PARA2012 – Elmer Tutorial

Elmer

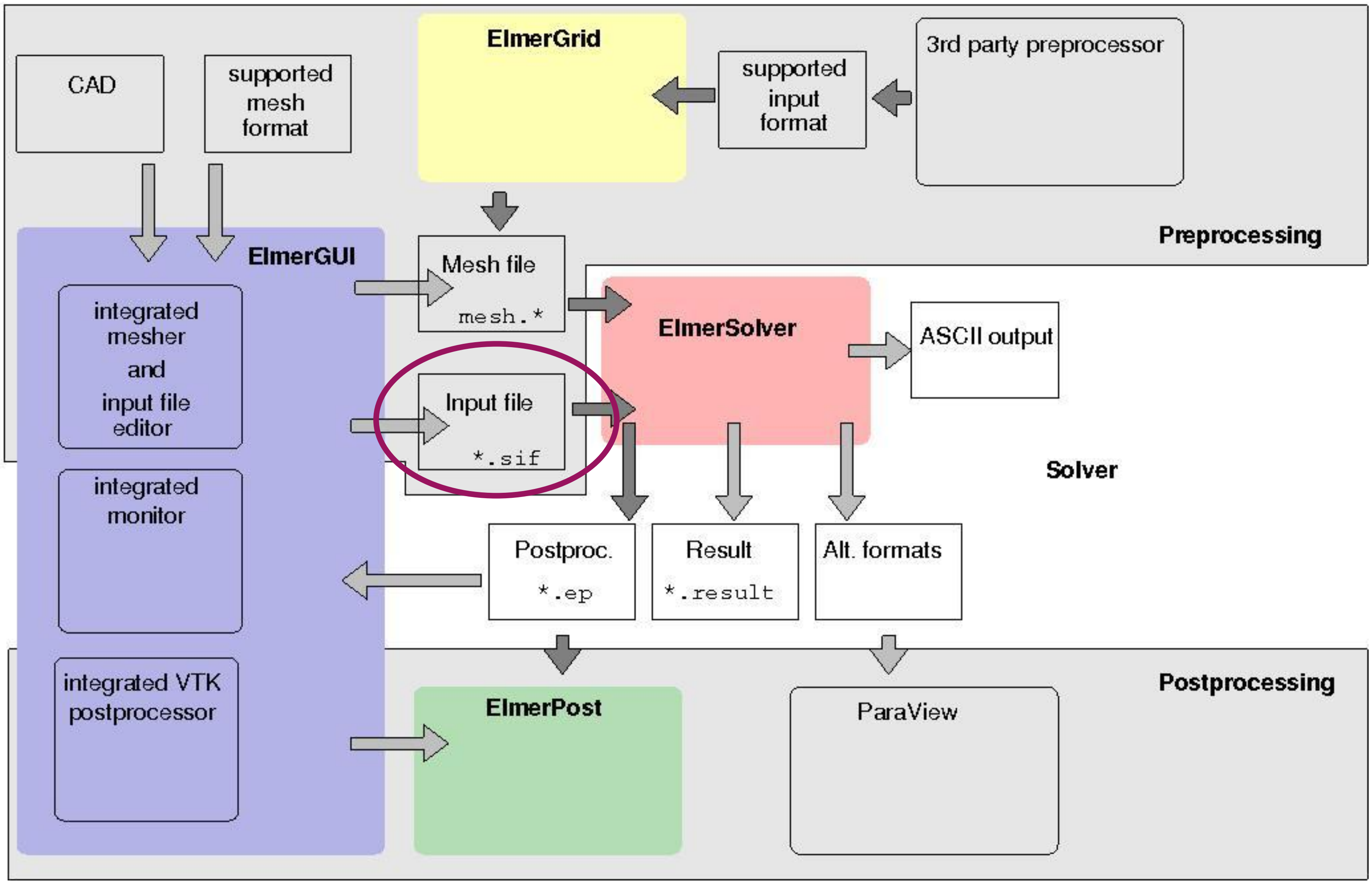
Elmer Solver Input File -
Almost all about the text interface to ElmerSolver

Elmer Team
CSC – IT Center for Science Ltd.

