Elmer FEM Webinar Series

CSC, Espoo, Finland
via Zoom
Thursdays
15 EET, 14 CET, 8 ET, 22 JST
Spring 2021

Practical guidelines for the webinar

- For questions that you want answered use the Q&A
  - Will be addressed at the end of the presentation
- Chat may be used for general discussion
  - You may write about your application area, geographic location etc.
- The presentation slides will be made available at
  - https://www.nic.funet.fi/pub/sci/physics/elmer/webinar/
- This webinar will be recorded and will for most parts be available later on youtube

Elmer FEM webinar series - program

- 11.3. Peter Råback & Thomas Zwinger: *Introduction to Elmer & How to teach yourself Elmer*
- 18.3. Peter Råback & Jonathan Velasco*: Overview of capabilities of Elmer - where to go from here?*
- 25.3. Peter Råback & Thomas Zwinger: *Parallel Computing with Elmer*
- 1.4. Juris Vencels: *Elmer-OpenFOAM library*
- 8.4. Eelis Takala & Frederic Trillaud: *Electrical circuits with Elmer with applications*
- 15.4. Mika Malinen: *Solvers for solid mechanics - Recent progress*
- 22.4. Minhaj Zaheer:
  Induction Machine Open-source FEA Computations comparison with Measurement and Commercial FEA
- 29.4. Arved Enders-Seidlitz: *pyelmer - Python interface for Elmer workflow*
- 13.5. Roman Szewczyk, Anna Ostaszewska-Liżewska, Dominika Kopala & Jakub Szałatkiewicz:
  Industrial applications oriented, microwave modelling in Elmer
- Additional slots available: contact organizers if you're interested!

Overview of capabilities of Elmer

Physical models and some of their common features

ElmerTeam
CSC – IT Center for Science, Finland

Elmer FEM webinar

2021

Outline for today

- Overview of physical models
  - From Models Manual
  - Example: 12 Solvers
- Library features used by many/all models
  - Iteration scheme & coupling
  - Generality of fetching Real valued keywords
  - Executions of solver
  - Time dependency modes
  - Bounday conditions
  - Mapping between boundaries and meshes
  - …
- Where to go next?

Example of minimal sif file

```
! Minimal sif file example
Check Keywords "Warn"

Header :: Mesh DB "." "square"

Simulation
  Max Output Level = 5
  Coordinate System = Cartesian
  Simulation Type = Steady
  Output Intervals(1) = 1
  Steady State Max Iterations = 1
  Post File = "case.vtu"
End

Body 1
  Equation = 1
  Material = 1
End

Equation 1
  Active Solvers(1) = 1
End

Solver 1
  Equation = "HeatEq"
  Variable = "Temperature"
  Procedure = "HeatSolve" "HeatSolver"
  Linear System Solver = Direct
End

Material 1
  Heat Conductivity = 1.0
End

Boundary Condition 1
  Name = "Fixed"
  Target Boundaries(1) = 1
  Temperature = 0.0
End

Boundary Condition 2
  Name = "Flux"
  Target Boundaries(1) = 2
  Heat Flux = 1.0
End
```

solver specific keywords

Elmer – abstraction of Solvers

- High level of abstraction ensures flexibility in implementation and simulation
- Solver is an asbtract dynamically loaded object with standard API
  - Solver may be developed and compiled without touching the main library
  - No upper limit to the number of Solvers
- Solvers may be active in different domains, and even meshes
  - Automatic mapping of field values when requested!
- Solvers perform limited well defined tasks
  - Solution of a PDE (roughly 50%)
  - Computing some postprocessed fields
  - Saving of results
  - ....
- Solver may utilize a large selection of services from the library
  - The library has (almost) no knowledge of physical models

Physical Models of Elmer -> Elmer Models Manual

- Heat transfer
  - ✓ Heat equation
  - ✓ Radiation with view factors
  - ✓ convection and phase change
- Fluid mechanics
  - ✓ Navier-Stokes (2D & 3D)
  - ✓ RANS: *SST k-Ω, k-ε, v²-f*
  - ✓ LES: VMS
  - ✓ Thin films: Reynolds (1D & 2D)
- Structural mechanics
  - ✓ General elasticity (unisotropic, lin & nonlin)
  - ✓ Plates & Shells
- Acoustics
  - ✓ Helmholtz
  - ✓ Linearized time-harmonic N-S
  - ✓ Monolithic thermal N-S
- Species transport
  - ✓ Generic convection-diffusion equation
- Electromagnetics
  - ✓ Solvers for either scalar or vector potential (nodal elements)
  - ✓ Edge element based AV solver for magnetic and electric fields
- Mesh movement (Lagrangian)
  - ✓ Extending displacements in free surface problems
  - ✓ ALE formulation
- Level set method (Eulerian)
  - ✓ Free surface defined by a function
- Electrokinetics
  - ✓ Poisson-Boltzmann
- Thermoelectricity
- Quantum mechanics
  - ✓ DFT (Kohn Scham)
- Particle Tracker

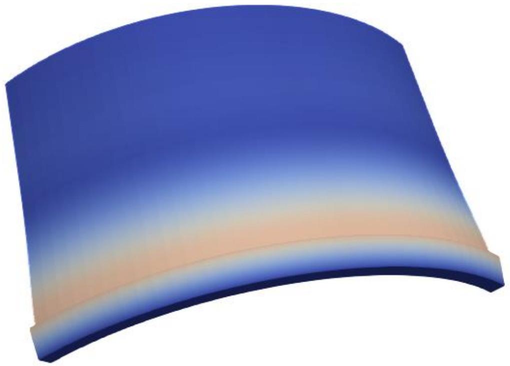
Most important physical modules in Elmer?

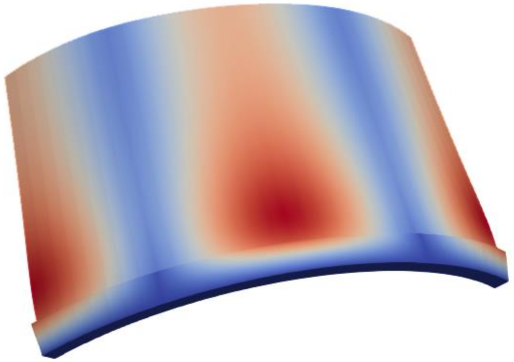
Historically main solver in each field

- **HeatSolve**
  - Heat equation
  - Radiation with view factors
  - convection and phase change
- **FlowSolve**
  - Robust solver for low Re-flows
  - Nonlinear fluids, slip conditions,..
  - Key solver for Elmer/Ice community
- **StressSolve**
  - Versatile solver for linear elasticity
- **WhitneyAVSolver**
  - Hcurl conforming elements
  - Key solver for Elmer/EM community

Solvers with unresolved potential

- **ShellSolver**
  - Enables economical treatment of thin structures
  - Now can be combined with 3D elasticity
- **VectorHelmholtz**
  - Hcurl basis
  - Electromagnetics wave solver
- **ModelMixedPoisson**
  - Hybrid solution employing Hdiv basis
- **ParticleAdvector**
  - Uses particles to advect fields without diffusion
  - Particles & finite elements often a good combination





Series 1. Question 1. (one choice)

- Number of solvers in your most complicated Elmer simulation setup so far?
  - Zero
  - 1
  - 2-3
  - 4-6
  - 7-10
  - >10

Undocumented Models

AllocateSolver.F90 % just dummy solver for allocation
CoordinateTransform.F90 % RotMSolver: using distances create direction
CraigBamptonSolver.F90 % model reduction solver
DCRComplexSolve.F90 % complex diffusion-convection-reaction
DFTSolver.F90 % charge density using the eigenvectors
DirectionSolver.F90
DistanceSolve.F90 % compute distance in two methods
DistributeSource.F90 % local to global mesh sources
ElementSizeSolver.F90 % element size with Galerkin
EliminateDirichlet.F90
EliminatePeriodic.F90
EnergyRelease.F90 % energy release rate for crack propagation
FacetShellSolve.F90
FDiffusion3D.F90 % Complex nodal equation for vector fields
FDiffusion.F90 % Complex nodal equation for scalar fields
HeatSolveVec.F90 % new generation HeatSolve
HelmholtzProjection.F90 % purifying vector potential
IncompressibleNSVec.F90 % new generation flowsolve
KESolver.F90 % turbulence
Komega.F90 % turbulence
Mesh2MeshSolver.F90 % interpolation
MeshChecksum.F90 % utility to check mesh consistency
ModelPDE.F90 % simple advection-diffusion-reaction
NormalSolver.F90 % calculate normal

OdeSolver.F90 % ordinary differential equation solver
PartitionMesh.F90 % partition mesh solver
PoissonDG.F90
Poisson.F90
RigidBodyReduction.F90 % reduction of rigid pieces
SaveMesh.F90 % mesh saving
ScannedFieldSolver.F90 % treating scanned field solutions
ShallowWaterNS.F90
ShearrateSolver.F90
Spalart-Allmaras.F90 % turbulence
SSTKomega.F90 % turbulence
StatCurrentSolveVec.F90 % new version of StatCurrentSolve
ThermoElectricSolver.F90 % strongly coupled thermal & electrostatics
TransientCost.F90 % integral over cost
UMATLib.F90
V2FSolver.F90 % turbulence
WPotentialSolver.F90 % potential for directions

- 76 models have been documented
- Tens of modules have not been documented!
- Some are of little use but might find users if people would find them.
- 5 undocumented turbulence models exist but they are not that robust…

Example: TwelveSolvers2D

- The purpose of the example is to show how number of different solvers are used
- 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
- Added now also as test case

