Solution Strategies with Elmer

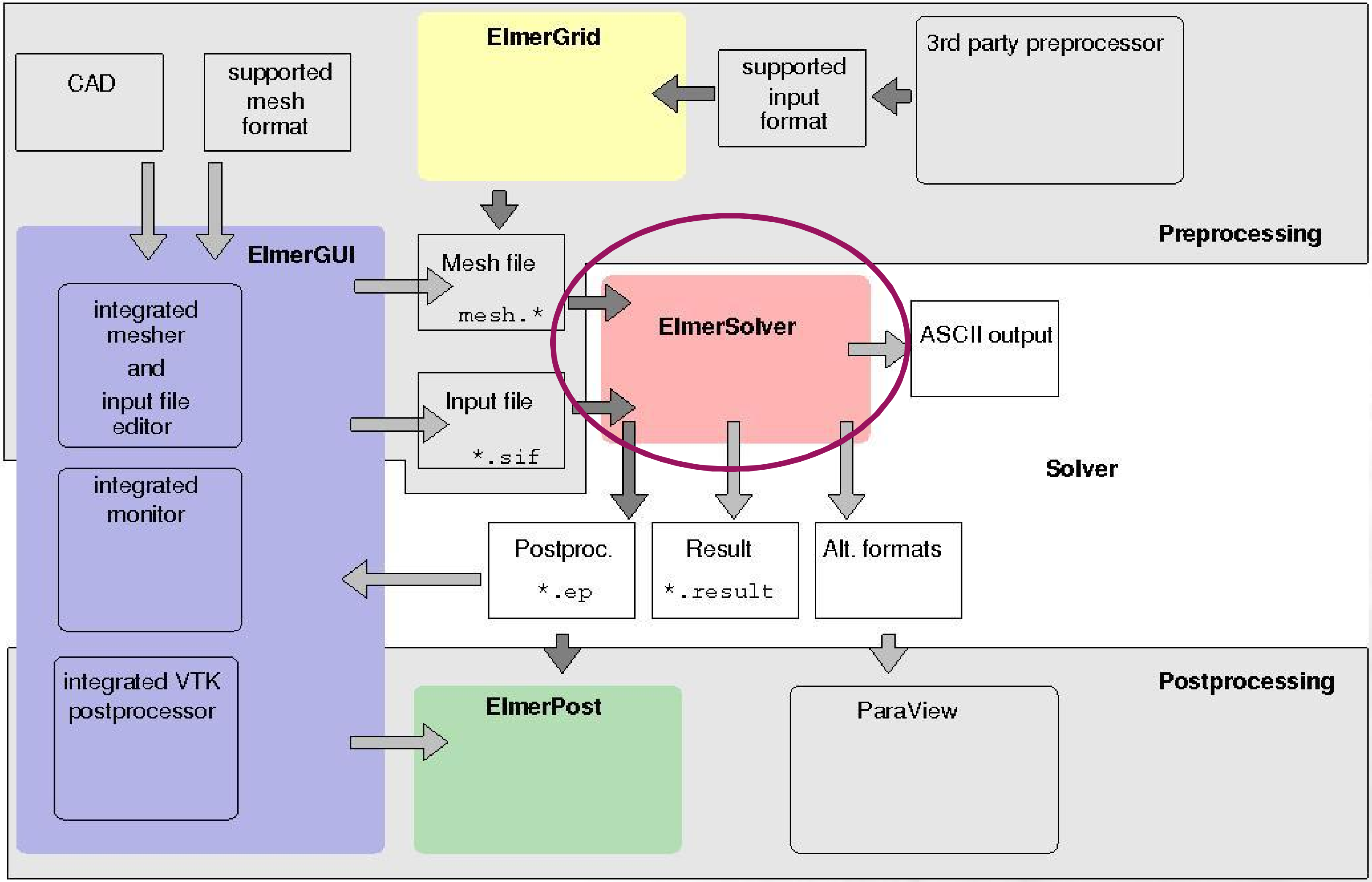
Solution Strategies with Elmer

Elmer Team
CSC – IT Center for Science Ltd. Elmer

Contents

- Elmer – Modules
  - Workflow
- From the PDE to the SIF
  - Discretization
  - Linearization
  - Different levels of iteration
- Time Integration
- Steady State Iteration
- Non-linear Iteration
- Pre-conditioner
- Linear(ized) Problem
  - Direct Solver
  - Iterative Solver
- On Bodies and Boundaries