Contents

- Elmer Modules
- From PDE to SIF
- Syntax of SIF
  - Parameters, etc.
- Sections of SIF:
  - Header
  - Constants
  - Simulation
  - Solver
  - Body
  - Equation
  - Body Force
  - Material
  - Initial Condition
  - Boundary Condition
- Tables and Arrays
- MATC
- User Defined Functions

Elmer - Modules



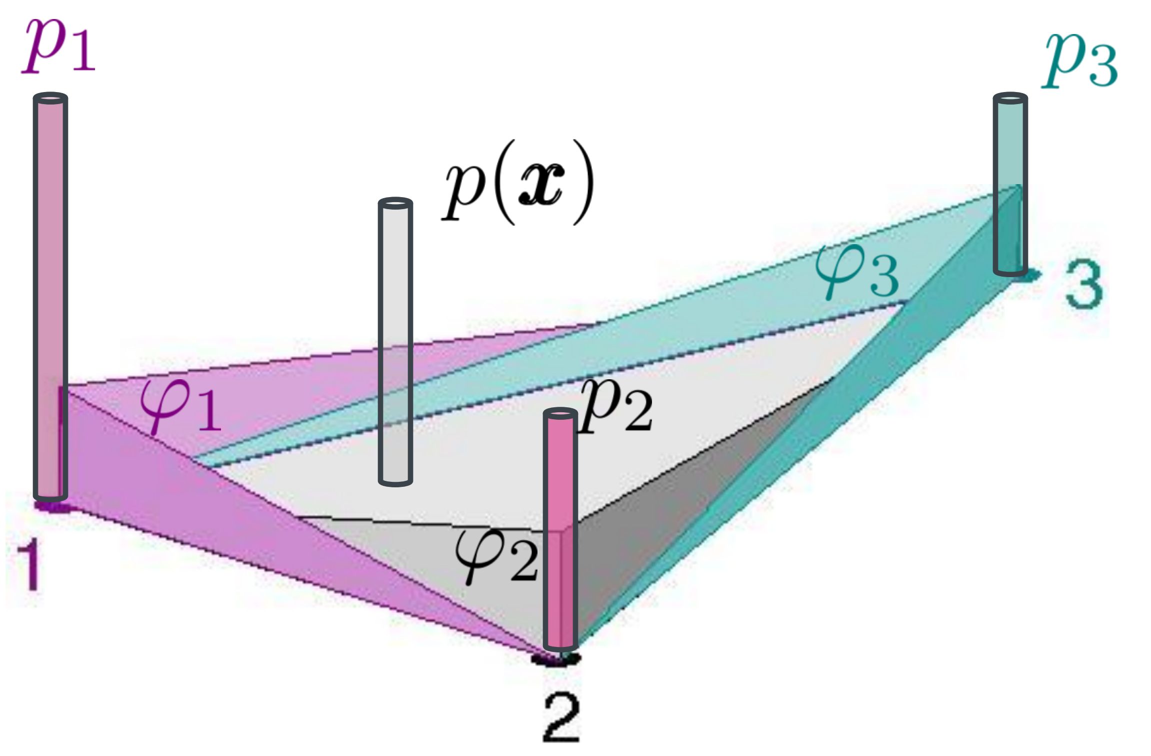
From the PDE to the SIF

- General advection-diffusion problem

$$c\varrho\left(\frac{\partial\Psi}{\partial t} + \mathbf{u}\nabla\Psi\right) = \nabla\cdot\underbrace{(-\kappa\nabla\Psi)}_{\mathbf{q}} + \varrho\sigma$$

  - For instance, heat transfer problem: $\Psi = T$
  - Coupled to (Navier-)Stokes via velocity: $\mathbf{u}$
  - Non-linearities via material parameters, e.g., $c = c(\Psi)$

From the PDE to the SIF



$$p(\boldsymbol{x}) = p_1\varphi_1|_{\boldsymbol{x}} + p_2\varphi_2|_{\boldsymbol{x}} + p_3\varphi_3|_{\boldsymbol{x}}$$

