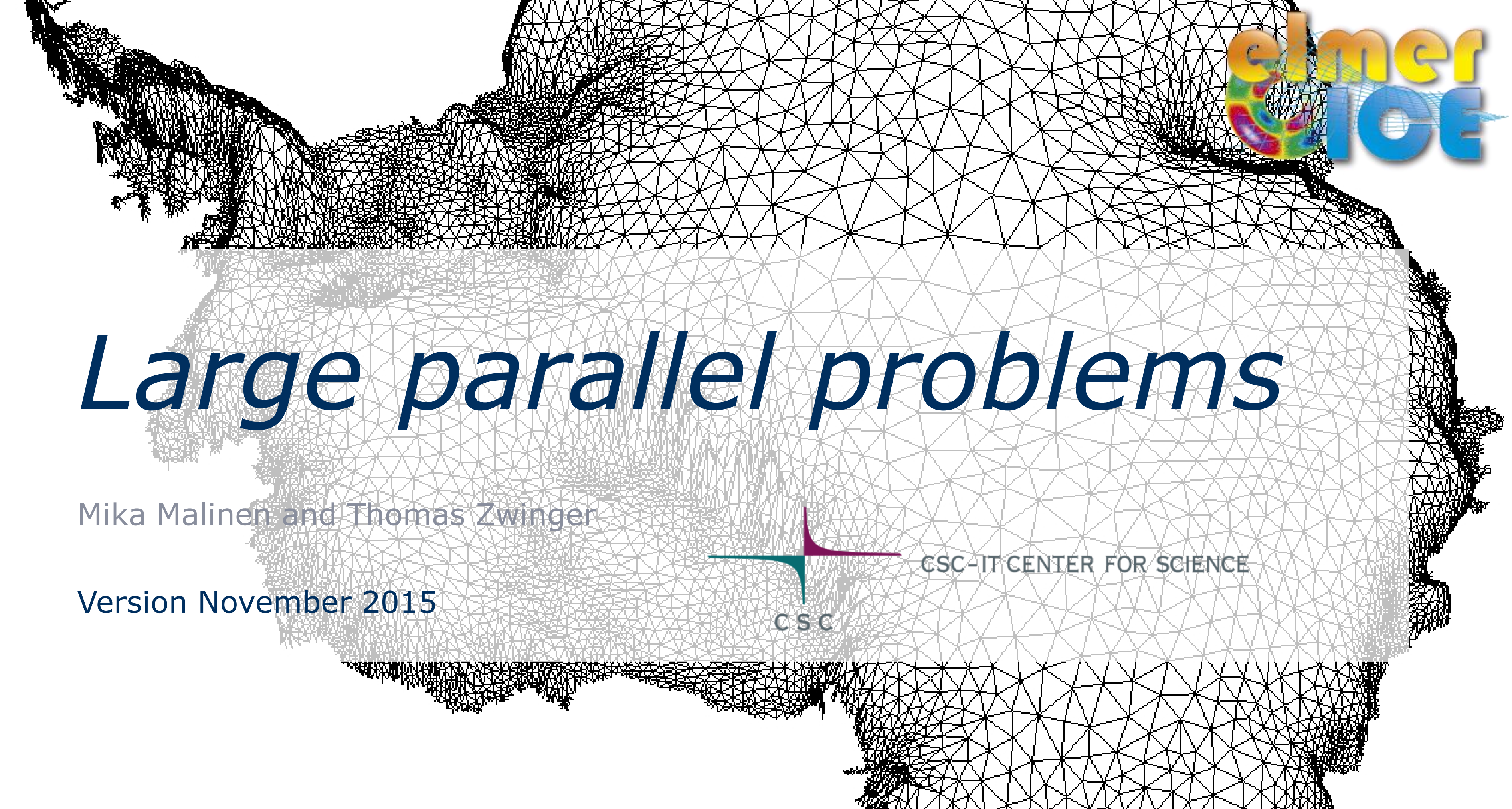


# *Large parallel problems*

Mika Malinen and Thomas Zwinger

Version November 2015

CSC–IT CENTER FOR SCIENCE

# *Background*

- **Motivation**: improving the ability of Elmer solver to handle large discrete partial differential equation models
- The **bottleneck** is typically associated with the performance of iterative solvers for linear systems $\mathbf{K}\mathbf{x} = \mathbf{b}$
- A **key challenge**: identify an efficient preconditioner P which makes solving

$$\mathbf{K}\mathrm{P}^{-1}\mathbf{z} = \mathbf{b}, \text{ with } \mathbf{z} = \mathbf{K}\mathbf{x},$$

  quick and which is also amenable for a parallel implementation

- A special **feature** of many challenging problems: strong (physical) coupling of constituent fields
- **Basic approaches** to design preconditioners: Fully **algebraic** or **physics**-based/block preconditioning

# Background cntd.

- Traditionally in Elmer: the algebraic approach, like ILU
- Coupled multi-physic problems via segregation:

$$F_1(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \mathbf{0}$$

$$\ldots$$

$$F_N(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \mathbf{0}$$

- Solved via Gauss-Seidel type of iteration:

$$F_1(\mathbf{x}_1^{(k+1)}, \mathbf{x}_2^{(k)}, \mathbf{x}_3^{(k)}, \ldots, \mathbf{x}_N^{(k)}) = \mathbf{0}$$

$$F_2(\mathbf{x}_1^{(k+1)}, \mathbf{x}_2^{(k+1)}, \mathbf{x}_3^{(k)}, \ldots, \mathbf{x}_N^{(k)}) = \mathbf{0}$$

$$\ldots$$

# *Background cntd.*

- If you succeed with an algebraic preconditioner: smile, whistle and be happy
  - no other approach will usually outperform your current one
  - BUT: In difficult situations (= large parallel runs) you usually fail with purely algebraic preconditioning
- In such (difficult) cases: Physics-based/block preconditioning
  - Alternatively: monolithic discretization (=all variables in one sweep); direct solvers
  - Examples: Ice flow: (equation of motion + free surface + incompressibility); Acoustic wave propagation: (equation of motion + energy conservation + continuity); Coupled systems: utilize the block structure of the monolithic system to derive a preconditioner

# Design of a new preconditioner

- Solution of: $\mathbf{K}\mathsf{P}^{-1}\mathbf{z} = \mathbf{b}$
- Traditionally: produce iterates of $\mathbf{z} = \mathsf{P}\mathbf{x}$
- New approach: minimize $||\mathbf{b} - \mathbf{K}\mathbf{x}^{(k)}||$

  Residual

  over $\mathcal{V}_k = \mathbf{x}^{(0)} + \mathrm{span}\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(k)}\}$

  Search directions
- Preconditioner = operator, which from previous iterate produces new search directions by solution of:

Residual correction system

$$\mathsf{P}\mathbf{s}^{(k+1)} = \mathbf{b} - \mathbf{K}\mathbf{x}^{(k)} \qquad \mathsf{P} \approx \mathbf{K}$$

# Algorithm (GCR)

- Implemented in the *ParStokes* solver
- Needs additional pseudo-solvers to provide the matrix space for velocity as well as pressure block
- Use only for large scale problems, where algebraic preconditioner + Krylov-subspace solvers don’t work and direct solver (MUMPS) exceed sensible memory resources

```
$k = 0$
$\mathbf{r}^{(k)} = \mathbf{f} - \mathbf{K}\mathbf{u}^{(k)}$
while $(\|\mathbf{r}^{(k)}\| < TOL\|\mathbf{f}\|$ and $k < m)$
    Generate the search direction $\mathbf{s}^{(k+1)}$
    $\mathbf{v}^{(k+1)} = \mathbf{K}\mathbf{s}^{(k+1)}$
    do $j = 1, k$
        $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k+1)} - \langle \mathbf{v}^{(j)}, \mathbf{v}^{(k+1)} \rangle \mathbf{v}^{(j)}$
        $\mathbf{s}^{(k+1)} = \mathbf{s}^{(k+1)} - \langle \mathbf{v}^{(j)}, \mathbf{v}^{(k+1)} \rangle \mathbf{s}^{(j)}$
    end do
    $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k+1)} / \|\mathbf{v}^{(k+1)}\|$
    $\mathbf{s}^{(k+1)} = \mathbf{s}^{(k+1)} / \|\mathbf{v}^{(k+1)}\|$
    $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \langle \mathbf{v}^{(k+1)}, \mathbf{r}^{(k)} \rangle \mathbf{s}^{(k+1)}$
    $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \langle \mathbf{v}^{(k+1)}, \mathbf{r}^{(k)} \rangle \mathbf{v}^{(k+1)}$
    $k = k + 1$
end while
```

# *Requirements*

- **Robustness**:
  - Iteration counts do not depend on the problem size
  - Robust with respect to variations of essential model parameters
  - If robust, parallel scalability (weak) depends heavily on the scalability of the subsidiary computations
- **Efficiency**:
  - The subsidiary computations corresponding to the application of the preconditioner done efficiently by exploiting optimal complexity solvers.
  - Need preconditioners the action of which operations may be computed by solving elementary models
- We focus here on exploring to what extent the requirements of the robustness and efficiency are met in the case of the examples considered.

# *Full Stokes*

- Solver for:

$$-\operatorname{div}[2\eta(\mathbf{D})\mathbf{D}(\mathbf{v})] + \nabla p = \rho\mathbf{g},$$
$$-\operatorname{div}\mathbf{v} = 0$$

- Strain-rate tensor

$$\mathbf{D} = \mathbf{D}(\mathbf{v}) = 1/2(\nabla\mathbf{v} + \nabla\mathbf{v}^T).$$

- Glen’s flow law:

$$\eta = 1/2A^{-k}[I_2(\mathbf{D})]^{(k-1)/2}$$

$$= 1/2(\mathbf{D}\cdot\mathbf{D})$$

# *Weak formulation + linearization*

- Find for any $(\mathbf{z}, q) \in \mathcal{V}$ a set of $(\mathbf{v}, p) \in \mathcal{U}$ such that

$$\int_\Omega 2\mu(\mathbf{D}(\mathbf{v}_k))\mathbf{D}(\mathbf{v}_{k+1}) \cdot \mathbf{D}(\mathbf{z})\, d\Omega - \int_\Omega p_{k+1} \nabla \cdot \mathbf{z}\, d\Omega = \int_\Omega \mathbf{b} \cdot \mathbf{z}\, d\Omega + \int_{\Gamma_N} \hat{\mathbf{s}} \cdot \mathbf{v},$$

$$-\int_\Omega \nabla \cdot \mathbf{v}_{k+1} q\, d\Omega = 0$$

Picard linearization

# Stabilization

- Inf-Sup condition: stabilization by using different approximation spaces for velocity and pressure (saddle-point problem)
- Bubble stabilization: $V_h = S_h + B_h$

$$B_h = \{v_h \mid v_{h|K} \in P_r(K) \text{ and } v_{h|\partial K} = 0 \text{ for any element } K\}$$

- Recommended degrees of bubbles: brick 7, tetrahedron 5, wedge 6
- Bubbles are eliminated from matrix, but cost during assembly

# The preconditioner

Velocity block ~ Laplacian

grad p

- The full linearized system:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix}$$

continuity

stabilization

- The preconditioner:

$$\mathsf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$$

Pressure-Schur complement

- Replacement of pressure-Schur complement: $\mathbf{Q} = \mu(\mathbf{D})^{-1}\mathbf{M}$

# *Performance*

- A thin domain ⇒ high element aspect ratios ⇒ weakened finite element
  - stability may have an effect on the effectiveness of the preconditioner
- The robustness of the preconditioner with respect to natural variations of the ice viscosity
- The solver performance for different linearization strategies
- The use of the stress-divergence form couples the solution of the components of the velocity, i.e. $A$ is not block diagonal ⇒ ways to utilize component-wise linear solves?

# *Different linearization strategies*

| | Picard | Hybrid $\delta_{NL} = 10^{-1}/2$ | Hybrid $\delta_{NL} = 10^{-2}$ |
|---|---|---|---|
| Nonlin Step | Iters | Linearization/Iters | Linearization/Iters |
| 0 | 24 | Picard/24 | Picard/24 |
| 1 | 20 | Picard/20 | Picard/20 |
| 2 | 19 | Picard/19 | Picard/19 |
| 3 | 18 | Picard/18 | Picard/18 |
| 4 | 17 | Picard/17 | Picard/17 |
| 5 | 15 | Newton/20 | Picard/15 |
| 6 | 14 | Newton/19 | Picard/14 |
| 7 | 13 | Newton/15 | Picard/13 |
| 8 | 13 | Newton/7 | Picard/13 |
| 9 | 12 | Convergence | Newton/16 |
| 10 | 11 | | Newton/14 |
| 11 | 10 | | Newton/7 |
| 12 | 9 | | Convergence |
| 13...24 | 4.5 (Aver.) | | |
| 25 | Convergence | | |

