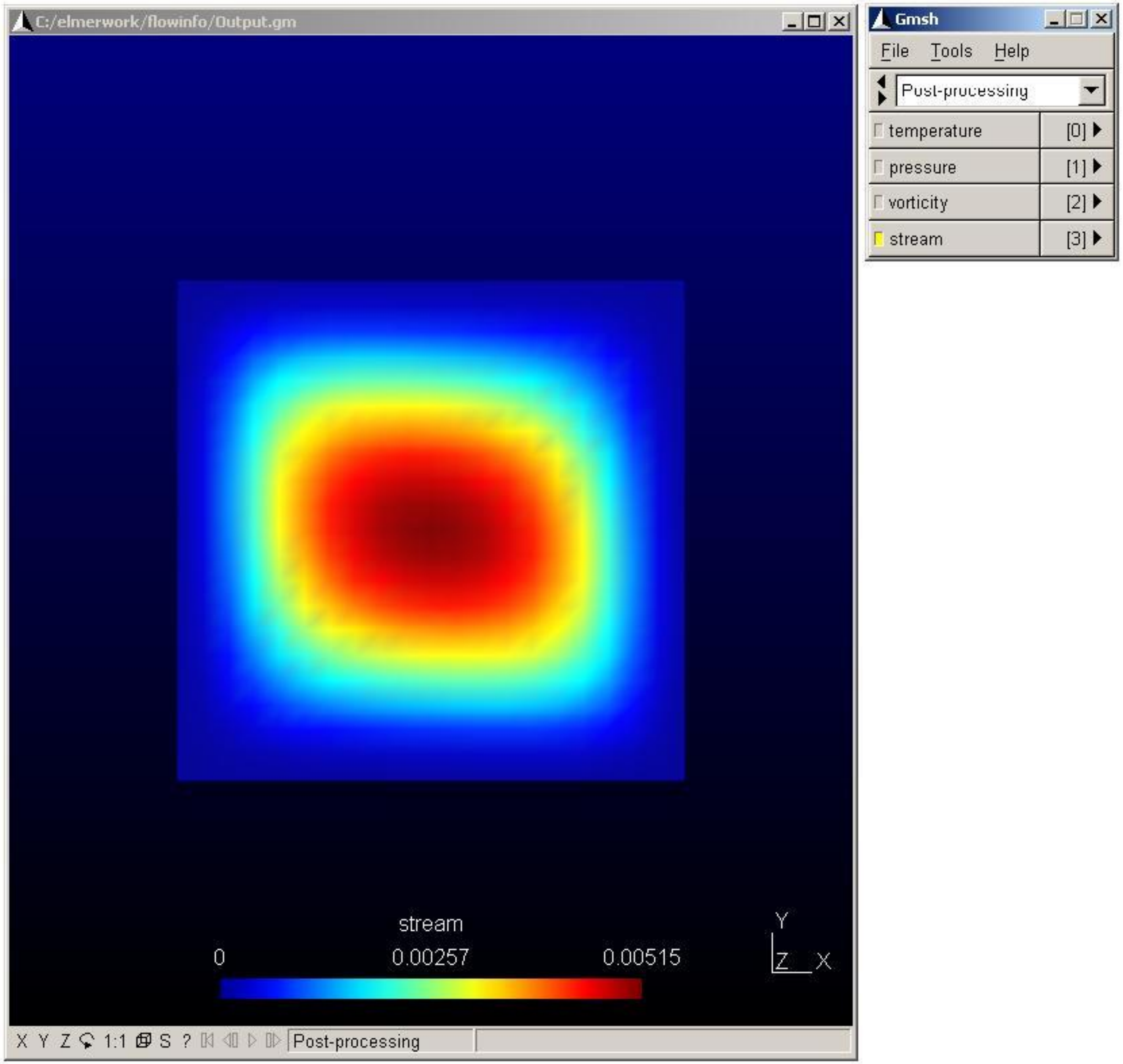


Nodal heat loads

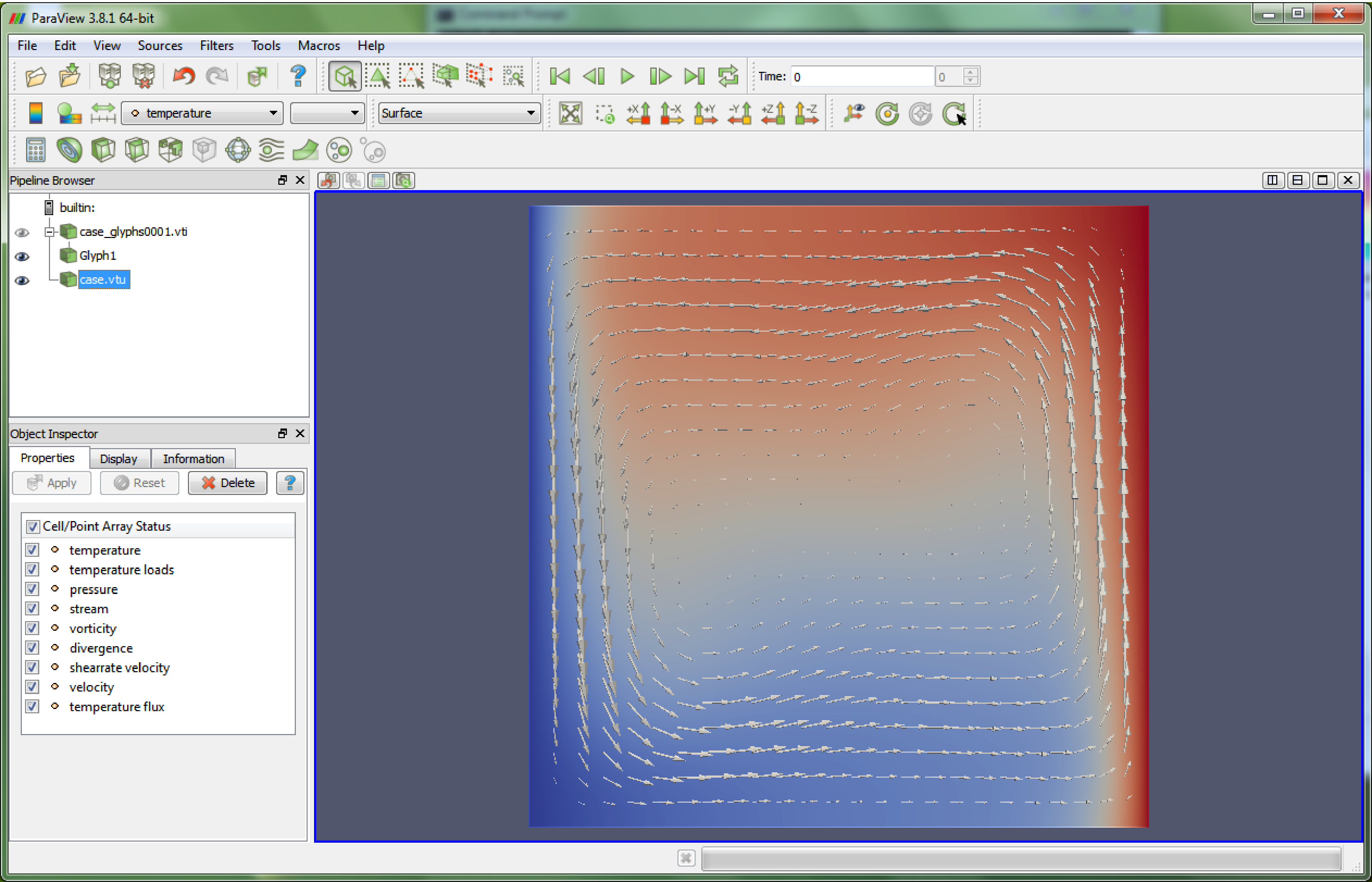
Example:
view in GiD



Example:
view in Gmsh