From the PDE to the SIF

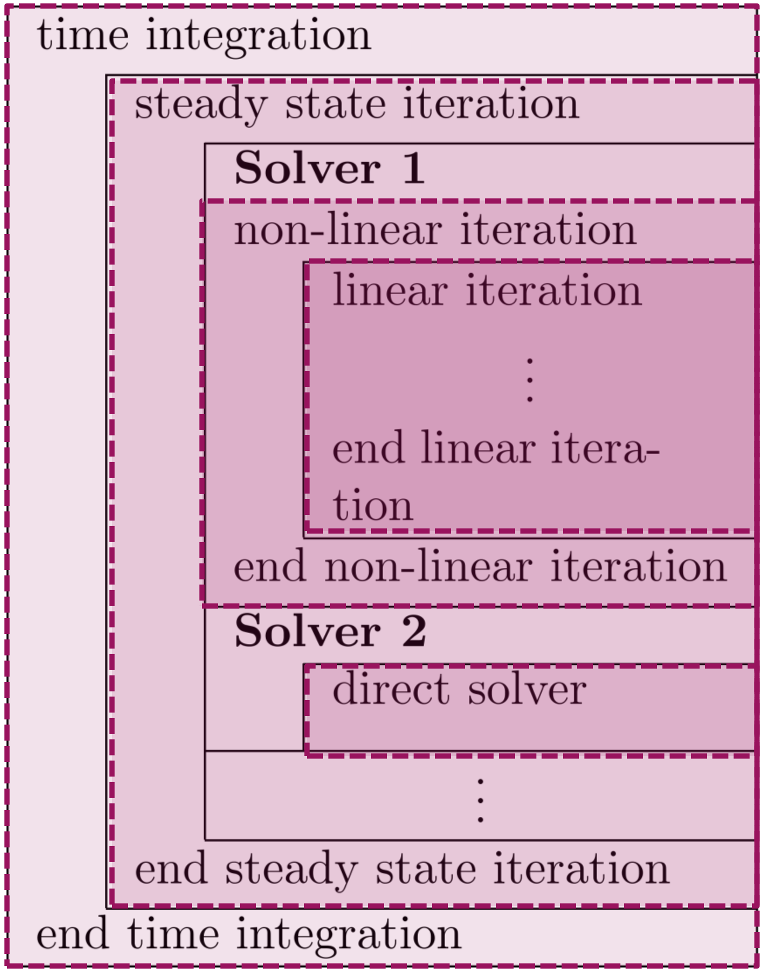
- Weak formulation:

$$\underbrace{\Psi_\beta\, a(\Delta t) \int\limits_\Omega \varrho c \varphi_\beta \varphi_\alpha d\Omega}_{\mathbf{M}} + \underbrace{\Psi_\beta \int\limits_\Omega \left[\varrho c \mathbf{u} \cdot \nabla \varphi_\beta \varphi_\alpha + \kappa \nabla \varphi_\beta \cdot \nabla \varphi_\alpha\right] d\Omega}_{\mathbf{S}} =$$

$$\underbrace{\oint\limits_{\partial\Omega} (\mathbf{q}\varphi_\alpha) \cdot \mathbf{n} d\Omega}_{\text{nat. BC}} + \underbrace{\int\limits_\Omega \varrho \sigma \varphi_\alpha d\Omega}_{\mathbf{f}}$$

  - Time integration
  - Steady State:dependence on other variables
  - Non-linear iteration: internal dependence

From the PDE to the SIF



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.

Syntax of SIF

- Do not use non-printable characters!
  - No Tabulators, etc.
- A comment is preceded by a **`!`**
- Parameters in general have to be casted by their type:
  - **`Real`**, **`Integer`**, **`Logical`**, **`String`**, **`File`**
  - Exception: entry in Keyword DB
- Arrays have to be declared with the name:

```
Array(4) = Real 1.0 2.0 3.0 4.0
```

Sections of SIF

- The SIF is structured into sections
  - **`Header`**
  - **`Constants`**
  - **`Simulation`**
  - **`Solver`**
  - **`Body`**
  - **`Equation`**
  - **`Body Force`**
  - **`Material`**
  - **`Initial Condition`**
  - **`Boundary Condition`**

The contents of each section is between the keyword above and an **`End`**-statement

Sections of SIF: Header

- Declares search paths for mesh

```
Header
 Mesh DB "." "dirname"
End
```

- preceding path + directory name of mesh database
- Replace path and ***dirname*** to fit your case

Sections of SIF: Constants

- Declares simulation-wide constants

```
Constants
  Gas Constant = Real 8.314E00
  Gravity (4) = 0 -1 0 9.81
End
```

- a casted scalar constant
- Gravity vector, an array with a registered name

Sections of SIF: Simulation

- Declares details of the simulation:

```
Simulation
  Coordinate System = "Cartesian 2D"
```