# Block diagonal approximation

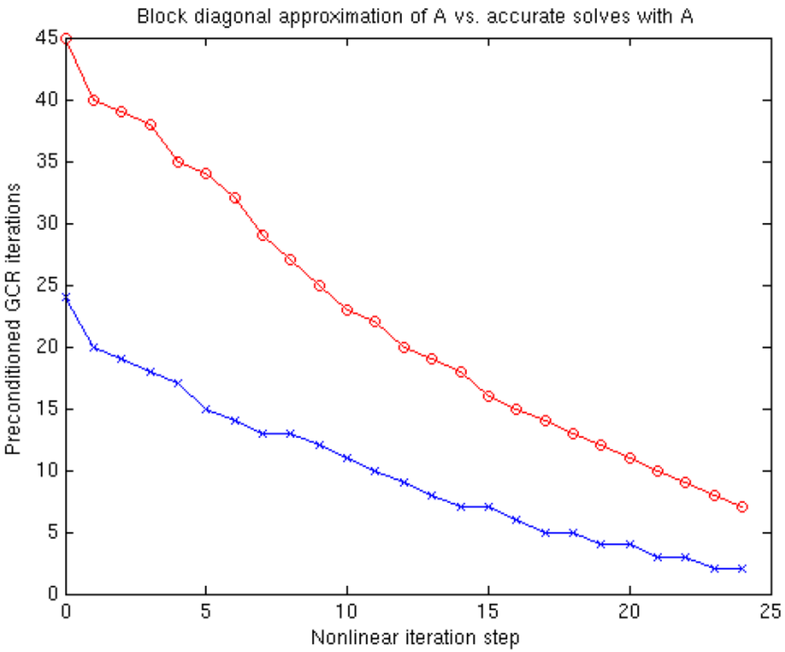
- For Picard linearization

$$\mathsf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$$

$$\Downarrow$$

$$\mathsf{P} = \begin{bmatrix} \mathsf{P}_A & \mathbf{B}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix}$$

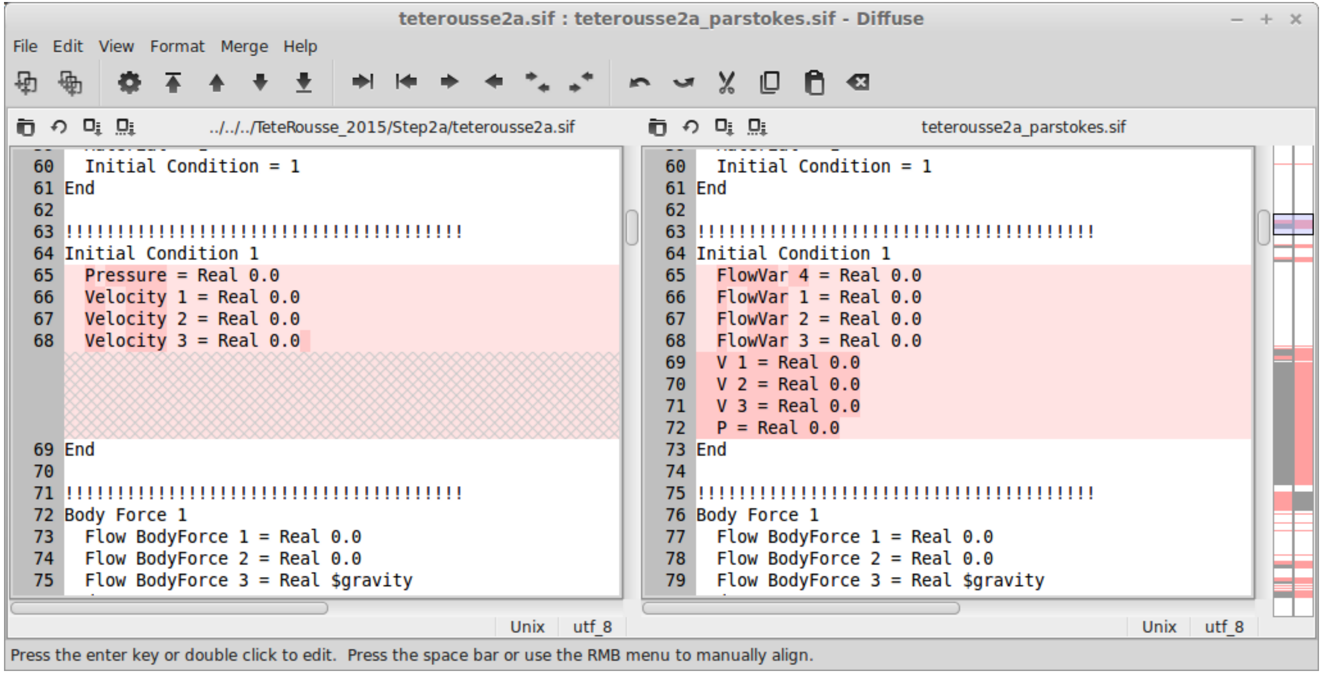
$$P_A \sim \mathrm{diag}(\varepsilon\Delta\mathbf{v}_1, \varepsilon\Delta\mathbf{v}_2, \varepsilon\Delta\mathbf{v}_3)$$



# *Example: Tete-Rousse*

- Replacing standard Navier-Stokes solver in teterousse2a.sif with ParStokes
- Warning: you will be disappointed in terms of performance, because:
  - This is a very small case
  - The aspect ratio of elements is very small
  - The original case works with algebraic pre-conditioner, which always is faster
- So, this is just a demo on how to set up the simulation
- Next slides show the side-by-side changes