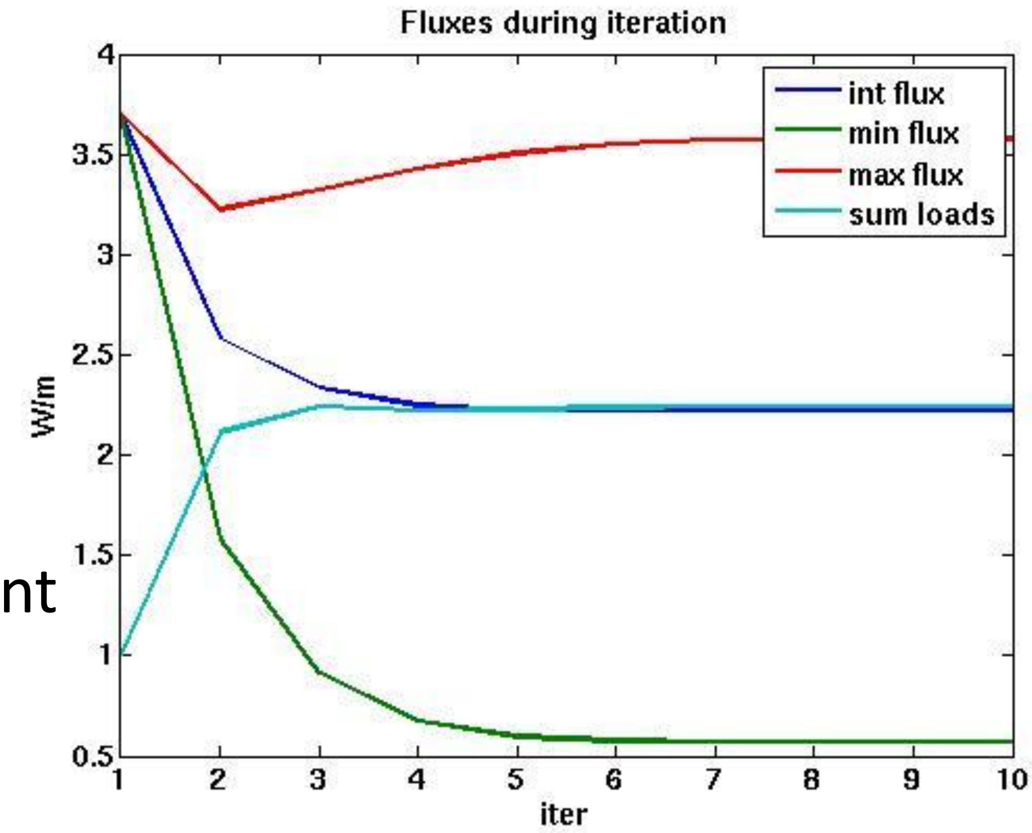


Case: View in Paraview



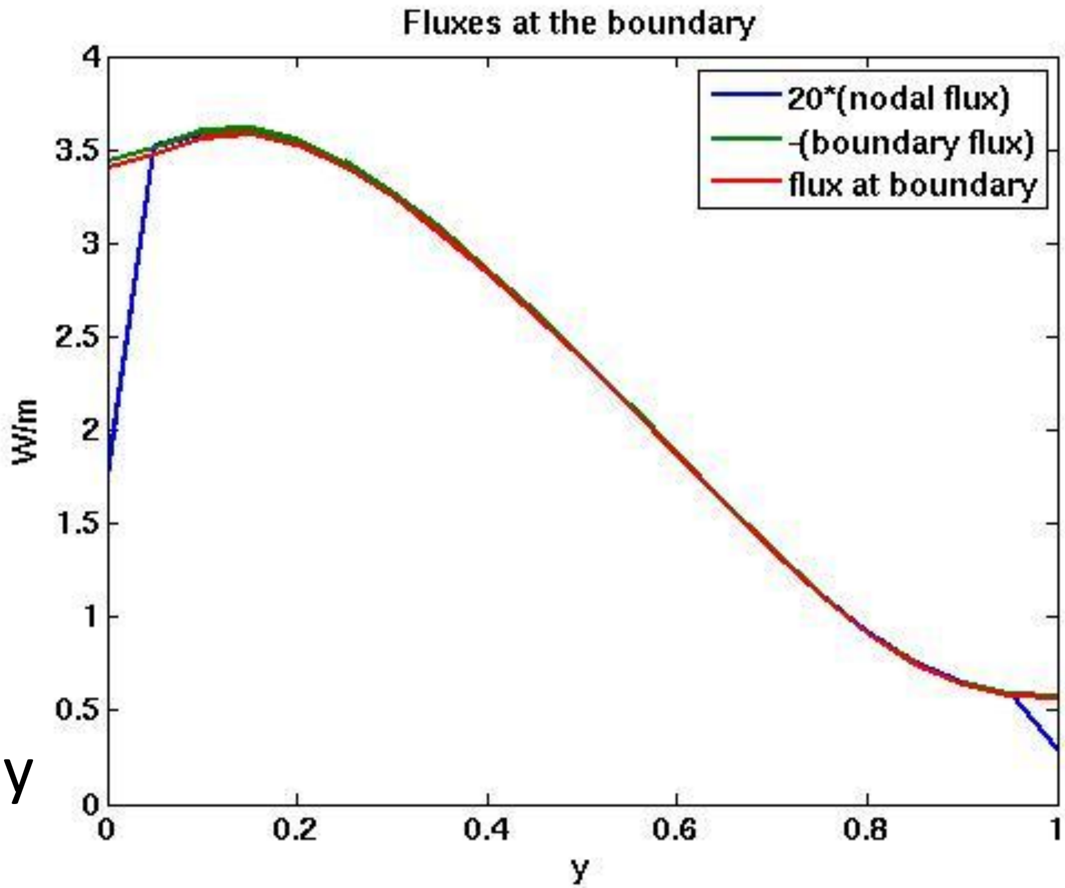
Example: total flux

- 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
- Often the same reduction operations may be done also at later stages but with significantly greater effort