Elmer - Modules



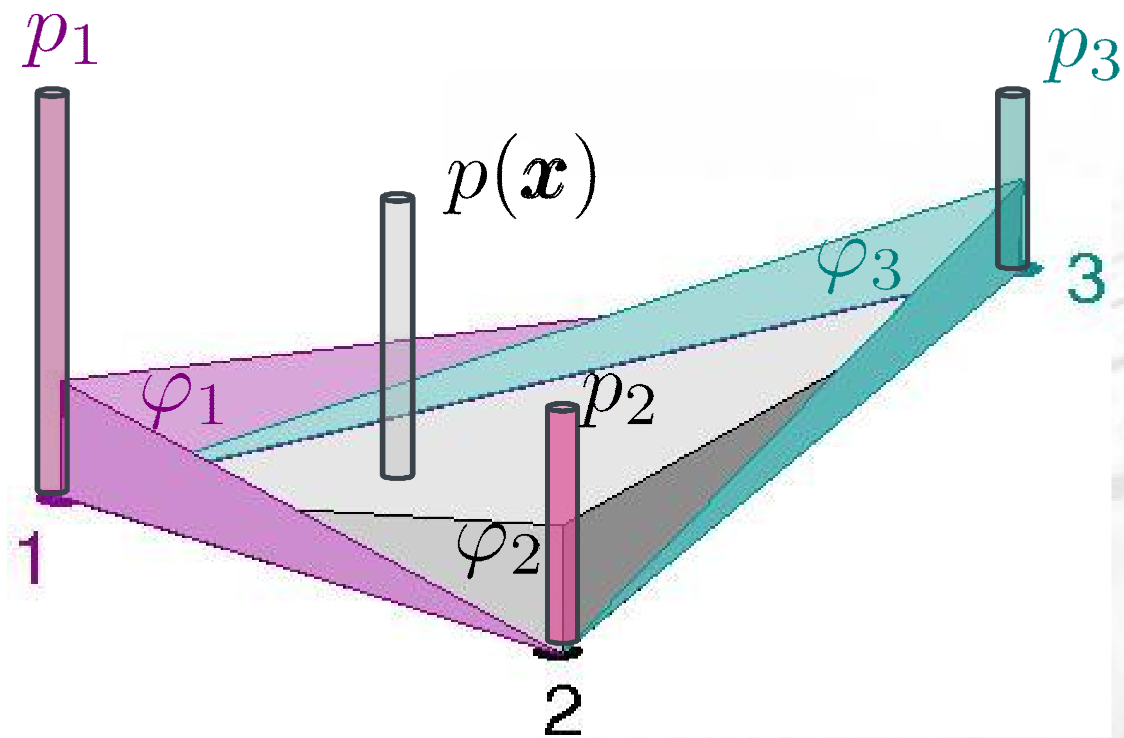
From the PDE to the SIF

- General advection-diffusion problem

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

  - For instance, heat transfer problem: $\Psi = T$
  - Coupled to (Navier-)Stokes via velocity: **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:

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

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

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

From the PDE to the SIF

- Weak formulation:

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

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

  - System matrix: $\mathbf{M}+\mathbf{S}=\mathbf{A} \quad \Rightarrow \quad \mathbf{A}\,\mathbf{\Psi}=\mathbf{f}$
  - Non-linearities: e.g. $c(\Psi)\Psi\big|^{(i)} \Rightarrow c(\Psi^{(i-1)})\Psi^{(i)}$
  - Natural BC: either to **A** or **f**

From the PDE to the SIF

time integration	1. `Timestep Intervals`
steady state iteration	2. `Steady State Max Iterations`
Solver 1	
non-linear iteration	3. `Nonlinear Max Iterations`
linear iteration	4. `Linear System Max Iterations`
⋮	
end linear iteration	4. `Linear System Convergence Tolerance`
end non-linear iteration	3. `Nonlinear System Convergence Tolerance`
Solver 2	
direct solver	
⋮	
end steady state iteration	2. `Steady State Convergence Tolerance`
end time integration	1.

Time Integration

- Two different methods:
  - – Crank – Nicholson: `Crank - Nicholson`
  - – Backward Difference Formula: `BDF`
    - `BDF Order` (if 1, then backward Euler – only choice for adaptive time-stepping)
- Additional settings:
  - – `Time Derivative Order` (if 2 then Bossak)
  - – `Timestep Intervals`
  - – `Timestep Sizes`

Steady State Iteration

- Mutual dependence between Solvers
  - (e.g., Flow solution and convected temperature)
  - `Steady State Convergence Tolerance`

$$||\mathbf{\Psi}^{(j)} - \mathbf{\Psi}^{(j-1)}||/||\mathbf{\Psi}^{(j)}|| < \epsilon_{\mathrm{st}}$$

  - `Steady State Max Iterations` $j < j_{\mathrm{max}}.$
  - `Steady State Relaxation Factor`

$$\lambda_{\mathrm{st}} : \quad \mathbf{\Psi}^{(j)} \rightarrow \lambda_{\mathrm{st}}\, \mathbf{\Psi}^{(j)} + (1 - \lambda_{\mathrm{st}})\, \mathbf{\Psi}^{(j-1)}$$

Nonlinear Solver Iteration

- **Nonlinear Problem**:

$$\underbrace{\mathbf{A}(\mathbf{\Psi})}_{\mathbf{M}+\mathbf{S}}\,\mathbf{\Psi} = \mathbf{f}(\mathbf{\Psi}) \quad \Rightarrow \quad \mathbf{A}(\mathbf{\Psi}^{(i-1)})\,\mathbf{\Psi}^{(i)} = \mathbf{f}(\mathbf{\Psi}^{(i-1)})$$

– `Nonlinear System Convergence Tolerance`

$$||\mathbf{\Psi}^{(i)} - \mathbf{\Psi}^{(i-1)}||/||\mathbf{\Psi}^{(i)}|| < \epsilon_{\mathrm{nl}}$$

– `Nonlinear System Max Iterations` $i \leq i_{\max}$

– `Nonlinear System Relaxation Factor`

$$\lambda_{\mathrm{nl}}: \quad \mathbf{\Psi}^{(i)} \to \lambda_{\mathrm{nl}}\,\mathbf{\Psi}^{(i)} + (1-\lambda_{\mathrm{nl}})\,\mathbf{\Psi}^{(i-1)}$$

Pre-conditioner

- But, before we solve, we usually apply a **pre-conditioner**
  - Find $\mathbf{P} \sim \mathbf{A}$, 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`

- Large potential for improving scalability!

Linear(ized) Problem

- 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 (built-in or via externla libraries)

Linear(ized) Problem

- **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

Linear(ized) Problem

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

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

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

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