- Square with hot and cold walls
- Filled with viscous fluid
- Bouyancy modeled with Boussinesq approximation
- Temperature difference initiates a convection roll

test case: TwelveSolvers2D

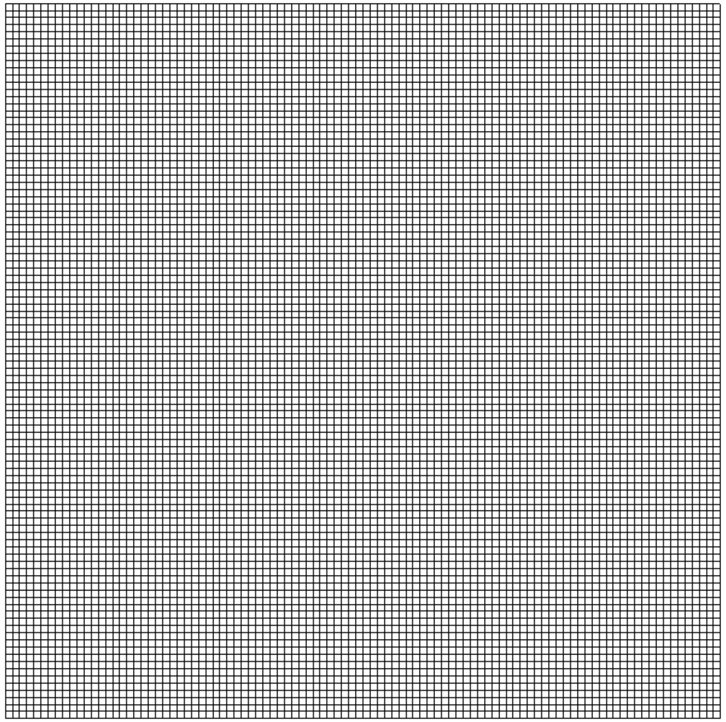
Example: the 12 solvers

1. HeatSolver
2. FlowSolver

Weakly coupled system iterated until convergence

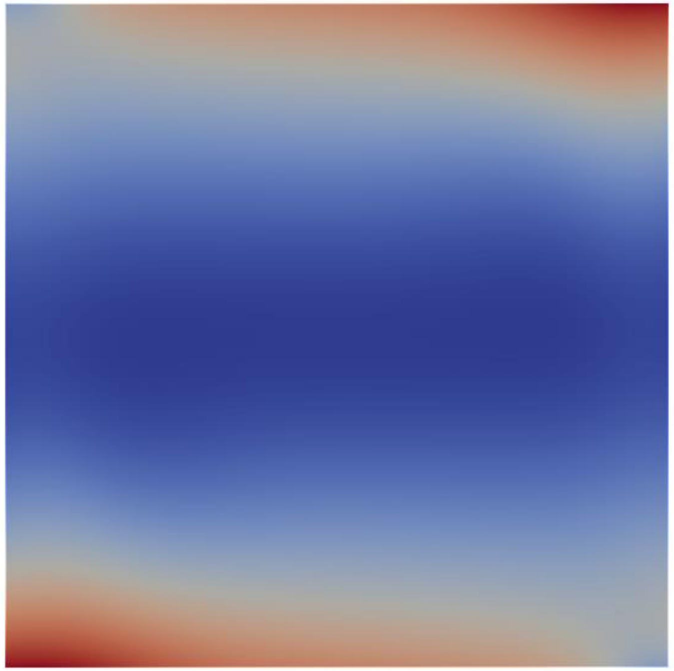
---

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

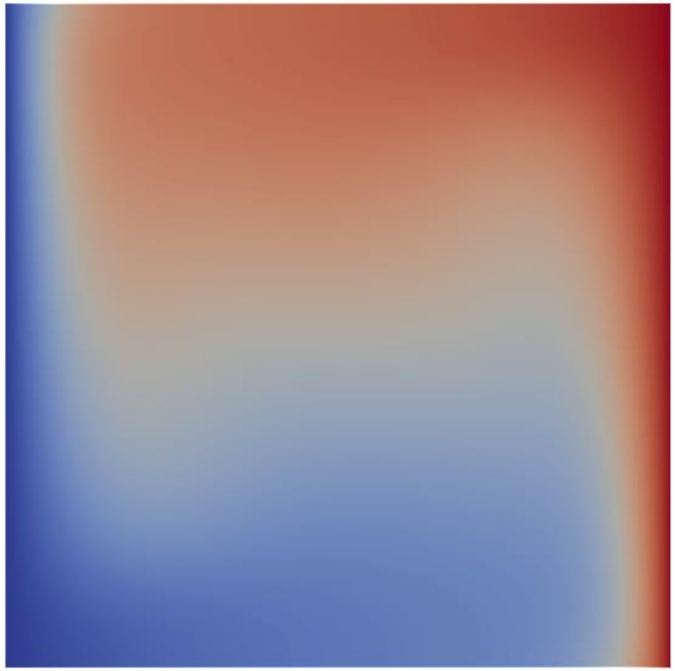


Mesh of 10000 bilinear elements

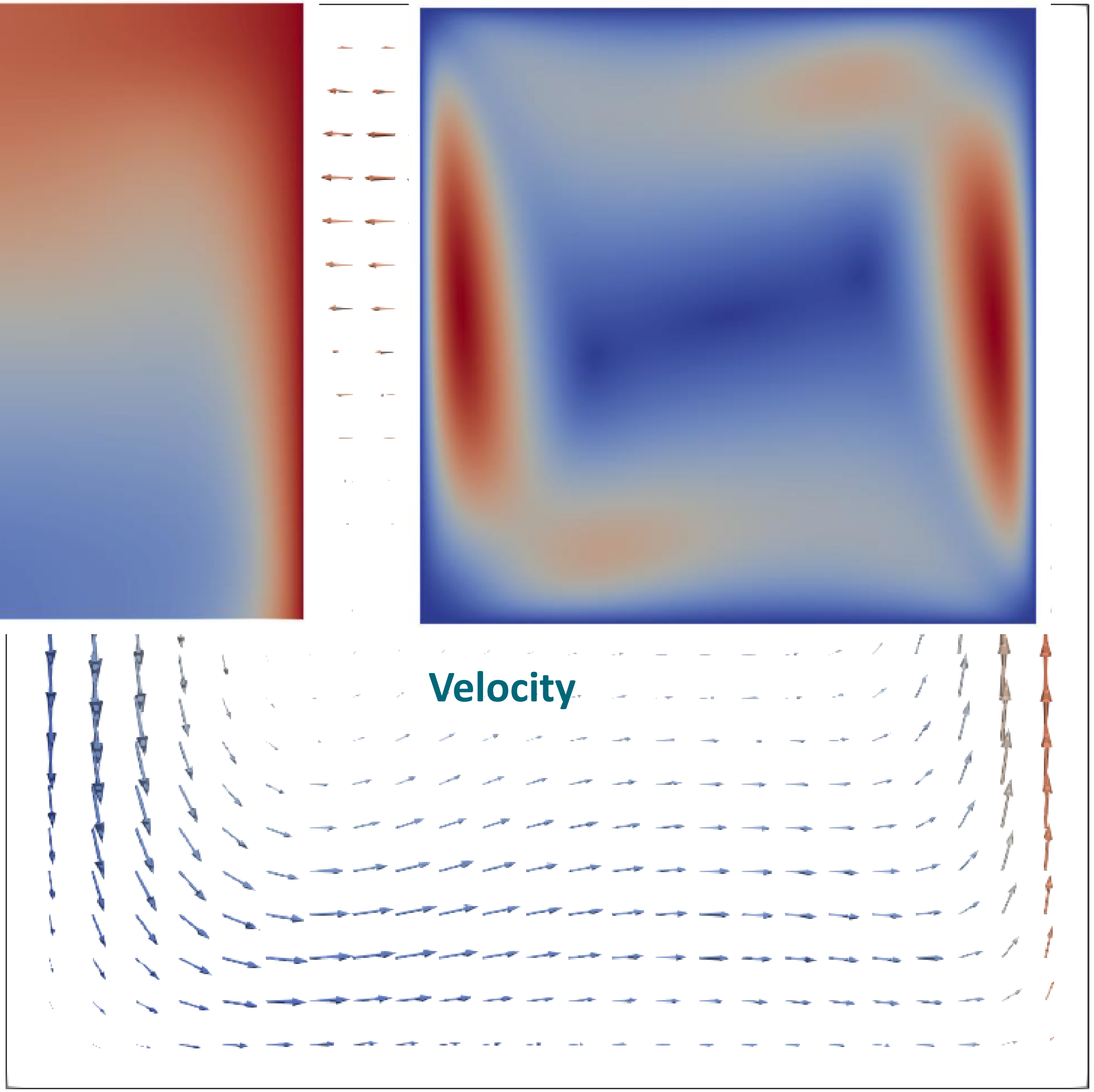
Example: Primary fields for natural convection



Pressure

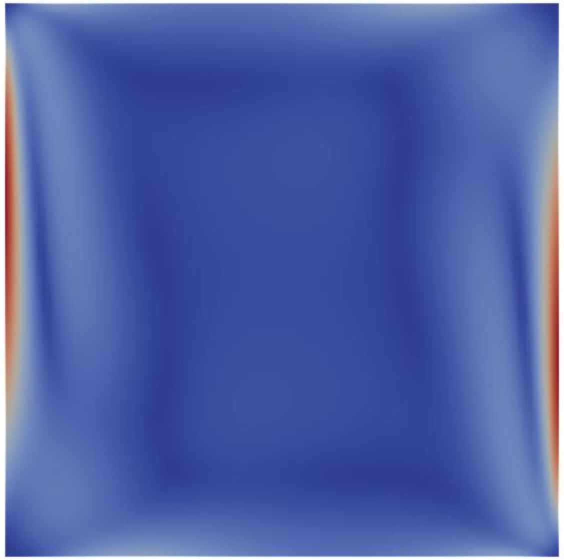


Temperature

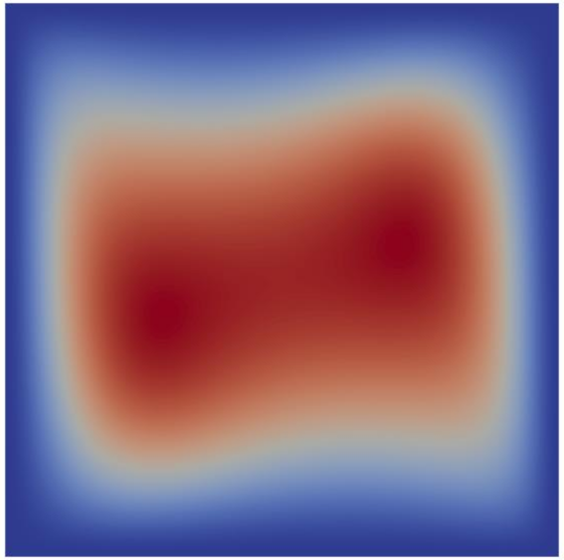


Velocity

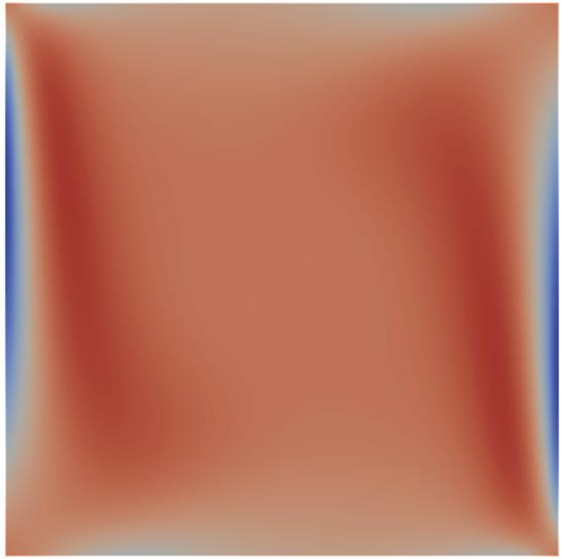
Example: Derived fields for Navier-Stokes solution