- choices: **Cartesian{1D,2D,3D}, Polar{2D,3D}, Cylindric, Cylindric Symmetric, Axi Symmetric**

```
  Coordinate Mapping(3) = Integer 1 2 3
```

- Permute, if you want to interchange directions in mesh

```
  Simulation Type ="Transient"
```

- **Steady State** or **Transient**

```
  Output Intervals(2) = 10 1
```

- Interval of results being written to disk

Sections of SIF: Simulation

- Declares details of the simulation:

```
Steady State Max Iterations = 10
Steady State Min Iterations = 2


Timestepping Method = "BDF"

Timestep Intervals(2) = 10 100

Timestep Sizes(2) =  0.1 1.0

Output File = "name.result"
Post File = "name.ep"
```

- How many min/max rounds on one timelevel/in a steady state simulation
- Choices: **BDF** or **Crank-Nicholson**
- Has to match array dimension of **Timestep Sizes**
- The length of one time step
- Contains data for restarting
- Contains ElmerPost data

Sections of SIF: Simulation

- Declares details of the simulation:

```
  Restart File = "previous.result"
  Restart Position = 10
  Restart Time = 100

  Initialize Dirichlet Condition =
   False

  Restart Before Initial Conditions =
   True

  Max Output Level = 3
End
```

- Restart from this file at file-entry (not necessarily timestep!) no. 10 and set time to 100 time-units
- Default is True. If false, Dirichlet conditions are called at Solver execution and not at beginning
- Default is False. If True, then Initial Condition can overwrite previous results
- Level of verbosity. 1 = basics, 9 = all and everything

Sections of SIF: Solver

- Declares a physical model to be solved

```
Solver 3
 Equation = "Navier-Stokes"
 Exec Solver = "Always"

 Linear System Solver = "Iterative"
 Linear System Iterative Method =
   "BiCGStab"
 Linear System Convergence Tolerance
   = 1.0E-06
Linear System Abort Not Converged =
   True
Linear System Preconditioning =
   "ILU2"
```

- Numbering from 1 (priority)
- The name of the equation
- **Before/After Simulation/Timestep**
- Choices: **Iterative** or **Direct**
- Lots of choices here
- Convergence criterion
- If not True (default) continues simulation in any case
- Lots of choices

Sections of SIF: Solver

- Solving the **Linear(ized) Problem**

$$\mathbf{A}\,\mathbf{\Psi} = \mathbf{f}$$

- Keyword:

```
Linear System Solver
```

- 3 ways to do that in Elmer:
  1. `Direct` methods (= inversion of $\mathbf{A}$)
  2. `Iterative` methods (=working with approximations to $\mathbf{A}$)
  3. Multi-grid methods (not discussed here)

Sections of SIF: Solver

- **Direct linear system solver**
- Keyword:

`Linear System Direct Method`

  - `Banded` (default) LAPack
  - `UMFPACK` Unsymmetric MultiFrontal method (only serial)
  - `MUMPS` Unsymmetric MultiFrontal method (only parallel)

– Sometimes the only way to go (if bad conditioned)

– Costly: Elimination takes $\sim N^3$ operations and needs to store N^2 unknows in memory

Sections of SIF: Solver

- **Iterative solvers**:
  - Krylov subspace: $\mathbf{x}_n \in \operatorname{span}(K_n)$

$$K_n = [\mathbf{f}, \mathbf{A}\mathbf{f}, \mathbf{A}^2\mathbf{f}, \mathbf{A}^3\mathbf{f}, \ldots, \mathbf{A}^{n-1}\mathbf{f}]$$

$$\mathbf{R} = (\mathbf{f} - \mathbf{A}\,\mathbf{x}_n) \rightarrow \min.$$

  - `Linear System Iterative Method`
    - `GMRES` Generalized Minimal Residual Method
    - `CG`, `CGS`, `BiCGStab` Conjugate Gradient
    - `TFQMR` Transpose-free quasi-minimal residual
    - `GCR` Generalized Conjugate Residual