# Example: Tete-Rousse

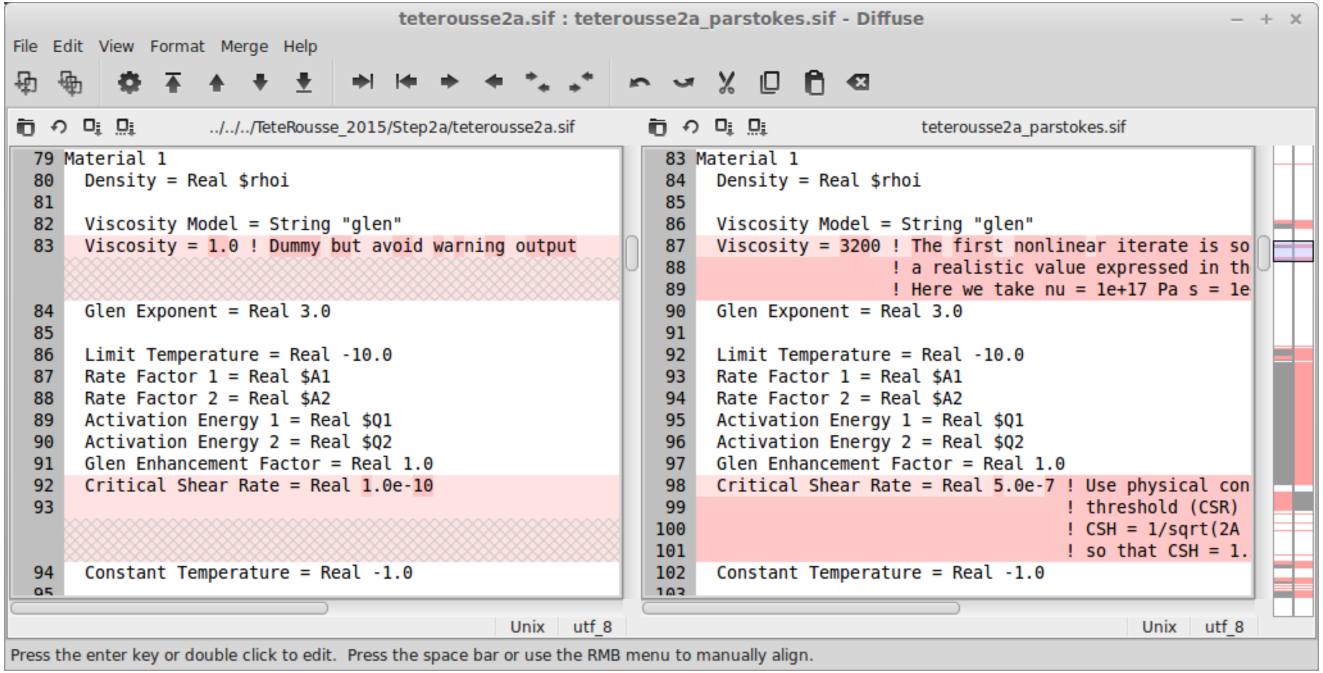


```
  Initial Condition = 1
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Initial Condition 1
  Pressure = Real 0.0
  Velocity 1 = Real 0.0
  Velocity 2 = Real 0.0
  Velocity 3 = Real 0.0
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Body Force 1
  Flow BodyForce 1 = Real 0.0
  Flow BodyForce 2 = Real 0.0
  Flow BodyForce 3 = Real $gravity
```

```
  Initial Condition = 1
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Initial Condition 1
  FlowVar 4 = Real 0.0
  FlowVar 1 = Real 0.0
  FlowVar 2 = Real 0.0
  FlowVar 3 = Real 0.0
  V 1 = Real 0.0
  V 2 = Real 0.0
  V 3 = Real 0.0
  P = Real 0.0
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Body Force 1
  Flow BodyForce 1 = Real 0.0
  Flow BodyForce 2 = Real 0.0
  Flow BodyForce 3 = Real $gravity
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

# Example: Tete-Rousse

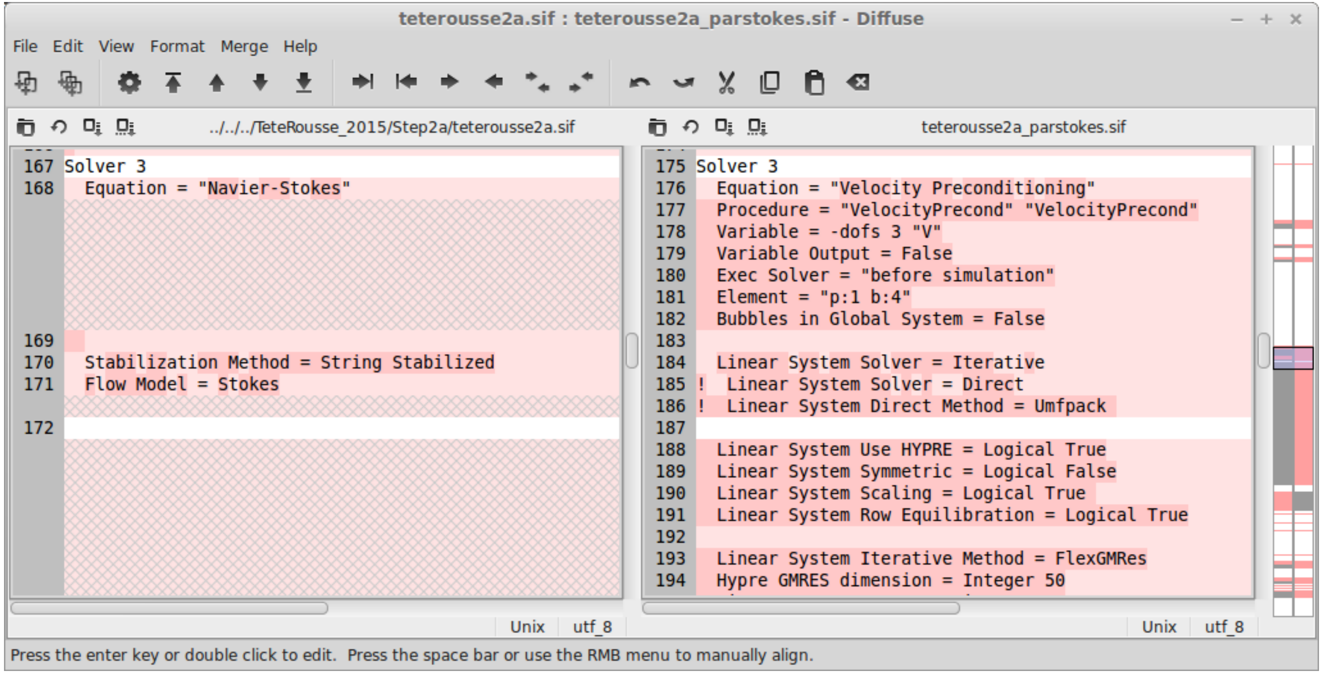


```
Material 1
  Density = Real $rhoi

  Viscosity Model = String "glen"
  Viscosity = 1.0 ! Dummy but avoid warning output
  Glen Exponent = Real 3.0

  Limit Temperature = Real -10.0
  Rate Factor 1 = Real $A1
  Rate Factor 2 = Real $A2
  Activation Energy 1 = Real $Q1
  Activation Energy 2 = Real $Q2
  Glen Enhancement Factor = Real 1.0
  Critical Shear Rate = Real 1.0e-10

  Constant Temperature = Real -1.0
```

```
Material 1
  Density = Real $rhoi

  Viscosity Model = String "glen"
  Viscosity = 3200 ! The first nonlinear iterate is so
                   ! a realistic value expressed in th
                   ! Here we take nu = 1e+17 Pa s = 1e
  Glen Exponent = Real 3.0

  Limit Temperature = Real -10.0
  Rate Factor 1 = Real $A1
  Rate Factor 2 = Real $A2
  Activation Energy 1 = Real $Q1
  Activation Energy 2 = Real $Q2
  Glen Enhancement Factor = Real 1.0
  Critical Shear Rate = Real 5.0e-7 ! Use physical con
                                    ! threshold (CSR)
                                    ! CSH = 1/sqrt(2A
                                    ! so that CSH = 1.
  Constant Temperature = Real -1.0
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