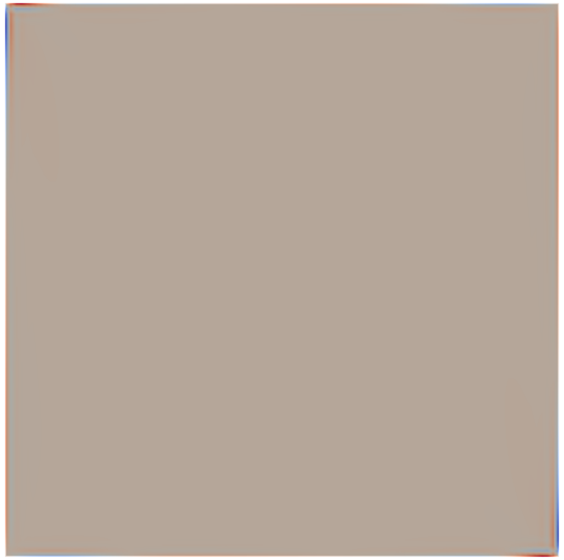
Shearrate field



Stream function

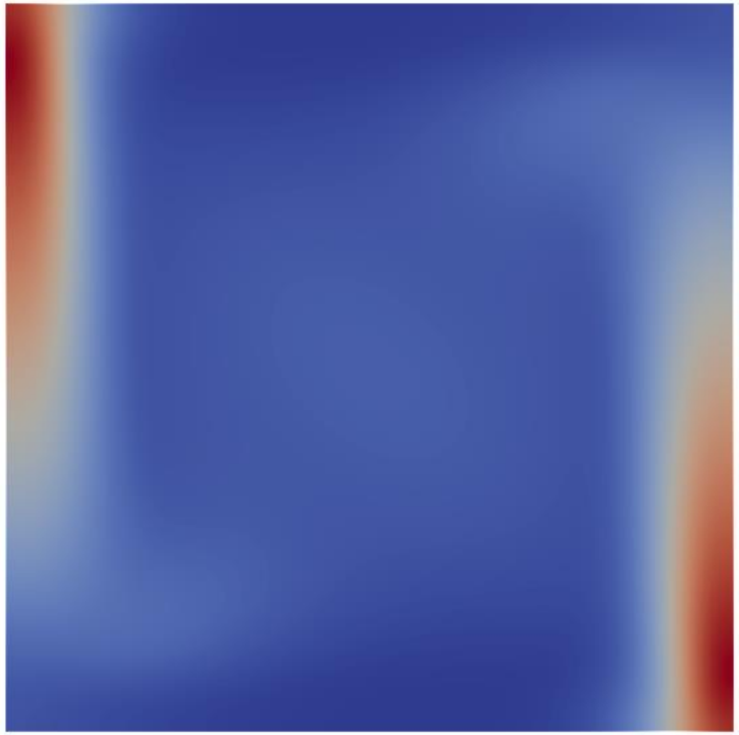


Vorticity field

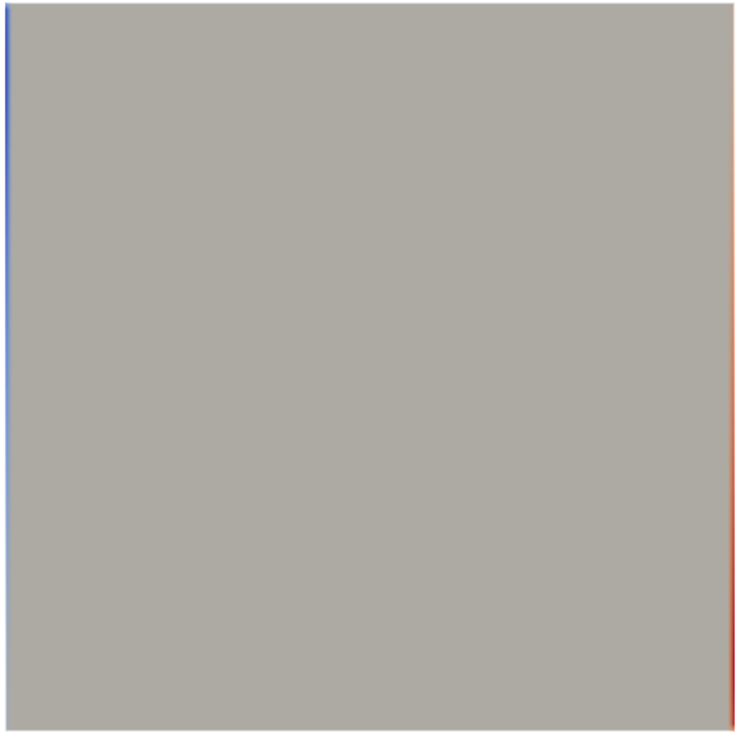


Divergence field

Example: Derived fields for heat equation



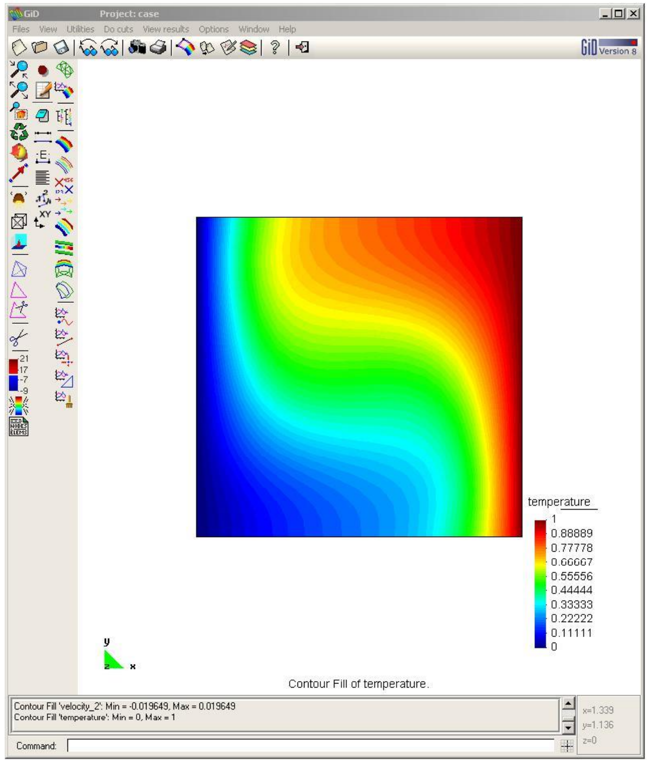
Heat flux

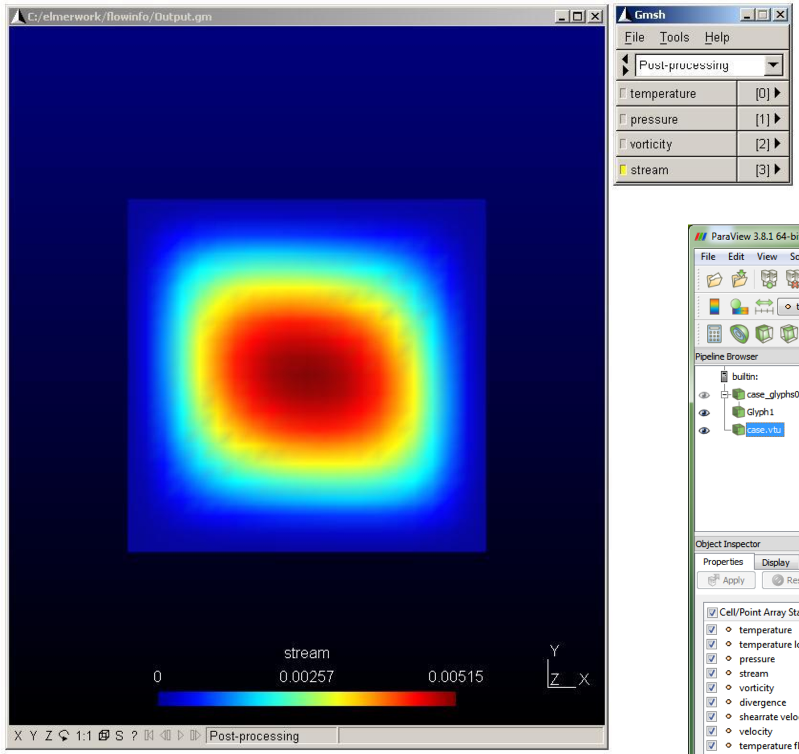


Nodal heat loads

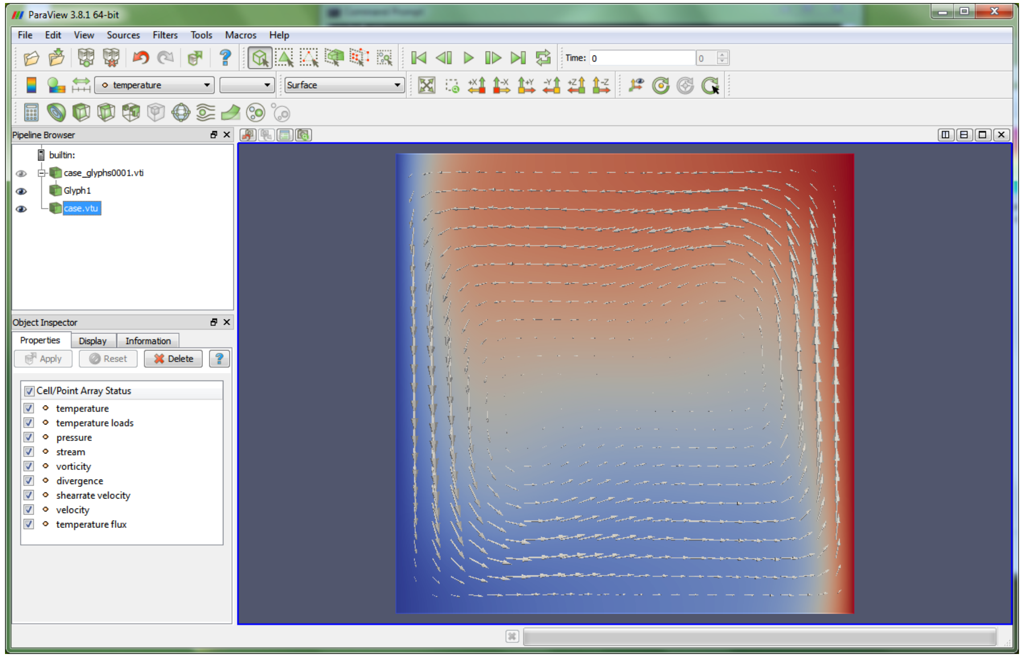
- Nodal loads only occur at boundaries (nonzero heat source)
- Nodal loads are associated to continuous heat flux by element size factor

