Elmer/Ice
Rovaniemi 2018

Shallow models in Elmer/Ice

Fabien Gillet-Chaulet & Olivier Gagliardini

IGE - Grenoble - France

Outline

- ✓ Shallow Shelf / Shallow stream Solver
- ✓ Thickness Solver
- ✓ A glacier example

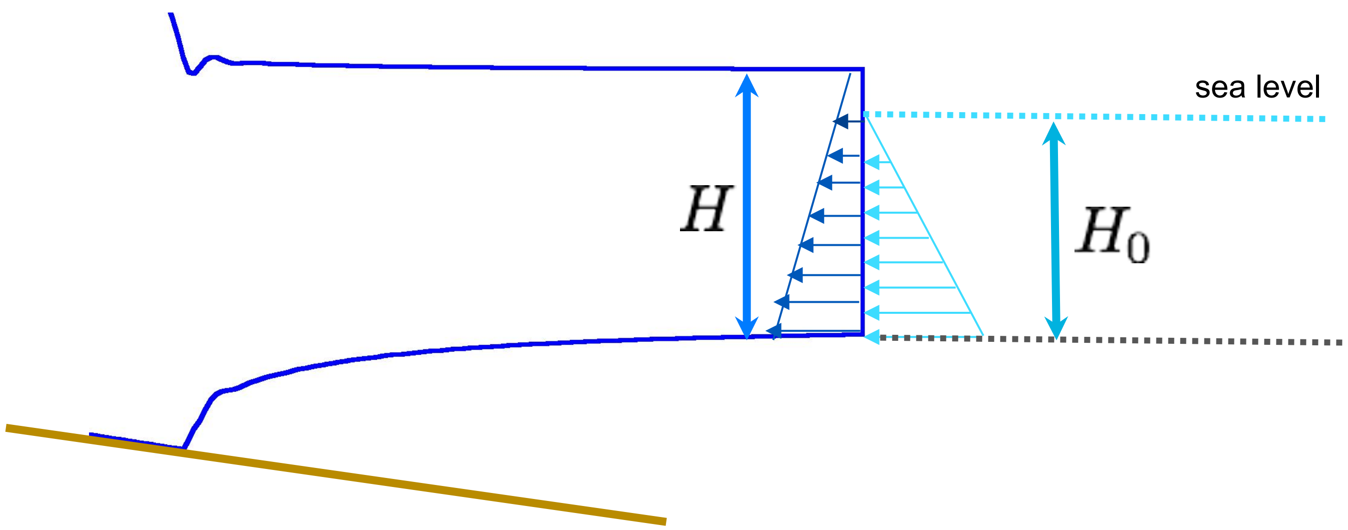
Shallow Shelf Approximation/Shallow Stream Approximation

Field equations:

$$\begin{cases} \frac{\partial}{\partial x}\left(2H\nu\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)-\beta u=\rho g H\frac{\partial z_s}{\partial x} \\ \frac{\partial}{\partial x}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(2H\nu\left(\frac{\partial u}{\partial x}+2\frac{\partial v}{\partial y}\right)\right)-\beta v=\rho_i g H\frac{\partial z_s}{\partial y} \end{cases}$$

Boundary Conditions:

$$\begin{cases} 4H\nu\frac{\partial u}{\partial x}n_x+2H\nu\frac{\partial v}{\partial y}n_x+H\nu(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial x})n_y=(\rho_i g H-\rho_w g H_0)n_x \\ 4H\nu\frac{\partial v}{\partial y}n_y+2H\nu\frac{\partial v}{\partial x}n_y+H\nu(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial x})n_x=(\rho_i g H-\rho_w g H_0)n_y \end{cases}$$



Shallow Shelf Approximation/Shallow Stream Approximation

Field equations:

$$\begin{cases} \frac{\partial}{\partial x}\left(2H\nu\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)-\beta u=\rho g H\frac{\partial z_s}{\partial x} \\ \frac{\partial}{\partial x}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(2H\nu\left(\frac{\partial u}{\partial x}+2\frac{\partial v}{\partial y}\right)\right)-\beta v=\rho_i g H\frac{\partial z_s}{\partial y} \end{cases}$$

$$H = Zs - Zb$$

Elmer/Ice Solvers:

Solver Fortran File: `SSASolver.f90`
Solver Name: `SSABasalSolver`

`Required Output Variable(s):`

- `SSAVelocity`

Required Input Variable(s):

- (1) `Zb`, `Zs` and `Effective Pressure` when using the Coulomb type friction law

The SSABasalSolver solve the classical SSA equation, it has been modified in Rev. 6440 to be executed either on a grid of dimension lower than the problem dimension itself (i.e. the top or bottom grid of a 2D or 3D mesh for a SSA 1D or 2D problem), or on a grid of the same dimension of the problem (i.e. 2D mesh for a 2D plane view SSA solution).