Section of SIF: Solver

- But, before we solve, we usually apply a

pre-conditioner

  – Find **P~A**, but much easier to invert
  – $\mathbf{P}^{-1}\mathbf{A} \sim \mathbf{I}$ has favourable condition number

$$\mathbf{P}^{-1}\mathbf{A}\,\boldsymbol{\Psi} = \mathbf{P}^{-1}\mathbf{f}$$

`Linear System Preconditioning`

- `None`
- `Diagonal`
- `ILU`*n* (*n*=0,1,2,...) and `ILUT`

Sections of SIF: Solver

- Declares a physical model to be solved

```
Nonlinear System Convergence
  Tolerance = 1.0E-05
Nonlinear System Max Iterations = 20
Nonlinear System Min Iterations = 1
Nonlinear System Newton After
  Iterations = 10
Nonlinear System Newton After
  Tolerance = 1.0e-03

Steady State Convergence Tolerance =
  1.0E-03
Stabilization Method = Stabilized
End
```

- Convergence criterion for non-linear problem
- The maximum rounds
- The minimum rounds
- Switch from Picard to Newton scheme after 10 iterations ...
- ... or after this criterion (NV.: has to be smaller than convergence criterion ot hit)
- The convergence on the time-level
- Convection needs stabilization. Alternatives: **Bubbles** or **P2/P1**

Sections of SIF: Body

- Declares a physical model to be solved

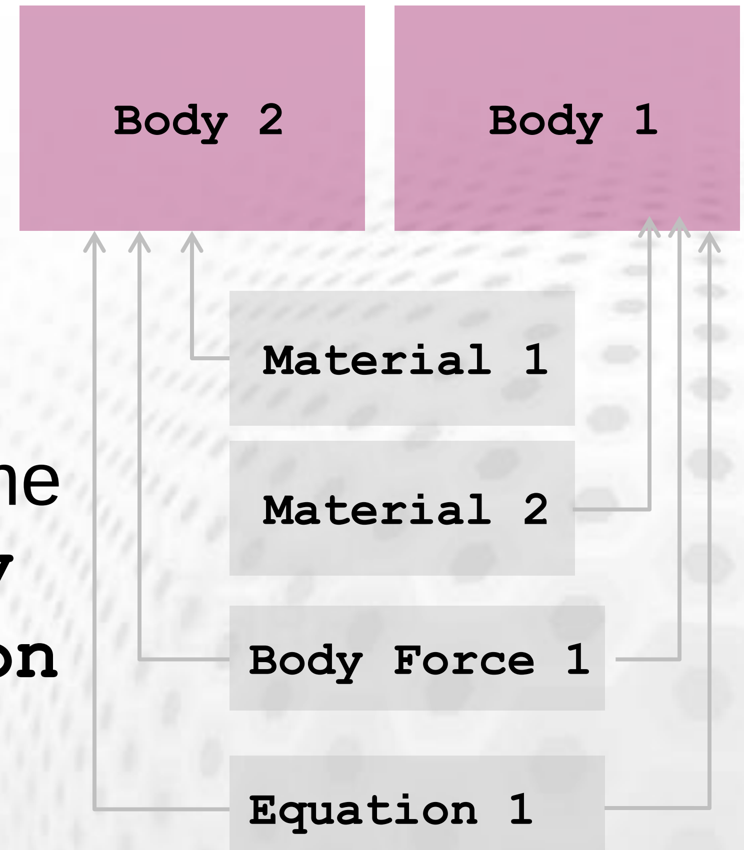
```
Body 2

 Name = "pipe"
 Equation = 2
 Material = 2
 Body Force = 1
 Initial Condition = 2
End
```

- Numbering from 1 to number of bodies
- Identifier of the body
- The assigned set of equations
- The assigned material section
- The assigned body force
- The assigned initial condition

Sections of SIF: Body

- Each **Body** has to have an **Equation** and **Material** assigned
  - **Body Force**, **Initial Condition** optional
- Two bodies can have the same **Material/Equation/Body Force/Initial Condition** section assigned



Sections of SIF: Equation

- Declares set of solvers for a body

```
Equation 2

 Active Solvers(2) = 1 3

Convection = Computed



 NS Convect = False



End
```

- Numbering from 1 to number of equation sets
- Declares the solvers (according to their numbers) to be solved within this set
- Important switch to account for convection term. Alternatives: None and Constant (needs Convection Velocity to be declared in the Material section)
- Sets no convection for Navier-Stokes (=Stokes) alternative:

  `Flow Model = Stokes`

  in the Solver section of Navier-Stokes

Sections of SIF: Body Force

- Declares body forces and bulk and execution conditions for a body

```
Body Force 3

 Flow Body Force 1 = 0.0
 Flow Body Force 2 = -9.81

 MyVariable = Real 0.0

 Flow Solution Passive = 1.0
End
```