Example: Visualization in different postprocessors





gmsh

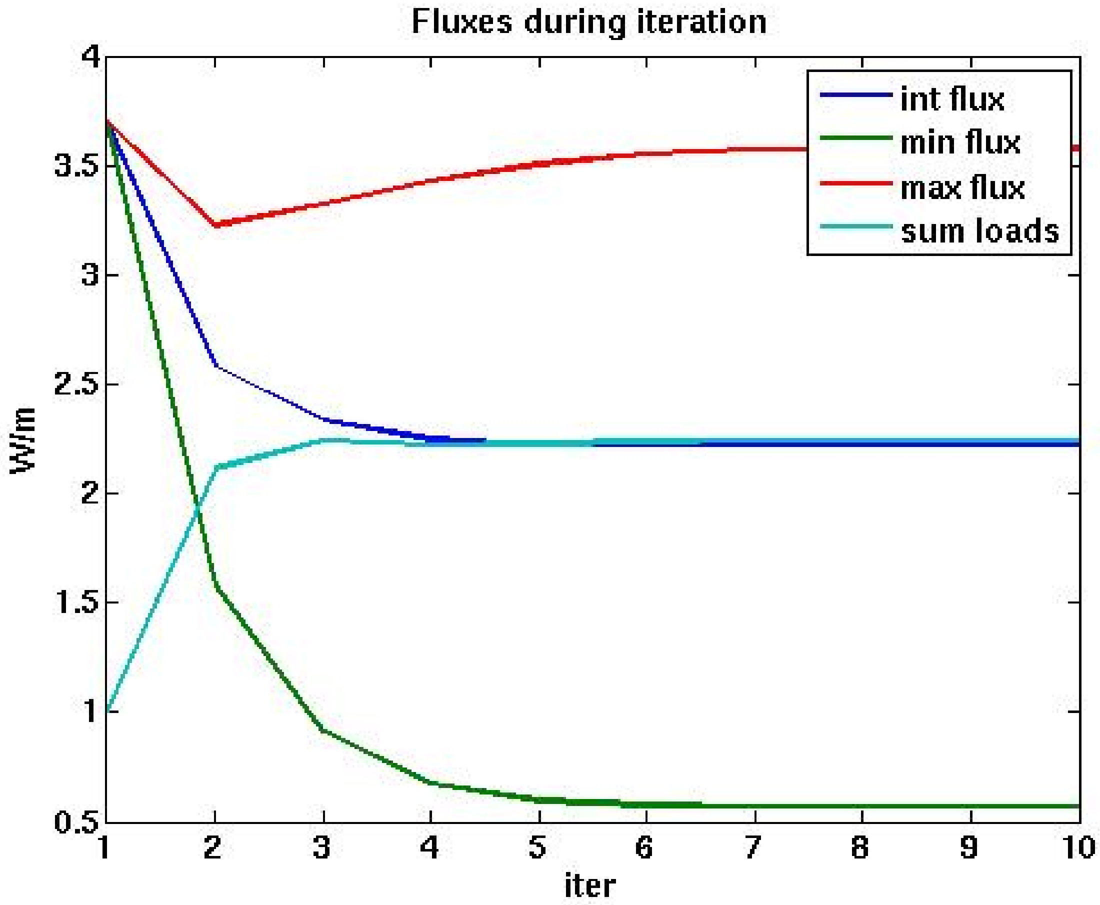


GiD

Paraview

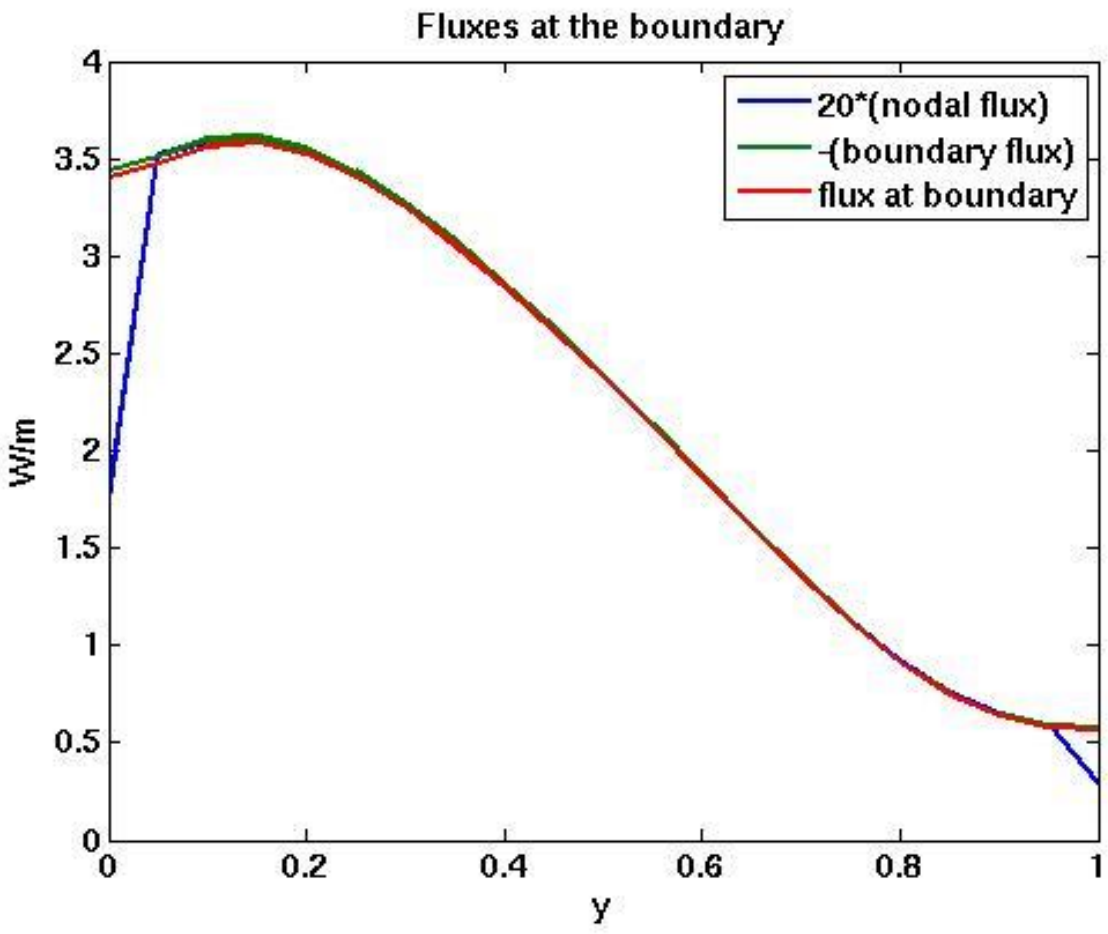
Example: total flux

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



Example: boundary flux

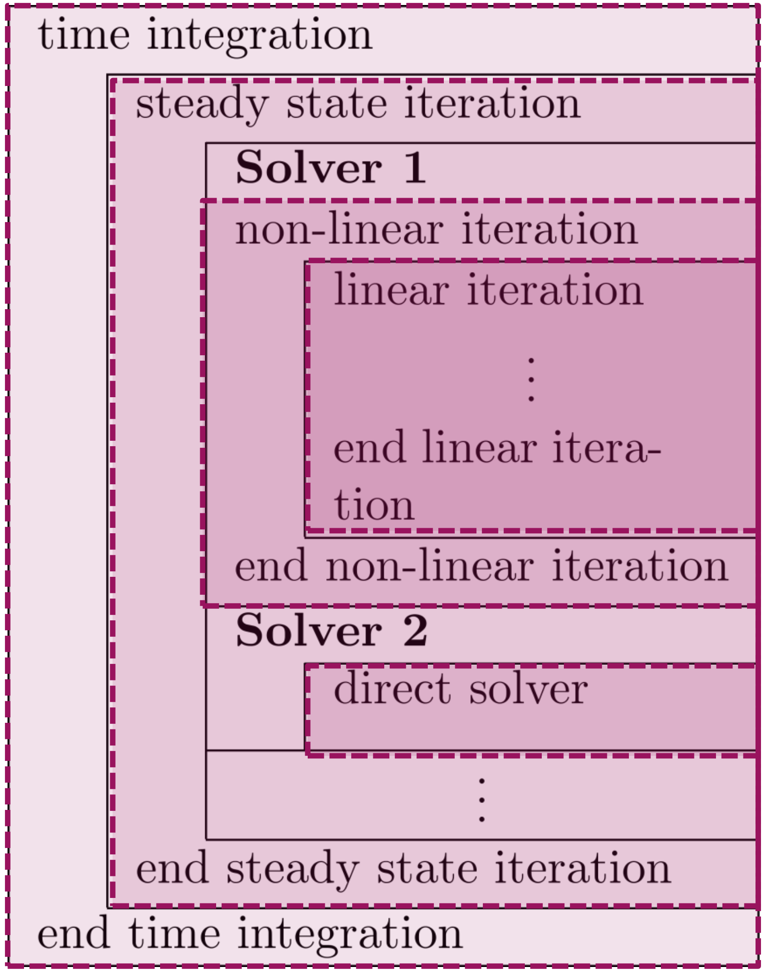
- Saved by SaveLine
- Three ways of computing the boundary flux give different approximations
- At the corner the nodal flux should be normalized using only *h/2*



Some common features for PDE solvers

- Iteration scheme & coupling: linear, nonlinear & steady state level
- Generalized fetching of keywords
- Execution of Solvers
- Time dependency modes
- Finite element basis
- Dirichlet BCs
- Nodal loads
- Shared boundary conditions
- Overlapping meshes
- …

Nested iterations in Elmer as defined by the SIF file



1. `Timestep Intervals`
2. `Steady State Max Iterations`
3. `Nonlinear Max Iterations`
4. `Linear System Max Iterations`
4. `Linear System Convergence Tolerance`
3. `Nonlinear System Convergence Tolerance`
2. `Steady State Convergence Tolerance`
1.

Solution of linear system

- Keywords starting with "**Linear System**"
- The lowest level operation
- Tens of different techqniques in serial and parallel
- We will go through these next week!

Solution of nonlinear system

- Keywords starting with "**Nonlinear System**"
- The level for iterating over one single nonlinear equation
  - By default ElmerGUI assumes nonlinear iteration => always two iterations

```
Solver i
  Nonlinear System Max Iterations = Integer
  Nonlinear System Convergence Tolerance = Real
  Nonlinear System Relaxation Factor = Real
  Nonlinear System Convergence Measure = String
  Nonlinear System Newton After Tolerance = Real
  Nonlinear System Newton After Iterations = Real
  Nonlinear system consistent norm = Logical
  …
```

$$u_i^{'} = \lambda u_i + (1-\lambda) u_{i-1}$$

"norm"

$$\delta = 2 * ||u_i| - |u_{i-1}|| / (|u_i| + |u_{i-1}|)$$

"solution"

$$\delta = 2 * |u_i - u_{i-1}| / (|u_i| + |u_{i-1}|)$$

"residual"