It will work on a 3D mesh only if the mesh as been extruded along the vertical direction and if the base line boundary conditions have been preserved (to impose neumann conditions). **Keyword *«Preserve Baseline = Logical True»* in section Simulation**

Shallow Shelf Approximation/Shallow Stream Approximation

Field equations:

$$\begin{cases} \frac{\partial}{\partial x}\left(2H\nu\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)-\beta u=\rho g H\frac{\partial z_s}{\partial x} \\ \frac{\partial}{\partial x}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(2H\nu\left(\frac{\partial u}{\partial x}+2\frac{\partial v}{\partial y}\right)\right)-\beta v=\rho_i g H\frac{\partial z_s}{\partial y} \end{cases}$$

SIF - Solver Section:

```
Solver 1
  Equation = "SSA"
  Procedure = File "ElmerIceSolvers" "SSABasalSolver"
  Variable = String "SSAVelocity"
  Variable DOFs = 2   ! 1 in SSA 1-D or 2 in SSA-2D

  Linear System Solver = Direct
  Linear System Direct Method = umfpack

  Nonlinear System Max Iterations = 100
  Nonlinear System Convergence Tolerance  = 1.0e-08
  Nonlinear System Newton After Iterations = 5
  Nonlinear System Newton After Tolerance = 1.0e-05

  Nonlinear System Relaxation Factor = 1.00

  Steady State Convergence Tolerance = Real 1.0e-3
End
```

Shallow Shelf Approximation/Shallow Stream Approximation

Field equations:

$$\begin{cases} \frac{\partial}{\partial x}\left(2H\nu\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)-\beta u=\rho_i gH\frac{\partial z_s}{\partial x} \\ \frac{\partial}{\partial x}\left(H\nu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(2H\nu\left(\frac{\partial u}{\partial x}+2\frac{\partial v}{\partial y}\right)\right)-\beta v=\rho_i gH\frac{\partial z_s}{\partial y} \end{cases}$$

SIF - Material Section:

```
Material 1
! Flow Law
  Viscosity Exponent = Real $1.0/n
  Critical Shear Rate = Real 1.0e-10
  SSA Mean Viscosity = Real $eta
  SSA Mean Density = Real $rhoi

! Friction Law
  ! Which law are we using
  SSA Friction Law = String («linear», «weertman» or «coulomb»)

  ! friction parameter
  SSA Friction Parameter = Real 0.1

! Needed for Weertman and Coulomb
  ! Exponent m
  SSA Friction Exponent = Real $1.0/n

  ! Min velocity for linearisation where ub=0
  SSA Friction Linear Velocity = Real 0.0001

! Needed for Coulomb only
  ! post peak exponent in the Coulomb law (q, in Gagliardini et al., 2007)
  SSA Friction Post-Peak = Real ...
  ! Iken's bound  tau_b/N < C (see Gagliardini et al., 2007)
  SSA Friction Maximum Value = Real ....
  SSA Min Effective Pressure = Real ...
End
```

Friction laws:

- Linear:

$$\tau_b = \beta u$$

- Weertman:

$$\tau_b = \beta |u|^{(m-1)} u$$

- Coulomb:

$$\tau_b = \frac{1}{A_s^{\frac{1}{n}}}\left[\frac{1}{\left(1+\alpha . \chi^q\right)}\right]^{\frac{1}{n}} . u_b^{\frac{1}{n}-1} . u$$

$$\alpha = \frac{(q-1)^{q-1}}{q^q} \qquad \chi = \frac{u_b}{C^n N^n A_s}$$

Shallow Shelf Approximation/Shallow Stream Approximation

Boundary Conditions:

$$\begin{cases} 4H\nu\dfrac{\partial u}{\partial x}n_x + 2H\nu\dfrac{\partial v}{\partial y}n_x + H\nu(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial x})n_y = (\rho_i g H - \rho_w g H_0)n_x \\ 4H\nu\dfrac{\partial v}{\partial y}n_y + 2H\nu\dfrac{\partial v}{\partial x}n_y + H\nu(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial x})n_x = (\rho_i g H - \rho_w g H_0)n_y \end{cases}$$