- Numbering from 1 to number of body forces
- Gravity pointing in negative x-direction applied to Navier-Stokes solver
- A Dirichlet condition for a variable set within the body
- Suspends execution of Navier-Stokes solver (-1.0 = default = execution)

Body forces can be functions of other variables

Sections of SIF: Material

- Declares set of material parameters for body

```
Material 1

 Density = 1000.0

Heat Conductivity(3,3) = 1 0 0\
                          0 1 0\
                          0 0 2
 Viscosity = Variable Temperature
    Real MATC "viscosity(tx)"

 MyMaterialParameter = Real 0.0

End
```

- Numbering from 1 to number of material
- Always declare a density (mandatory)
- Parameters can be arrays
- Or functions of other variables
- Non-keyword DB parameters have to be casted

Sections of SIF: Initial Condition

Declares initial conditions for a body

By default restart values are used

```
Initial Condition 2

 Velocity 1 = Variable Coordinate 2
    Real MATC "42.0*(1.0 - tx/100.0)"
 Velocity 2 = 0.0

  MyVariable = Real 20.0

End
```

- Numbering from 1 to number of IC's
- Initial condition as a function of a variable ...
- ... and as a constant

- Non-keyword DB parameters have to be casted

Sections of SIF: Boundary Condition

- Declares conditions at certain boundaries

```
Boundary Condition 3

 Target Boundaries(2) = 1 4


 Velocity 1 = Variable Coordinate 2
    Real MATC ″42.0*(1.0 – tx/100.0)″
 Velocity 2 = 0.0

 Normal-Tangential Velocity = Logical True

End
```

- Numbering from 1 to number of BC's
- The boundaries of the mesh the BC is assigned to
- Variable as a function and ...

  ... as a constant

- Set velocities in normal-tangential system

Tables and Arrays

- Tables (piecewise linear or cubic):

```
Density = Variable Temperature
Real cubic
     0 900
   273 1000
   300 1020
   400 1000
End
```

- Arrays:

```
Target Boundaries(3) = 5 7 10

MyParamterArray(3,2) = Real 1 2\
                            3 4\
                            5 6
```

- Expresions:

```
OneThird = Real $1.0/3.0
```

MATC

- Syntax close to C
- Even if-conditions and loops
- Can be use for on-the-fly functions inside the SIF
- Documentation on web-pages
- Do not use with simple numeric expressions: is much faster than

```
OneThird = Real $1.0/3.0
```

```
OneThird = Real MATC “1.0/3.0”
```

MATC

- Use directly in section:

```
Heat Capacity = Variable Temperature
 Real MATC "2.1275D03 + 7.253D00*(tx - 273.16)"
```

- Even with more than one dependency:

```
Temp = Variable Latitude, Coordinate 3
 Real MATC "49.13 + 273.16 - 0.7576*tx(0) - 7.992E-03*tx”
```

- Or declare functions (somewhere in SIF, outside a section)

```
$ function stemp(X) {\
  stemp = 49.13 + 273.16 - 0.7576*X(0) - 7.992E-03*X(1) }
```

  – being called:

```
Temp = Variable Latitude, Coordinate 3
   Real MATC “_stemp(tx)”
```

User Defined Functions (UDF)

- Written in Fortran 90
- Dynamically linked to Elmer
- Faster, if more complicated computations involved
- Compilation command **elmerf90**

```
elmerf90 myUDF.f90 -o myUDF.f90
```

- Call from within section:

```
MyVariable = Variable Temperature
  Real Procedure "myUDF" "myRoutine"
```

User Defined Functions (UDF)

- Example: $\rho(T[K]) = 1000.0 \cdot \left[1 - 1 \times 10^{-4} \cdot (T - 273.15)\right]$

```
FUNCTION getdensity( Model, N, T ) RESULT(dens)
 USE DefUtils !important definitions
 IMPLICIT None
 TYPE(Model_t) :: Model
 INTEGER :: N
 REAL(KIND=dp) :: T, dens
 dens = 1000.0_dp*(1.0_dp - 1.0d-04*(T - 273.0_dp))
END FUNCTION getdensity
```

  – Definitions loaded from **`DefUtils`**
  – Header: **`Model`** access-point to all ElmerSolver inside data; Node number **`N`**; input value **`T`**