$$\delta = |Ax_{i-1} - b| / |b|$$

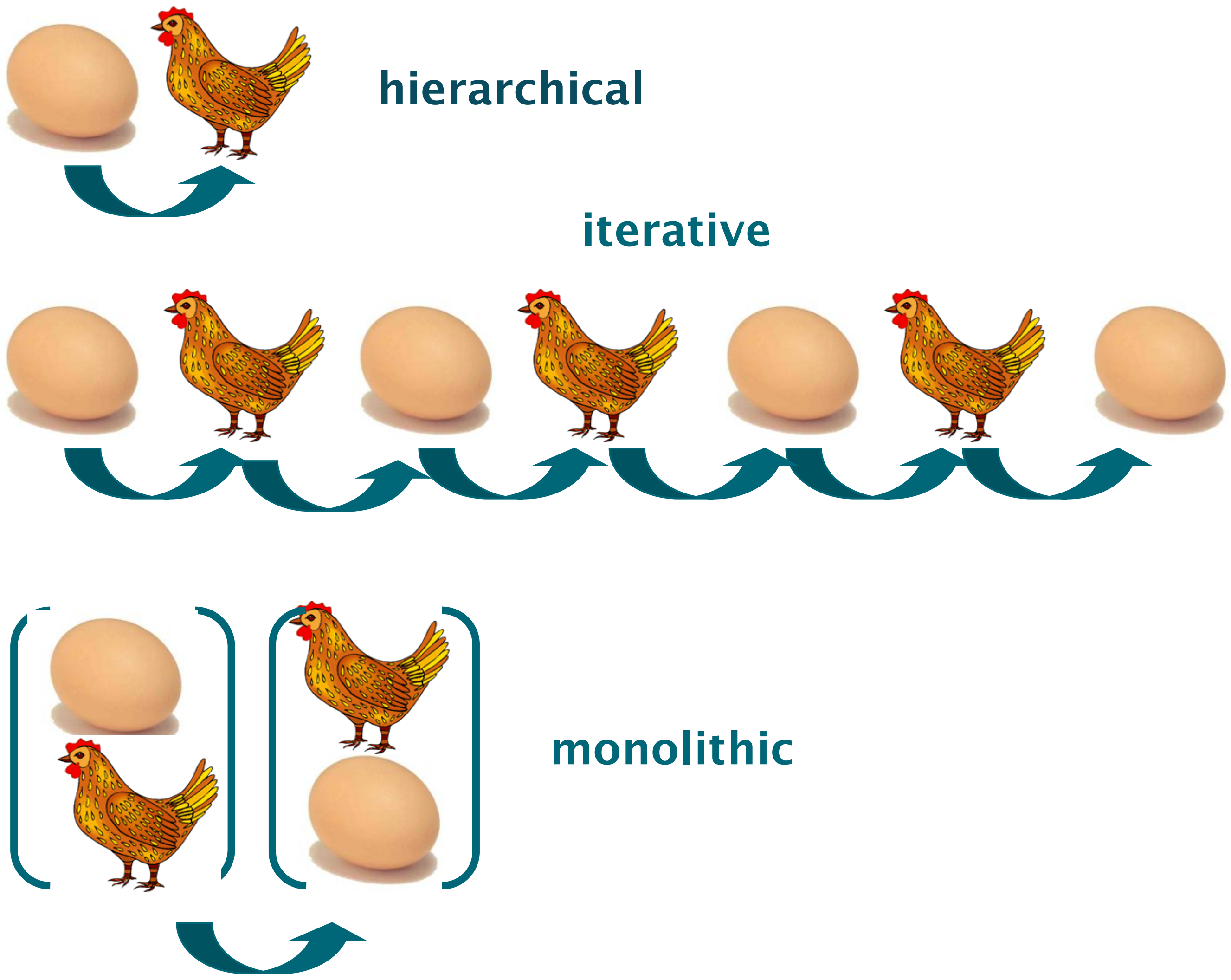
Solution of coupled system

- Keywords starting with "**Steady State**"
- The level for iterating over set of Solvers to find the solution satisfying all of them

Simulation
Steady State Max Iterations = Integer
Steady State Min Iterations = Integer

Solver i
Steady State Convergence Tolerance = Real
Steady State Relaxation Factor = Real
Steady State Convergence Measure = Real
…

Solution strategies for coupled problems



Assume phenomena $\mathcal{F}$ and $\mathcal{G}$ that both depend on field variables x and y.
Solution is obtained from a system of equations, $f(x, y) = 0$ and $g(y, x) = 0$.

one-directional coupling $\Rightarrow$ hierarchical solution

$$\begin{aligned} & f(x_1) &= 0 \\ \Rightarrow \quad & g(y_1, x_1) &= 0 \end{aligned}$$

weak coupling $\Rightarrow$ iterative or segregated solution

$$\begin{cases} f(x_{m+1}, y_m) &= 0 \\ g(y_{m+1}, x_{m+1}) &= 0 \end{cases}$$

strong coupling $\Rightarrow$ monolithic solution

$$\begin{bmatrix} f(x_{m+1}, y_{m+1}) \\ g(y_{m+1}, x_{m+1}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Monolithic approach requires iteration if either f or g is nonlinear.

Weak coupling in Elmer

- Parameters in equations depend on field values
  - Nonlinear iteration within on Solver
  - Coupled system iteration among solvers
- Consistency in ensured within nested iterations
- E.g. case of natural convection
  - Force on the Navier-Stokes depends on the temperature of the heat equation
  - Convection velocity in the heat equation depends on the solution of the Navier-Stokes equation
- Some dependencies have been ”coded in” while others take use of the generic way to give **Real** valued keywords in Elmer

Real valued keyword functions

1) Tables can be use to define a piecewise linear (or cubic) dependency of a variable

```
Density = Variable Temperature
    Real cubic
     173  990
     273 1000
     373 1010
    End
```

Inside range: Interpolation

Outside range: Extrapolation!

2) MATC: a library for numerical evaluation of mathematical expressions

```
Density = Variable Temperature
  MATC "1000*(1 - 1.0e-4*(tx(0)-273.0))"
```

or as constant expressions

3) LUA: external library, faster than MATC

```
Density = Variable Temperature
  LUA "1000*(1 - 1.0e-4*(tx[0]-273.0))"
```

4) User defined function

```
Density = Variable Temperature
  Procedure "mymodule" "myproc"
```

Four ways to present:

$$\rho = \rho_0(1 - \alpha(T - T_0)))$$

Example of F90 User Function

File mymodule.F90:

```
FUNCTION myproc( Model, n, T ) RESULT(dens)
USE DefUtils
IMPLICIT None
TYPE(Model_t) :: Model
INTEGER :: n
REAL(KIND=dp) :: T, dens

    dens = 1000*(1-1.0d-4 *(T-273.0_dp))

END FUNCTION myproc
```

Compilation script comes with installation: **`elmerf90`**

Linux

`$ elmerf90 mymodule.F90 -o mymodule.so`

Windows

`$ elmerf90 mymodule.F90 -o mymodule.dll`

ElmerSolver - Controlling execution order of Solvers

- By default each a Solver is executed in order of their numbering numbering
- "**Exec Solver**" keyword can be used to alter this
  - "**always**" – execute in the coupled system loop of the nested iteration
  - "**before all**" or "**before simulation**" -
  - "**after all**" or "**after simulation**" – perform something
  - "**before saving**" – perform before saving sequence, maybe compute something for saving
  - "**after saving**" – perform after saving sequence, maybe save someting
  - "**before timestep**" – perform before timestep only
  - "**after timestep**" – perform after timestep only
  - "**never**" – skip solver for debugging etc.
- "**Slave solver**" slots (rather new feature) may be used to have some master Solver call other solvers within their execution.
  - Added flexibility to complex cases

ElmerSolver – Time dependency modes

- Transient simulation
  - 1st order PDEs:
    - Backward differences formulae (BDF) up to 6th degree
    - Newmark Beta (Cranck-Nicolsen with β=0.5)
    - 2nd order Runge-Kutta
    - Adaptive timestepping
  - 2nd order PDEs:
    - Bossak
- Steady-state simulation
- Scanning
  - Special mode for parametric studies etc.
- Harmonic simulation
- Eigenmode simulation
  - Utilizes (P)Arpack library

```
Simulation
  Simulation Type = Transient
  Timestep Intervals = 100
  Timestep Sizes = 0.1
  Timestepping Method = implicit euler
```

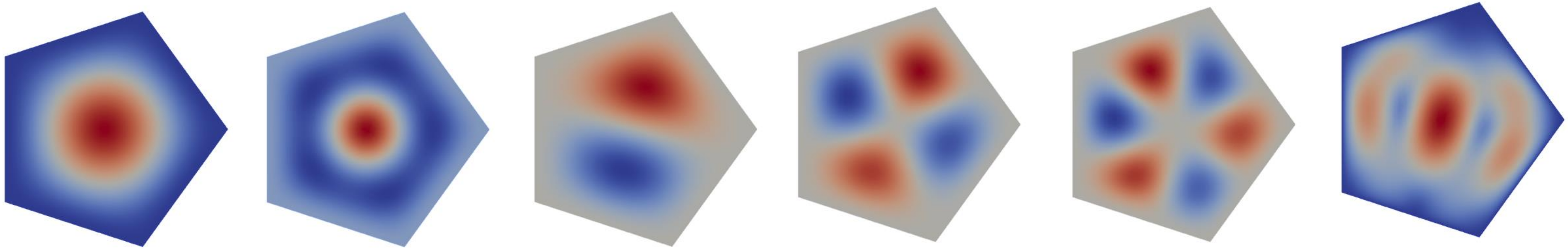
```
Simulation Type = Steady
```

```
Simulation Type = Scanning
```

Eigen Analysis

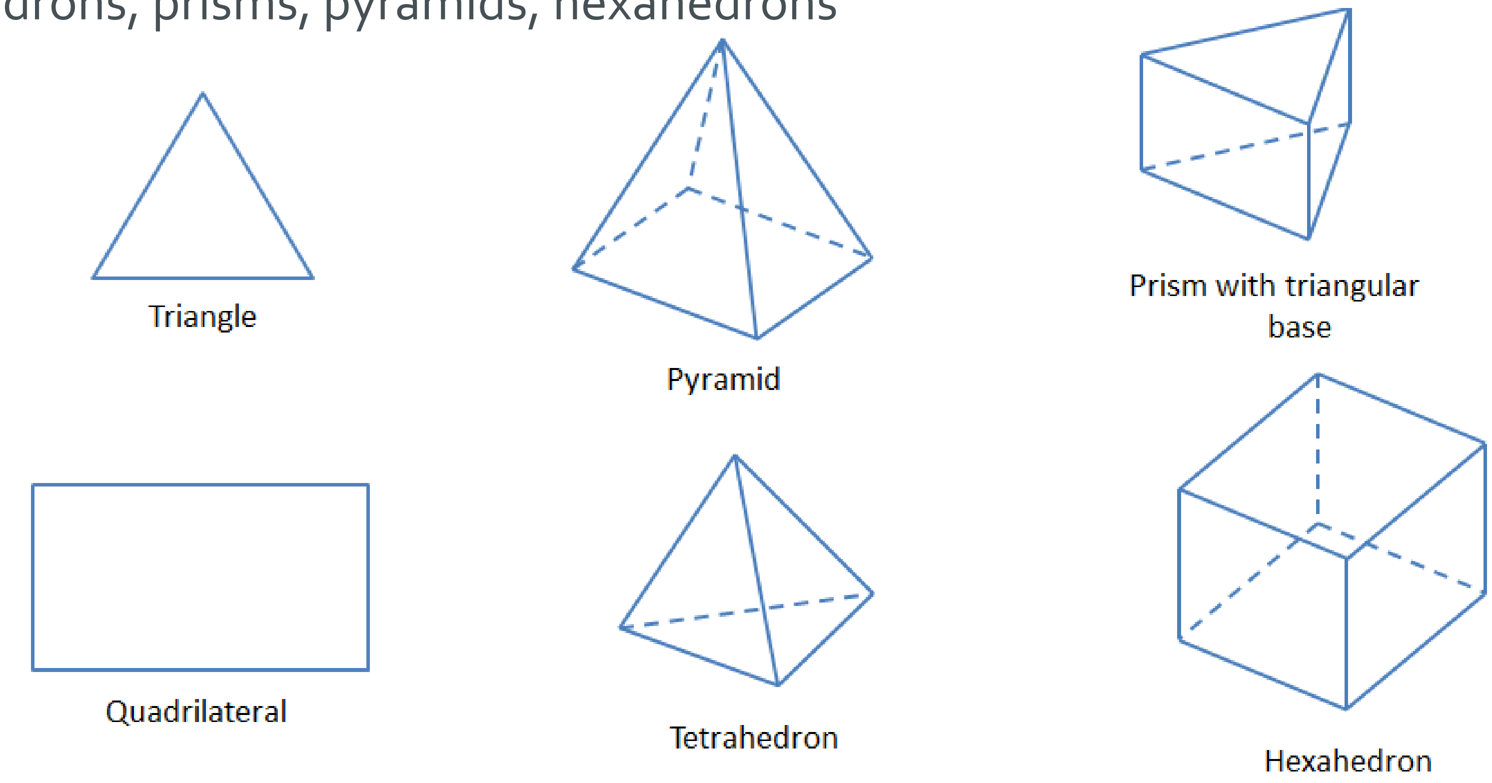
- Any 2nd order PDE may be solved as an eigen system
  - d/dt -> iω
- Example, eigenmodes from Smitc, the plate equation

```
Solver i
  Eigen Analysis = True
  Eigen System Values = 10
  Eigen System Convergence Tolerance = 1.0e-6
  Eigen System Select = Smallest Magnitude
```