SIF - Boundary Conditions / Constants / Body Forces:

```
Boundary Condition 1
! Dirichlet condition
  SSAVelocity 1 = Real ...
  SSAVelocity 2 = Real ...
End
Boundary Condition 1
! Neumann Condition
  Calving Front = Logical True
End
```

```
Constants
! Used for Neumann condition
  Water Density = Real ....
  Sea Level = Real ...
End
```

```
Body Force 1
! The gravity from Flow Body Force 2/3 (1D/2D)
    Flow BodyForce 3 = Real $gravity
End
```

Computing mean values (case of a 3d mesh)

SSA uses mean viscosity and density:

$$\nu(x,y) = \frac{1}{H}\int_{z_b}^{z_s} \mu(x,y,z)dz \longrightarrow$$ coupling with : **Temperature, Damage**

$$\bar{\rho}(x,y) = \frac{1}{H}\int_{z_b}^{z_s} \rho(x,y,z)dz \longrightarrow$$ coupling with : **Density**

You can use:

Elmer/Ice solver : *GetMeanValueSolver*

- **unstructured** meshes in the vertical direction

```
Solver 1
  Equation = "SSA-IntValue"
  Procedure = File "ElmerIceSolvers" "GetMeanValueSolver"
  Variable = -nooutput String "Integrated variable"
  Variable DOFs = 1

  Exported Variable 1 = String "Mean Viscosity"
  Exported Variable 1 DOFs = 1
  Exported Variable 2 = String "Mean Density"
  Exported Variable 2 DOFs = 1

  Linear System Solver = Direct
  Linear System Direct Method = umfpack

  Steady State Convergence Tolerance = Real 1.0e-3
End


!!! Upper free surface
Boundary Condition 1
  Depth = Real 0.0
  Mean Viscosity = Real 0.0
  Mean Density = real 0.0
End
```

Elmer solver : *StructuredProjectToPlane*

- **structured** meshes in the vertical direction

```
Solver 1
  Equation = "HeightDepth"
  Procedure = "StructuredProjectToPlane" "StructuredProjectToPlane"
  Active Coordinate = Integer 3

  Operator 1 = depth
  Operator 2 = height
  Operator 3 = thickness

  !! compute the integrated horizontal Viscosity and Density
  Variable 4 = Viscosity
  Operator 4 = int

  Variable 5 = Density
  Operator 5 = int
End

Material 1
  SSA Mean Viscosity = Variable "int Viscosity", thickness
       REAL MATC "tx(0)/tx(1)"
  SSA Mean Density = Variable "int Density", thickness
       REAL MATC "tx(0)/tx(1)"
End
```

Outline

- ✓ Shallow Shelf / Shallow stream Solver
- ✓ **Thickness Solver**
- ✓ A glacier example

Thickness Solver

Field equations:

$$\frac{\partial H}{\partial t} + \nabla(\bar{u}H) = a_s + a_b$$

Elmer/Ice Solvers:

- **Solver Fortran File:** `ThicknessSolver.f90`
- **Solver Name:** `ThicknessSolver`
- **Required Output Variable(s):** `H`
- **Required Input Variable(s):** `H residual`
- **Optional Output Variable(s):** `dhdt`
- **Optional Input Variable(s):** `FlowSolution`

• This solver is based on the FreeSurfaceSolver and use a **SUPG stabilsation** scheme by **default** (***residual free bubble stabilization*** can be use instead).

• As for the FreeSurfaceSolver ***Min*** **and** ***Max*** **limiters** can be used.

• As for the Free surface solver **only a Dirichlet boundary condition** can be imposed.