# Example: Tete-Rousse

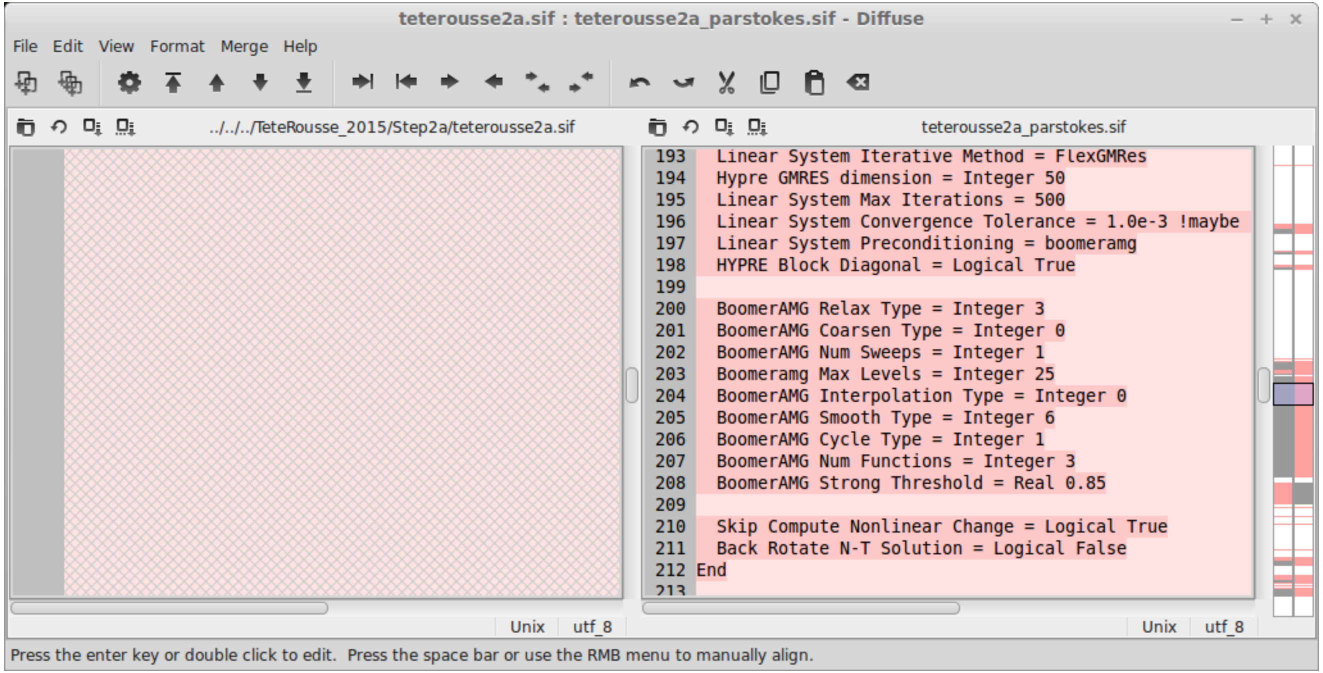


```
Solver 3
  Equation = "Navier-Stokes"

  Stabilization Method = String Stabilized
  Flow Model = Stokes

```

```
Solver 3
  Equation = "Velocity Preconditioning"
  Procedure = "VelocityPrecond" "VelocityPrecond"
  Variable = -dofs 3 "V"
  Variable Output = False
  Exec Solver = "before simulation"
  Element = "p:1 b:4"
  Bubbles in Global System = False

  Linear System Solver = Iterative
!  Linear System Solver = Direct
!  Linear System Direct Method = Umfpack

  Linear System Use HYPRE = Logical True
  Linear System Symmetric = Logical False
  Linear System Scaling = Logical True
  Linear System Row Equilibration = Logical True

  Linear System Iterative Method = FlexGMRes
  Hypre GMRES dimension = Integer 50
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

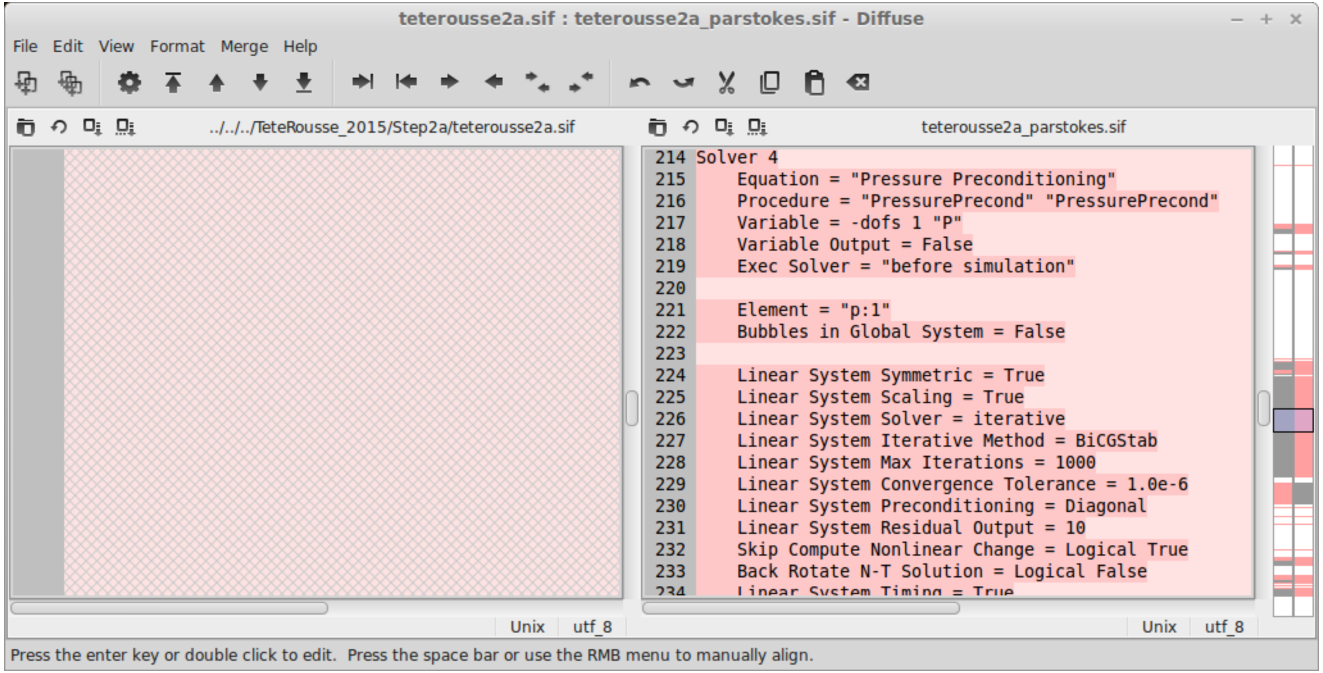
# Example: Tete-Rousse



```
  Linear System Iterative Method = FlexGMRes
  Hypre GMRES dimension = Integer 50
  Linear System Max Iterations = 500
  Linear System Convergence Tolerance = 1.0e-3 !maybe
  Linear System Preconditioning = boomeramg
  HYPRE Block Diagonal = Logical True