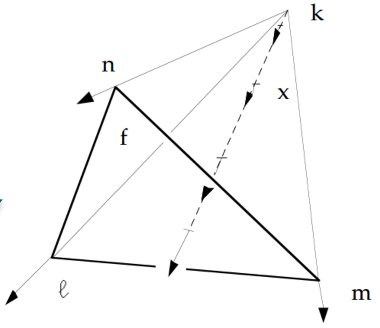
ElmerSolver – Finite element shapes

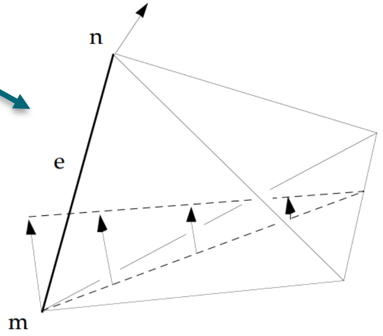
- Element shapes are define already in the mesh files
- 0D: vertex
- 1D: edge
- 2D: triangles, quadrilateral
- 3D: tetrahedrons, prisms, pyramids, hexahedrons



ElmerSolver – Finite element basis functions

- Element types and formulations are applied on elements Solver-wise
- Element families
  - Nodal (up to 2-4th degree)
  - p-elements (hierarchical basis)
  - Edge & face –elements
  - H(div) - often associated with”face” elements)
  - H(curl) - often associated with ”edge” elements)
- Formulations
  - Galerkin, Discontinuous Galerkin
  - Stabilization
  - Residual free bubbles





Examples of FEM basis

- Different equations may require different basis functions beyond the standard nodal finite element basis
- Solvers supporting p-elements (here 3rd order), e.g. **ModelPDE**
  - **Element = p:3**
- Lowest order ***Hcurl*** elements for **WhitneyAVSolver**
  - **Element = n:1 e:1** ! Hidden from end-user
- Lowest order ***Hdiv*** elements for **ModelMixedPoisson**
  - **Element = n:0 -tetra b:1 -brick b:25 -quad_face b:4 -tri_face b:1** ! Hidded from end-user
- …

ElmerSolver: Exported Variables

- Any solver may allocate additional variables called "**Exported Variables**"
- These may be used for various uses, for example create derived fields easily
  - May be used in the same way for "**Initial Condition**" as regular variables.
  - May be updated if defined in "**Body Force**" section and "**Update Exported Variables**" is requested.
- Often defined inside the code
- Exported variables may be of different types
  - **-nodal, -elem, -dg, -ip**

$$k = Ae^{\frac{-E_a}{RT}}$$

A – Arrhenius constant (frequency factor)

E_a – activation energy (J mol^{-1})

R – gas constant (8.31 J K^{-1} mol^{-1})

```
#R=8.31
#Ea=123.4
#A=5.67e-3

Solver i
  Exported Variable 1 = Rate
  Updated Exported Variables = True

Body Force j
  Rate = Variable "Tempeture"
    Real LUA "A0*exp(-Eact/(R*tx[0]))"
```

ElmerSolver - Dirichlet Conditions

- Dirichlet keywords are set by the library for "**Varname**"
  - **Temperature = 273.0**
  - **Velocity = Variable "Coordinate 2"; Real LUA "4*tx[0]*(1-tx[0])"**
  - **AV {e} = 0.0**   ! For edge degree of freedom
- Conditional Dirichelt conditions "**Varname Condition**"
  - Applied only when condition is positive
  - **Temperature = 273.0**
    Temperature Condition = Equals "Velocity 1" ! Set temperature for inflow only
- Boundary Condition i
- Body Force i
  - Enables bodywise Dirichlet conditions also

ElmerSolver – Computing nodal forces

- ElmerSolver allows for automatic computation of nodal forces from matrix equation: **$f=A_0x-b$**
- These are reactions to Dirichlet conditions that give equivalent r.h.s. terms that would result in exactly the same solution
  - **HeatSolver**: nodal heat flux (**Joule**)
  - **FlowSolve**: nodal force (**Newton**)
  - **StressSolve**: nodal force (**Newton**)
  - **StatElecSolve**: nodal charge (**Coulomb**)
  - **StatCurrentSolve**: nodal current (**Ampere**)
- Coupling between two solvers may be done either on the continuous or discrete level

```
Solver i
  Calculate Loads = True
```

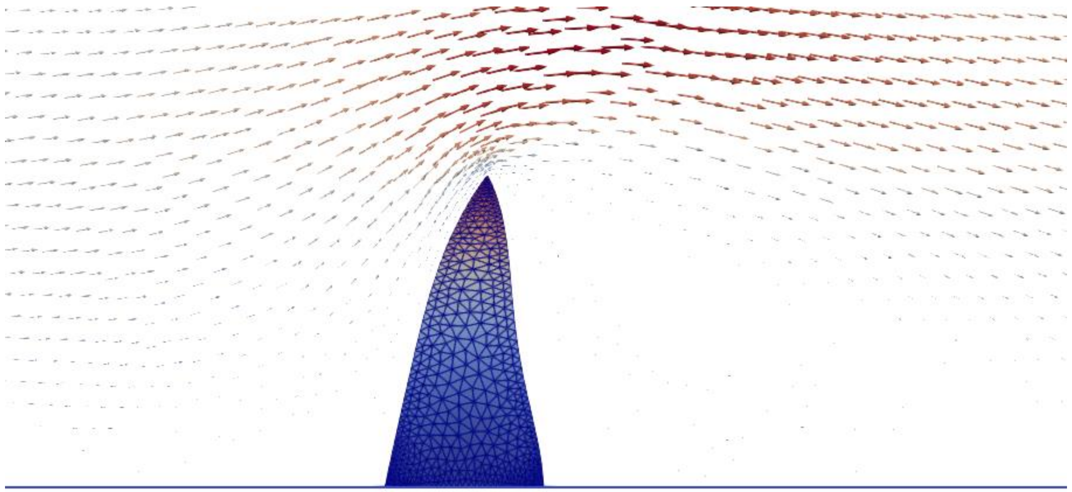
ElmerSolver – Setting nodal forces

- ElmerSolver allows for automatic setting of nodal forces to matrix equation r.h.s.
- The name is derived from the primary variable name
  - **HeatSolver: Temperature Load**
  - **FlowSolve: Flow Solution i Load**
  - **StressSolve: Displacement i Load**
  - **StatElecSolve: Potential Load**
  - …
- Coupling between two solvers may be done either on the continuous or discrete (matrix) level
- Discrete level can often be done without any additional coding and it is at least as accurate!

$$f_{solid}=-f_{fluid}$$

Test case: fsi_beam_nodalforce
Setting FSI conditions on the discrete level

```
Displacement 1 Load = Opposes "Flow Solution Loads 1"
Displacement 2 Load = Opposes "Flow Solution Loads 2"
```



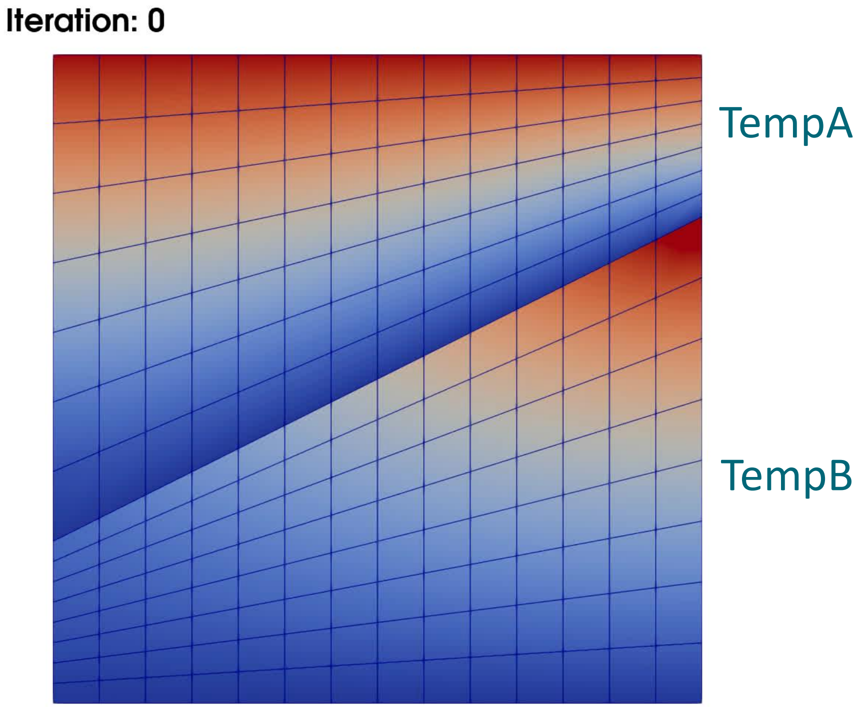
Test case: fsi_beam
Setting FSI internally on continuous level

```
Fsi Bc = True
```

Example: Dirichlet-Neumann Domain Decomposition

tests: DirichletNeumann

- Two equations for temperature: TempA and TempB
- We iterate on convergence solution such that
  - Same temperatures: $T_a = T_b$
  - Same fluxes: $-kdT_a/dn = kdT_b/dn$
- We use library functionalities
  - Dirichlet conditions
  - Computing nodal loads
  - Setting nodal loads
  - Steady state iteration
  - No coding required!