• This solver can be used on a mesh of the same dimension as the problem (e.g. solve on the bottom or top boundary of a 3d mesh to solve the 2d thickness field) or on a mesh of lower dimension (e.g. can be use in a 2D plane view mesh with the SSA solver for example)

Thickness Solver

Field equations: $$\frac{\partial H}{\partial t} + \nabla(\bar{u}H) = a_s + a_b$$

SIF:

```
Solver 1
   Equation = "Thickness"
   Variable = -dofs 1 "H"

   Exported Variable 1 = -dofs 1 "H Residual"

!! To compute dh/dt
   Exported Variable 2 = -dofs 1 "dHdt"
   Compute dHdT = Logical True

  Procedure = "ElmerIceSolvers" "ThicknessSolver"
!   Before Linsolve = "EliminateDirichlet" "EliminateDirichlet"

   Linear System Solver = Direct
   Linear System Direct Method = umfpack
   Linear System Convergence Tolerance = Real 1.0e-12

! equation is linear if no min/max
   Nonlinear System Max Iterations = 50
   Nonlinear System Convergence Tolerance  = 1.0e-6
   Nonlinear System Relaxation Factor = 1.00

! stabilisation method: [stabilized\bubbles]
  Stabilization Method = stabilized

!! to apply Min/Max limiters
  Apply Dirichlet = Logical True

!! to use horizontal ALE formulation
   ALE Formulation = Logical True

!! To get the mean horizontal velocity
!!  either give the name of the variable
     Flow Solution Name = String "SSAVelocity"
!!!!! or give the dimension of the problem using:
!    Convection Dimension = Integer
End
```

```
Body Force 1
!! Mass balance
  Top Surface Accumulation = Real ....
  Bottom Surface Accumulation = Real ....


!! if the convection velocity is not directly given by a variable
!! Then give //Convection Dimension = Integer// in the solver section
!! and the Mean velocity here:
  Convection Velocity 1 = Variable int Velocity 1, thickness
     REAL MATC "tx(0)/tx(1)"
  Convection Velocity 2 = Variable int Velocity 2, thickness
     REAL MATC "tx(0)/tx(1)"

End
```

```
Boundary Condition 1
! Dirichlet condition only
  H = Real ...
End
```

```
Material 1
!! Limiters
  Min H = Real ....
  Max H = Real ....

End
```

Coupling SSA solver / Thickness solver

SSASolver uses Zs and Zb (H=Zs-Zb)

=> requires an intermediate step between *ThicknessSolver* and *SSASolver*

```
Initial Condition 1
  H = Real ....
End

Body Force 1
! to update Zb and Zs according to H evolution
  Zb = Real ...
  Zs = Variable Zb , H
     REAL MATC "tx(0)+tx(1)"
End

Solver 1
   Equation = "UpdateExport"
   Procedure = "ElmerIceSolvers" "UpdateExport"
   Variable = -nooutput "dumy"

    Exported Variable 1 = -dofs 1 "Zb"
    Exported Variable 2 = -dofs 1 "Zs"
End

Solver 2
  Equation = "SSA"
  Procedure = File "ElmerIceSolvers" "SSABasalSolver"
  Variable = String "SSAVelocity"
  Variable DOFs = 2    ! 1 in SSA 1-D
End

Solver 3
   Equation = "Thickness"
   Variable = -dofs 1 "H"
End
```

you can write a User Function to apply flotation to Zb and Zs=Zb+H

1. From H compute Zb and Zs

look for definition of Exported variables in «Body Force»

2. From Zb and Zs compute u

3. From u compute H

Examples

Friction Laws:

ismip diagnostic test cases

[ELMER_TRUNK]/elmerice/Tests/SSA_Coulomb
[ELMER_TRUNK]/elmerice/Tests/SSA_Weertman

Coupling SSA/Thickness:

[ELMER_TRUNK]/elmerice/Tests/SSA_IceSheet

[ELMER_TRUNK]/elmerice/examples/Test_SSA → ismip prognostic test:

- 1D (2D mesh)
- 2D (2D mesh)
- 2D (3D mesh; use *StructuredProjectToPlane* to compute mean values))

Coupling Stokes/Thickness:

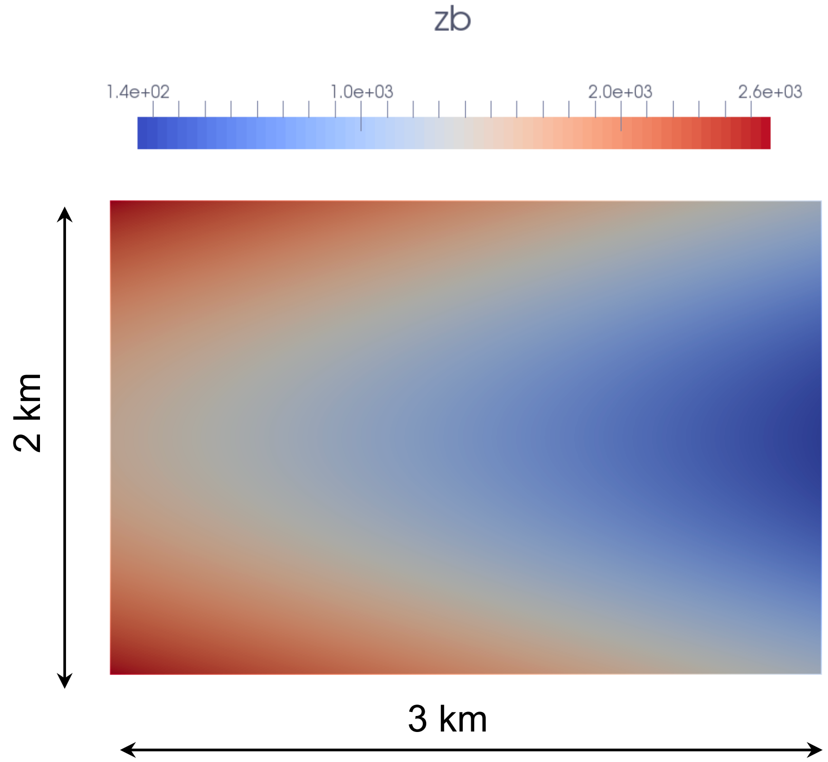
ismip prognostic test:

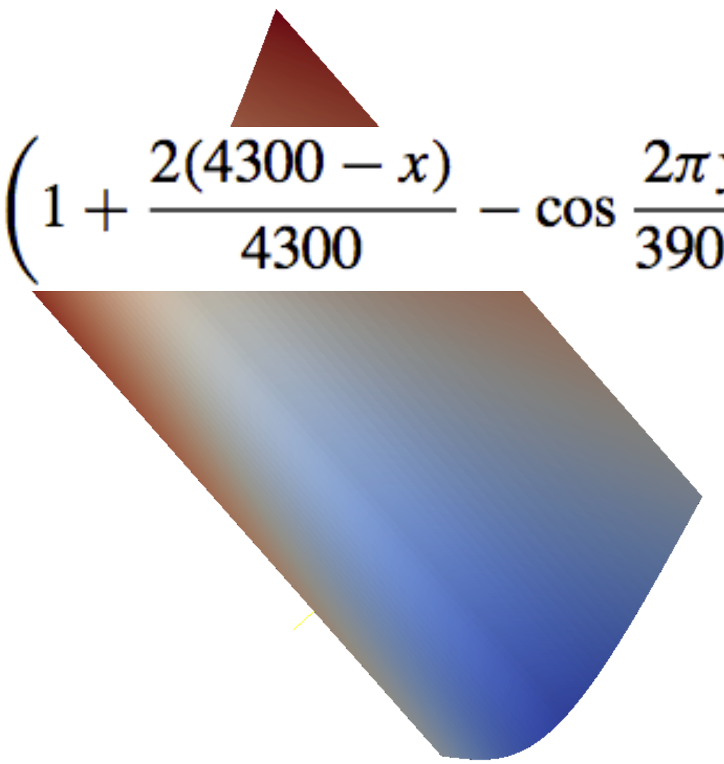
*[ELMER_TRUNK]/elmerice/*Tests/ThicknessSolver

Outline

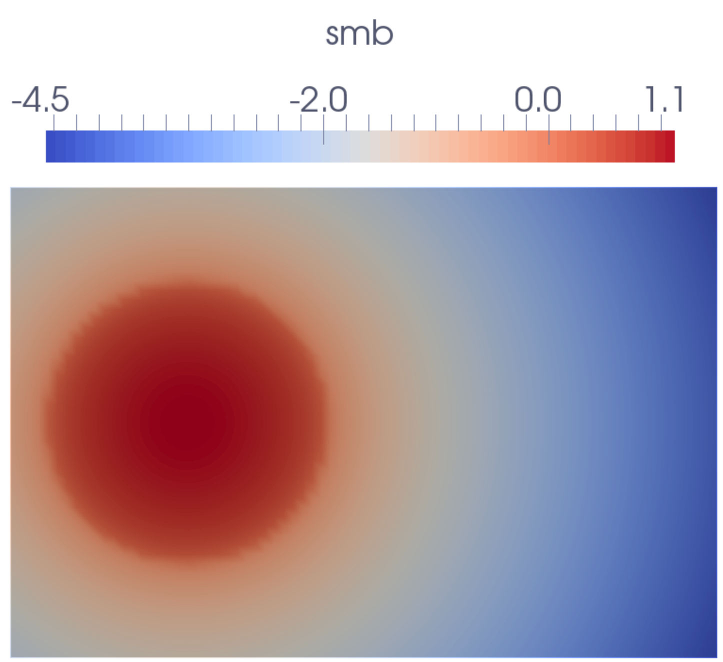
- ✓ Shallow Shelf / Shallow stream Solver
- ✓ Thickness Solver
- ✓ **A glacier example**