  BoomerAMG Relax Type = Integer 3
  BoomerAMG Coarsen Type = Integer 0
  BoomerAMG Num Sweeps = Integer 1
  Boomeramg Max Levels = Integer 25
  BoomerAMG Interpolation Type = Integer 0
  BoomerAMG Smooth Type = Integer 6
  BoomerAMG Cycle Type = Integer 1
  BoomerAMG Num Functions = Integer 3
  BoomerAMG Strong Threshold = Real 0.85

  Skip Compute Nonlinear Change = Logical True
  Back Rotate N-T Solution = Logical False
End
```

Press the enter key or double click to edit.  Press the space bar or use the RMB menu to manually align.

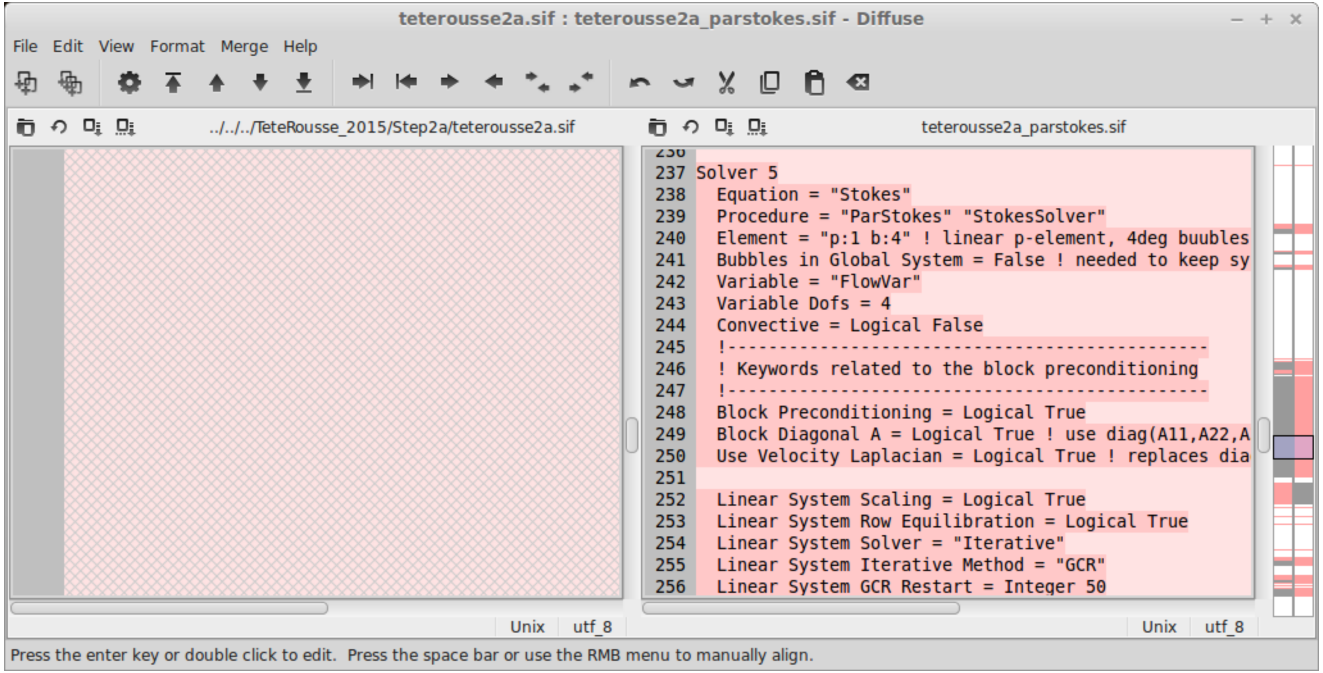
# Example: Tete-Rousse



```
Solver 4
    Equation = "Pressure Preconditioning"
    Procedure = "PressurePrecond" "PressurePrecond"
    Variable = -dofs 1 "P"
    Variable Output = False
    Exec Solver = "before simulation"

    Element = "p:1"
    Bubbles in Global System = False

    Linear System Symmetric = True
    Linear System Scaling = True
    Linear System Solver = iterative
    Linear System Iterative Method = BiCGStab
    Linear System Max Iterations = 1000
    Linear System Convergence Tolerance = 1.0e-6
    Linear System Preconditioning = Diagonal
    Linear System Residual Output = 10
    Skip Compute Nonlinear Change = Logical True
    Back Rotate N-T Solution = Logical False
    Linear System Timing = True
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

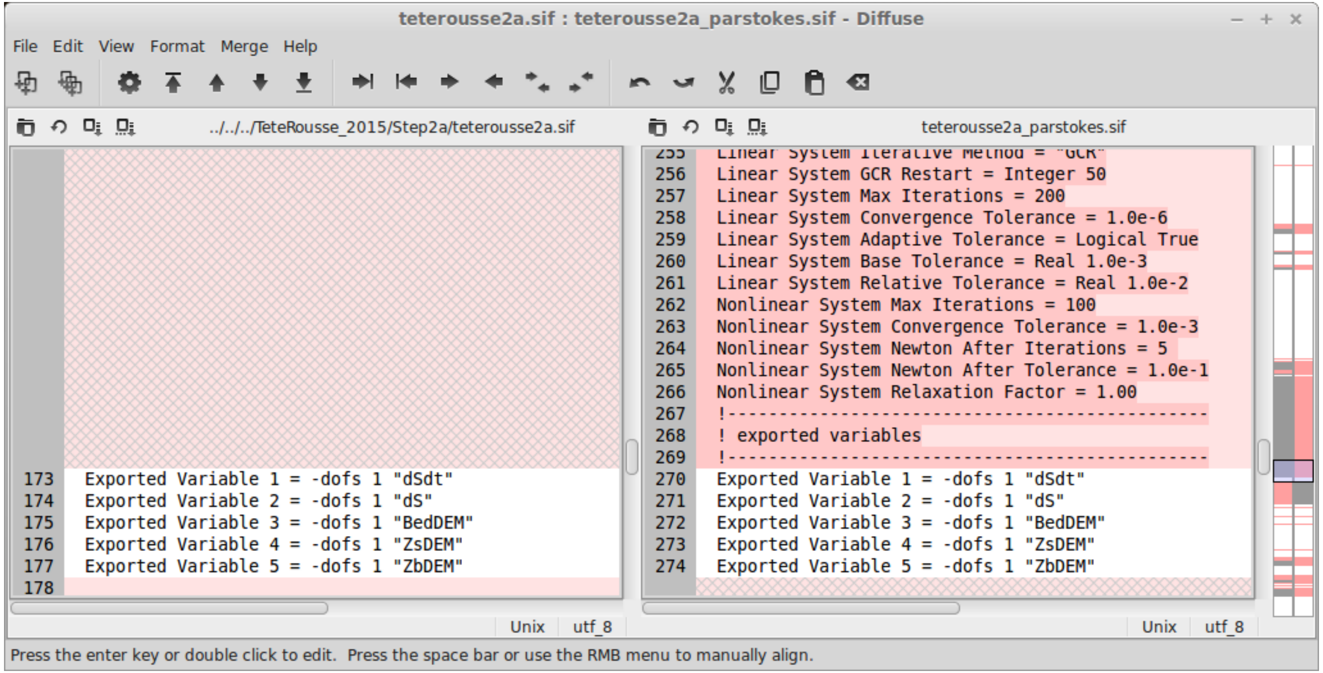
# Example: Tete-Rousse



```
Solver 5
  Equation = "Stokes"
  Procedure = "ParStokes" "StokesSolver"
  Element = "p:1 b:4" ! linear p-element, 4deg buubles
  Bubbles in Global System = False ! needed to keep sy
  Variable = "FlowVar"
  Variable Dofs = 4
  Convective = Logical False
  !----------------------------------------------------
  ! Keywords related to the block preconditioning
  !----------------------------------------------------
  Block Preconditioning = Logical True
  Block Diagonal A = Logical True ! use diag(A11,A22,A
  Use Velocity Laplacian = Logical True ! replaces dia

  Linear System Scaling = Logical True
  Linear System Row Equilibration = Logical True
  Linear System Solver = "Iterative"
  Linear System Iterative Method = "GCR"
  Linear System GCR Restart = Integer 50
```