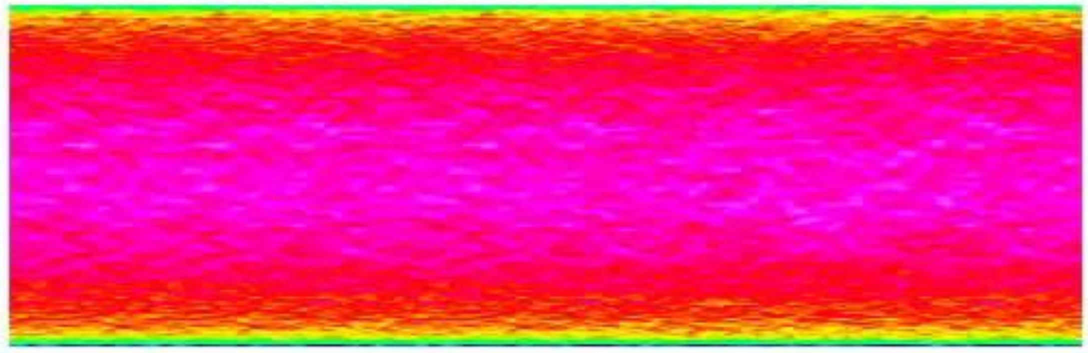



```
Boundary Condition 3
  Name = "Interface"
  Target Boundaries = 3

  TempB = Equals TempA
  TempA Load = Opposes "TempB Loads"
End
```

Periodic Boundary Conditions (node-to-surface)

- BC
  - **Periodic BC** = Integer
    ! Give the corresponding master boundary
  - **Periodic BC Varname** = Logical True
    ! Enforce periodicity for given variable
  - **Periodic BC Offset Varname** = Real
    ! Enforce desired constant offset in values
  - ...
- Creates surface-to-node mapping and keeps the size of the linear system the same
- Accuracy optimal for conforming meshes only
- Accuracy not optiomal for non-conforming meshes
  - Mortar Finite Elements

2D periodic LES simulation using VMS, by Juha Ruokolainen



```
Periodic BC = 2
Periodic BC Pressure = Logical True
Periodic BC Offset Pressure = Real 10.0
Periodic BC Velocity 1 = Logical True
Periodic BC Velocity 2 = Logical True
```

Mortar Boundary conditions (surface-to-surface)

- Solver
  - **Apply Mortar BCs** = Logical
    ! Should the solver apply the conditions?
- BC
  - **Mortar BC** = Integer
    ! Give the corresponding master boundary
  - **Galerkin Projector** = Logical
    !This enforces the weak projector for all dofs
  - **Mortar BC Static** = Logical
    ! Projectors may be assumed to be static
  - ...

- Mapping of accuracy is optimal
  - Adds lagrange multipliers to the system
  - Convergence of linear system becomes more challenging
- Provides a framework for many complicated problems
  - Rotating boundary conditions
  - Contact mechanics
  - Symmetric & nonconforming
- Also antiperiodic systems supported

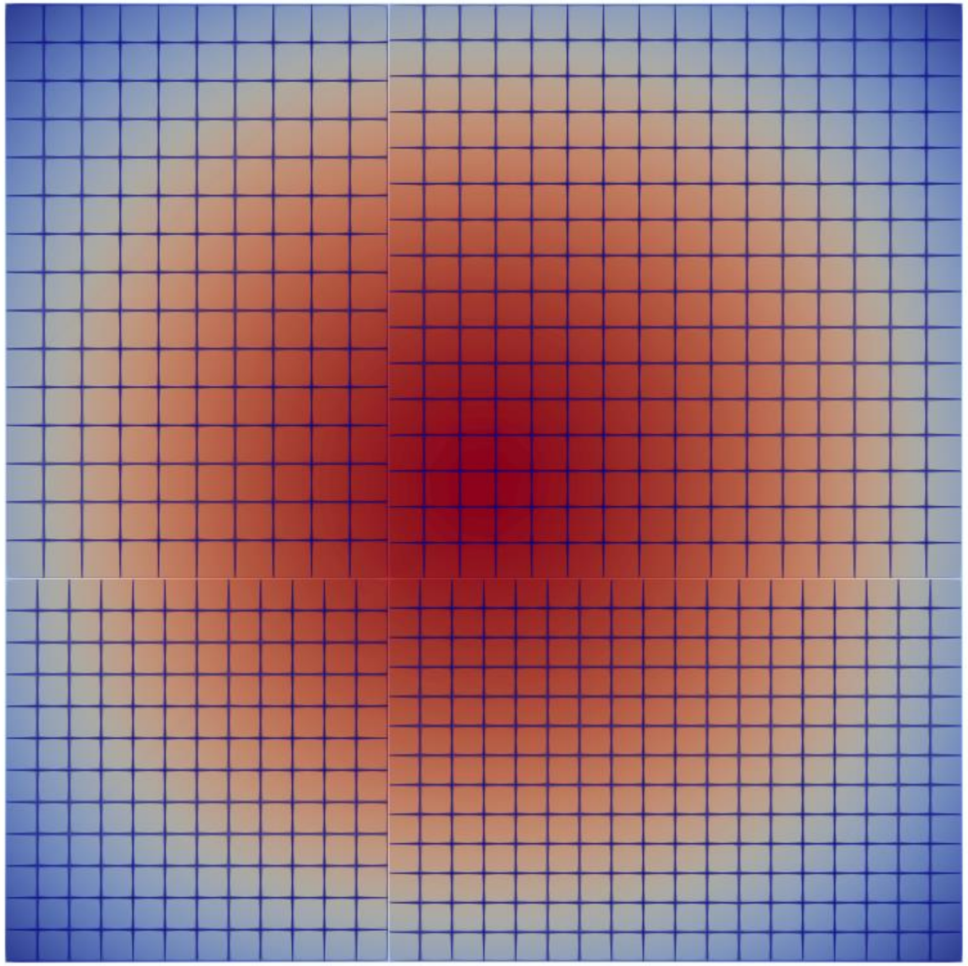
Periodic conditions (strong): $x_l = P x_r$
Mortar conditions (weak): $Q x_l - R x_r = 0$

Example: continuity with mortar projector in 2D

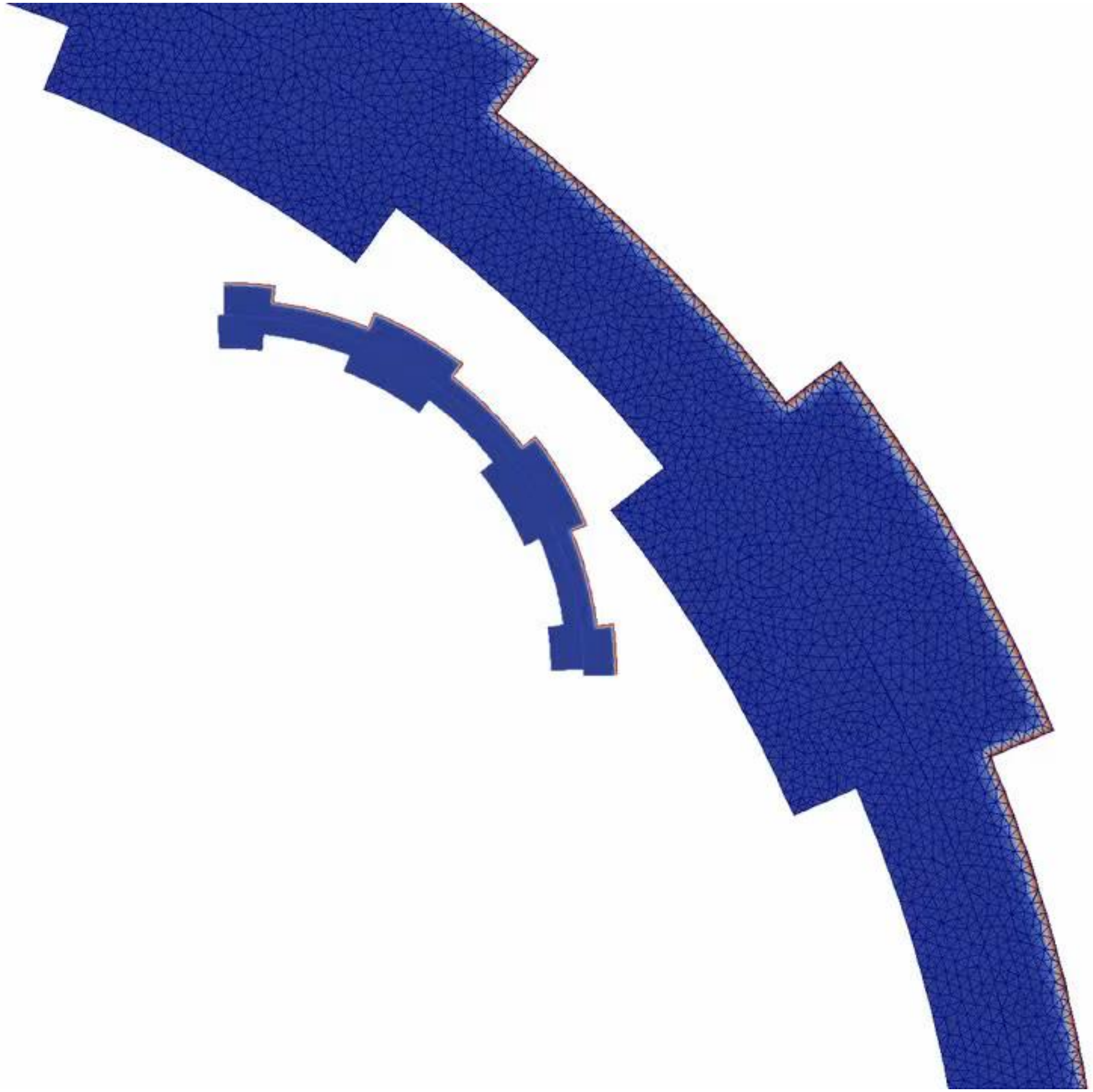
- Multiple mortar BCs possible at the same time
  - Weights may be summed up

```
Boundary Condition 6
  Target Boundaries(1) = 7
  Name = "Mortar Left Master"
  Mortar BC = Integer 7
  Galerkin Projector = Logical True
  Plane Projector = Logical True
End

Boundary Condition 7
  Target Boundaries(1) = 6
  Name = "Mortar Left Target"
End

….
```

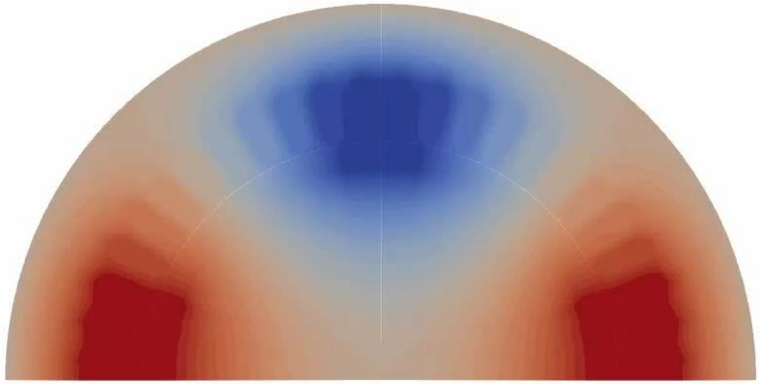
test case: MortarPoisson2Dsum

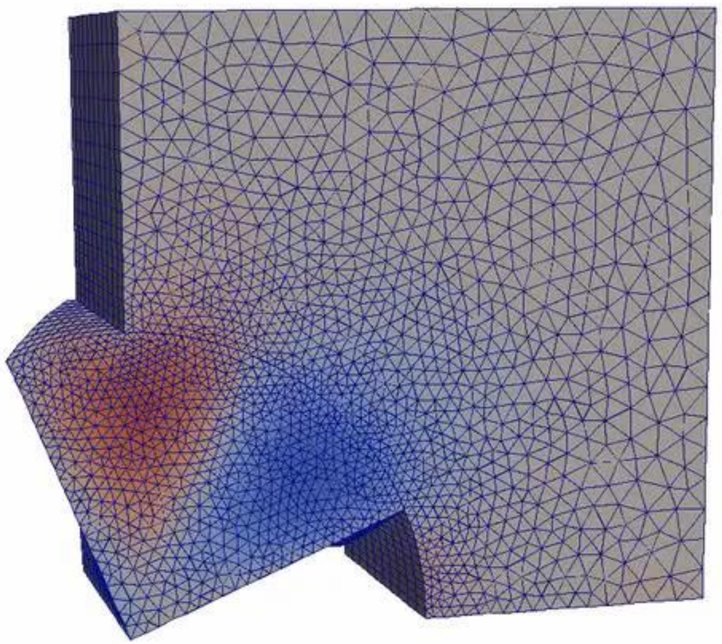


Example: toy model for temperature between 2D rotor and stator



Example: Rotating 2D and 3D "machines"