Glacier geometry, SMB and initial conditions





$$B(x,y) = 1000\left(1 + \frac{2(4300-x)}{4300} - \cos\frac{2\pi y}{3900}\right)$$



$$a(x,y) = a_0 \frac{|R_a^2 - R^2|}{R_a^2 - R^2} \times \frac{\sqrt{|R_a^2 - R^2|}}{R_a}$$

$$R^2 = (1750 - x)^2 + y^2$$

$$R_a = 600\ \mathrm{m} \qquad a_0 = 1.0\ \mathrm{m\,w.e.\,a^{-1}}$$

From Le Meur et al., 2004

We will start from an ice free domain and let the glacier growths under constant SMB.

User function USF_glacier3d.F90

$$B(x,y) = 1000\left(1 + \frac{2(4300-x)}{4300} - \cos\frac{2\pi y}{3900}\right)$$

```
FUNCTION Bedrock(x,y) RESULT(Zb)
USE types
IMPLICIT NONE
REAL(KIND=dp),INTENT(IN) :: x,y
REAL(KIND=dp) :: Zb

  Zb=1000._dp*(1._dp+2._dp*(4300._dp-x)/4300._dp-cos(2*Pi*y/3900._dp))

END FUNCTION Bedrock

FUNCTION Bed ( Model, nodenumber, VarIn) RESULT(VarOut)
USE types
IMPLICIT NONE
TYPE(Model_t) :: Model
INTEGER :: nodenumber
REAL(KIND=dp) :: VarIn
REAL(KIND=dp) :: VarOut

REAL(KIND=dp) :: Bedrock
REAL(KIND=dp) :: x,y

  x = Model % Nodes % x (nodenumber)
  y = Model % Nodes % y (nodenumber)

  VarOut=Bedrock(x,y)

END FUNCTION Bed
```

```
FUNCTION smb ( Model, nodenumber, VarIn) RESULT(VarOut)
USE types
IMPLICIT NONE
TYPE(Model_t) :: Model
INTEGER :: nodenumber
REAL(KIND=dp) :: VarIn
REAL(KIND=dp) :: VarOut

REAL(KIND=dp) :: Bedrock
REAL(KIND=dp) :: x,y,R2
REAL(KIND=dp),parameter :: a0=1.0/0.890, Ra=600._dp

  x = Model % Nodes % x (nodenumber)
  y = Model % Nodes % y (nodenumber)

  R2=(1750.-x)**2.+y**2.

  VarOut=0._dp
  IF (abs(Ra*Ra-R2).GT.0.) THEN
    VarOut=a0
    VarOut=VarOut*abs(Ra*Ra-R2)/(Ra*Ra-R2)
    VarOut=VarOut*sqrt(abs(Ra*Ra-R2))/Ra
  END IF

END FUNCTION smb
```

$$a(x,y) = a_0 \frac{|R_a^2 - R^2|}{R_a^2 - R^2} \times \frac{\sqrt{|R_a^2 - R^2|}}{R_a}$$