Press the enter key or double click to edit.  Press the space bar or use the RMB menu to manually align.

# Example: Tete-Rousse



teterousse2a.sif : teterousse2a_parstokes.sif - Diffuse

../../../TeteRousse_2015/Step2a/teterousse2a.sif

```
Exported Variable 1 = -dofs 1 "dSdt"
Exported Variable 2 = -dofs 1 "dS"
Exported Variable 3 = -dofs 1 "BedDEM"
Exported Variable 4 = -dofs 1 "ZsDEM"
Exported Variable 5 = -dofs 1 "ZbDEM"
```

teterousse2a_parstokes.sif

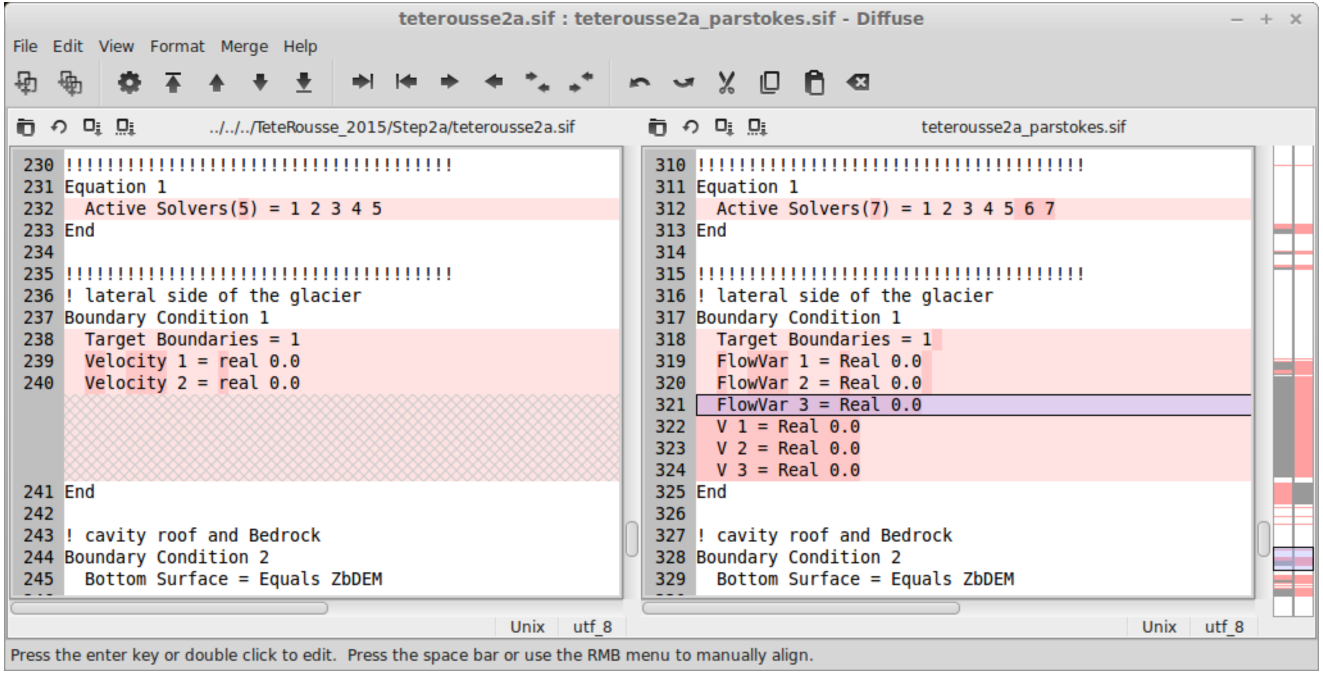
```
Linear System Iterative Method = "GCR"
Linear System GCR Restart = Integer 50
Linear System Max Iterations = 200
Linear System Convergence Tolerance = 1.0e-6
Linear System Adaptive Tolerance = Logical True
Linear System Base Tolerance = Real 1.0e-3
Linear System Relative Tolerance = Real 1.0e-2
Nonlinear System Max Iterations = 100
Nonlinear System Convergence Tolerance = 1.0e-3
Nonlinear System Newton After Iterations = 5
Nonlinear System Newton After Tolerance = 1.0e-1
Nonlinear System Relaxation Factor = 1.00
!----------------------------------------------
! exported variables
!----------------------------------------------
Exported Variable 1 = -dofs 1 "dSdt"
Exported Variable 2 = -dofs 1 "dS"
Exported Variable 3 = -dofs 1 "BedDEM"
Exported Variable 4 = -dofs 1 "ZsDEM"
Exported Variable 5 = -dofs 1 "ZbDEM"
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

# *Example: Tete-Rousse*

- Change all occurrences of “Flow Solution” into “FlowVar”
- And add the Flow-Preconditioner Variable “V” to boundary conditions
- Also, don’t forget to increase numbers of solvers in “Equation 1”
- And to change name for output, in order to compare the results

# Example: Tete-Rousse

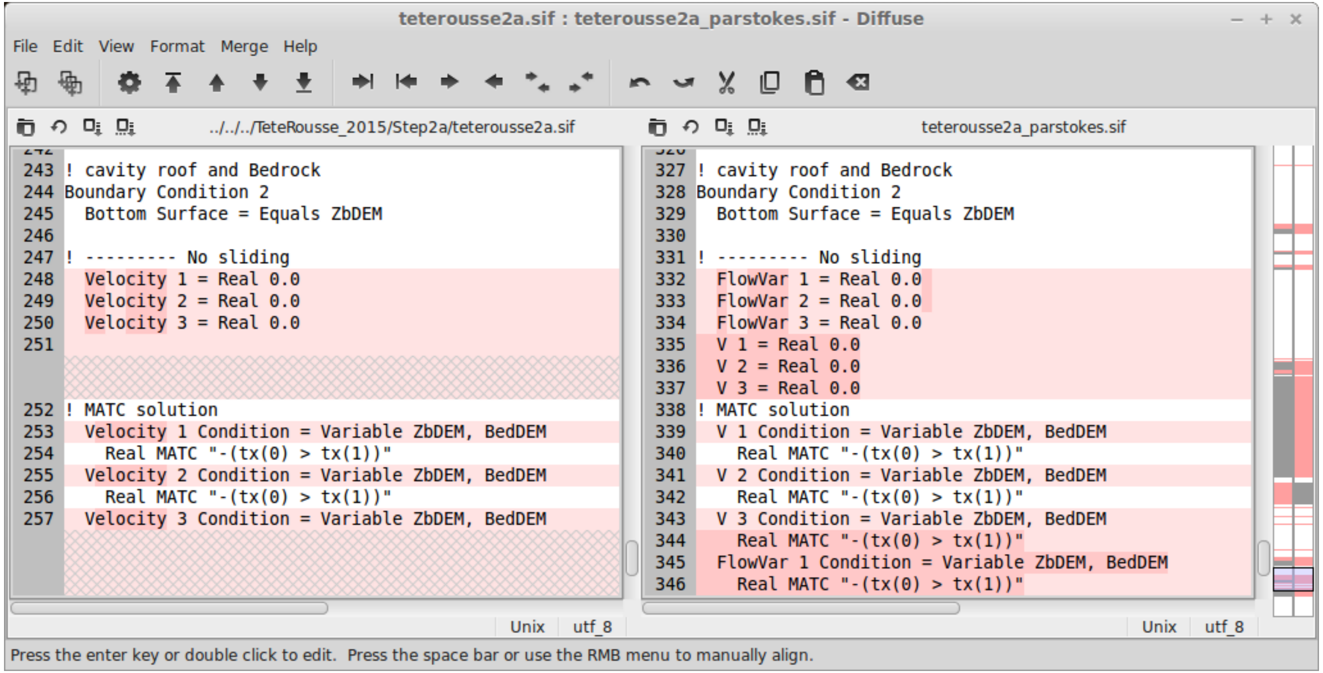