Comparison of shared boundary conditions in Elmer

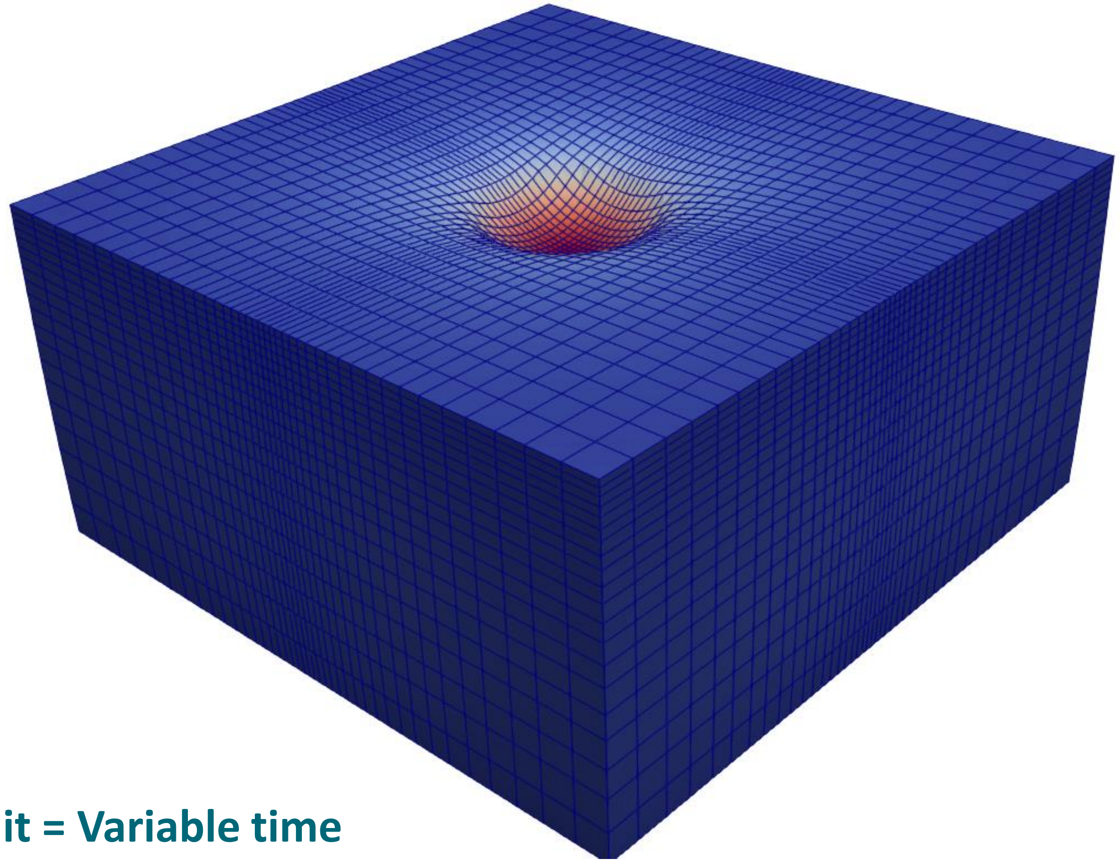
BC type	Mortar BC	Periodic BC	Conforming BC
also known as	surface-to-surface	node-to-surface	elimination
Non-confoming BCs	YES	YES	NO
Matrix size	N+M	N	N-M
Spoils the matrix	YES	yes	NO
Edges possible	YES	NO	YES

Soft Limiters in Elmer (Inequality Constraints)

- General way to ensure min/max limit of solution
- A priori contact surface => soft limiters
- Uses two consepts
  - Nodal load evaluation
  - Contact set
- Applicable to
  - Heat equation
  - Elasticity
  - Richards equation
  - ...

test case: LimitDisplacement2

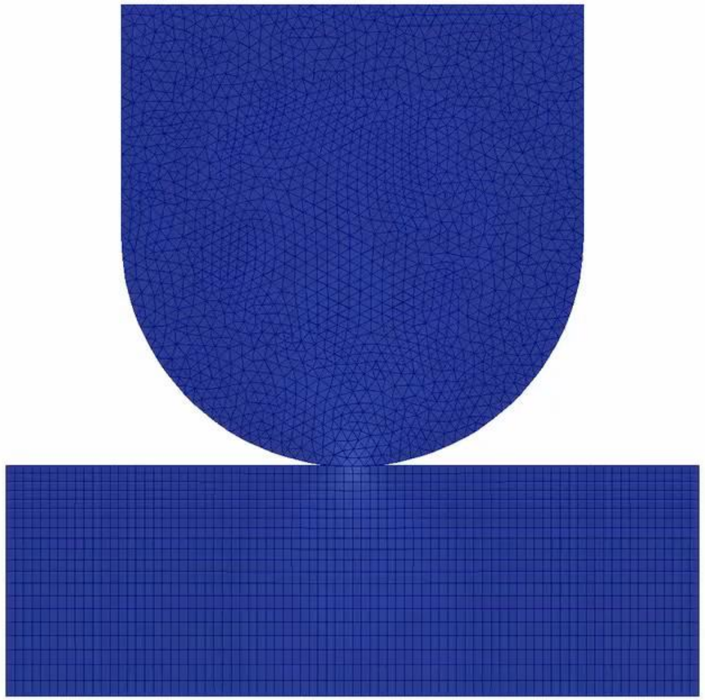
```
Solver 1
 Apply Limiter = True
 ...

Boundary Condition 2
  Name = "Contact"
  Target Boundaries(1) = 6

  Displacement 3 Upper Limit = Variable time
    Real Procedure "ContactBC" "SphereBottom"
End
```



Hertz problem

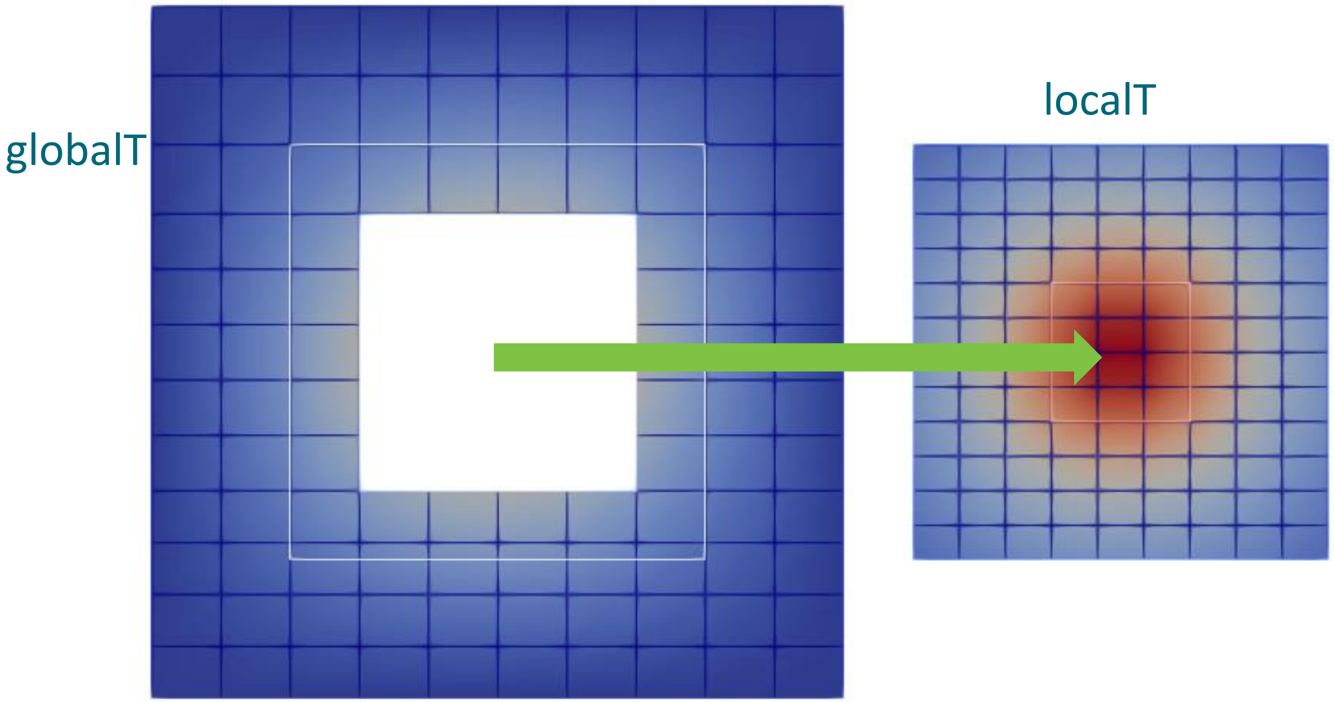
Contact mechanics in Elmer

- Utilizes the optimal mortar methods + contact sets of soft limiters
- Results to difficult linear systems
  - Elimination by using dual basis test functions
- Some challenges in general cases i.e. with conflicting normals



Consistency between meshes (as opposed to boundaries)

- Consistency between different meshes may only be enforced explicitely
- Mapping is done automatically when variables is needed
    - Octree search – optimal in speed



Triggers mesh-to-mesh interpolation

test case: multimesh

```
Boundary Condition 2
  Name = "Local2Global"
  Target Boundaries = 2
  globalT = Equals localT
End

Boundary Condition 3
  Name = "Global2Local"
  Target Boundaries = 3
  localT = Equals globalT
End
```

Summary

- ~100 Solvers that try to do some specific tasks
  - ElmerModels Manual
- ~200,000 lines of code in the library providing a wide variety of services
  - ElmerSolver Manual
- Many undocumented features still exist!

Discussion: where to go from here?

- Current focus in Elmer/Ice and electromechanics + some smaller projects
- Architectures change as a driver
  - Threading and GPU developments
  - Needed to take use of supercomputers
- Open source ecosystem
  - Focus to where there software shines
  - Take use of other tools when suitable -> interfaces
- Comments?

Most important Elmer resources

- http://www.csc.fi/elmer
  - Official Homepage of Elmer at CSC
- http://www.elmerfem.org
  - Discussion forum, wiki, elmerice community
- https://github.com/elmercsc/elmerfem
  - GIT version control
- http://youtube.com/elmerfem
  - Youtube channel for Elmer animations
- http://www.nic.funet.fi/pub/sci/physics/elmer/
  - Download repository
- Further information: elmeradm@csc.fi

Thank you for your attention!