```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Equation 1
  Active Solvers(5) = 1 2 3 4 5
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! lateral side of the glacier
Boundary Condition 1
  Target Boundaries = 1
  Velocity 1 = real 0.0
  Velocity 2 = real 0.0
End

! cavity roof and Bedrock
Boundary Condition 2
  Bottom Surface = Equals ZbDEM
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Equation 1
  Active Solvers(7) = 1 2 3 4 5 6 7
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! lateral side of the glacier
Boundary Condition 1
  Target Boundaries = 1
  FlowVar 1 = Real 0.0
  FlowVar 2 = Real 0.0
  FlowVar 3 = Real 0.0
  V 1 = Real 0.0
  V 2 = Real 0.0
  V 3 = Real 0.0
End

! cavity roof and Bedrock
Boundary Condition 2
  Bottom Surface = Equals ZbDEM
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

# Example: Tete-Rousse

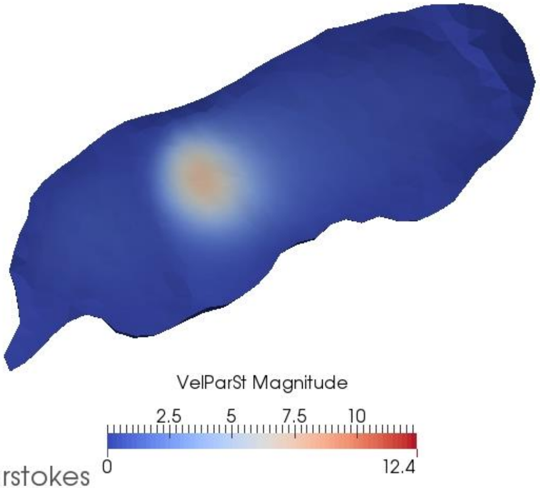


```
! cavity roof and Bedrock
Boundary Condition 2
  Bottom Surface = Equals ZbDEM

! --------- No sliding
  Velocity 1 = Real 0.0
  Velocity 2 = Real 0.0
  Velocity 3 = Real 0.0

! MATC solution
  Velocity 1 Condition = Variable ZbDEM, BedDEM
    Real MATC "-(tx(0) > tx(1))"
  Velocity 2 Condition = Variable ZbDEM, BedDEM
    Real MATC "-(tx(0) > tx(1))"
  Velocity 3 Condition = Variable ZbDEM, BedDEM
```

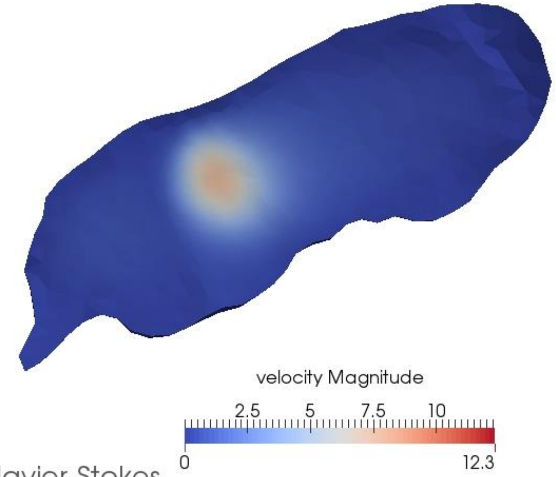
```
! cavity roof and Bedrock
Boundary Condition 2
  Bottom Surface = Equals ZbDEM

! --------- No sliding
  FlowVar 1 = Real 0.0
  FlowVar 2 = Real 0.0
  FlowVar 3 = Real 0.0
  V 1 = Real 0.0
  V 2 = Real 0.0
  V 3 = Real 0.0
! MATC solution
  V 1 Condition = Variable ZbDEM, BedDEM
    Real MATC "-(tx(0) > tx(1))"
  V 2 Condition = Variable ZbDEM, BedDEM
    Real MATC "-(tx(0) > tx(1))"
  V 3 Condition = Variable ZbDEM, BedDEM
    Real MATC "-(tx(0) > tx(1))"
  FlowVar 1 Condition = Variable ZbDEM, BedDEM
    Real MATC "-(tx(0) > tx(1))"
```

Press the enter key or double click to edit. Press the space bar or use the RMB menu to manually align.

# *Example: Tete-Rousse*



Parstokes



Navier-Stokes