$$R^2 = (1750-x)^2 + y^2$$

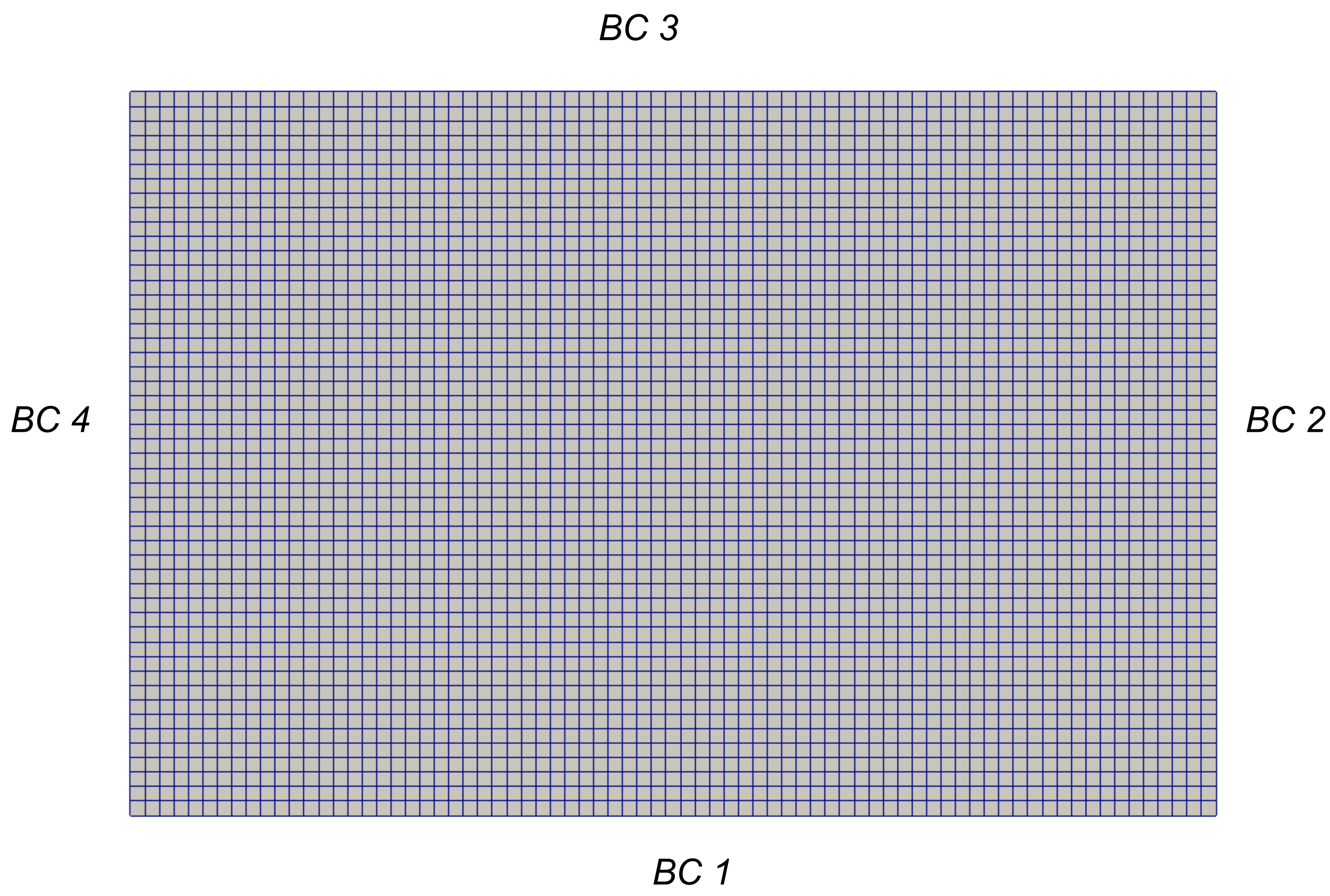
$$R_a = 600\ \mathrm{m} \qquad a_0 = 1.0\ \mathrm{m\,w.e.\,a^{-1}}$$

From Le Meur et al., 2004

Make the mesh

We use a grd input file to make a rectangular mesh of size [1000,4000] x [-1000,1000] of 75 x 50 rectangular elements

```
Coordinate System = Cartesian 2D
Subcell Divisions in 2D = 1 1
Subcell Limits 1 = 1000.0  4000.0
Subcell Limits 2 = -1000.0 1000.0
Material Structure in 2D
  1
End
Materials Interval = 1 1
Boundary Definitions
! type     out      int
  1       -1        1        1
  1       -2        1        1
  1       -3        1        1
  1       -4        1        1
End
Numbering = Vertical
Coordinate Ratios = 1
Decimals = 12
Element Innernodes = False
Element Degree = 1
Triangles = False
Element Divisions 1 = 75
Element Divisions 2 = 50
```



To create the Elmer mesh:

> ElmerGrid 1 2 glacier.grd

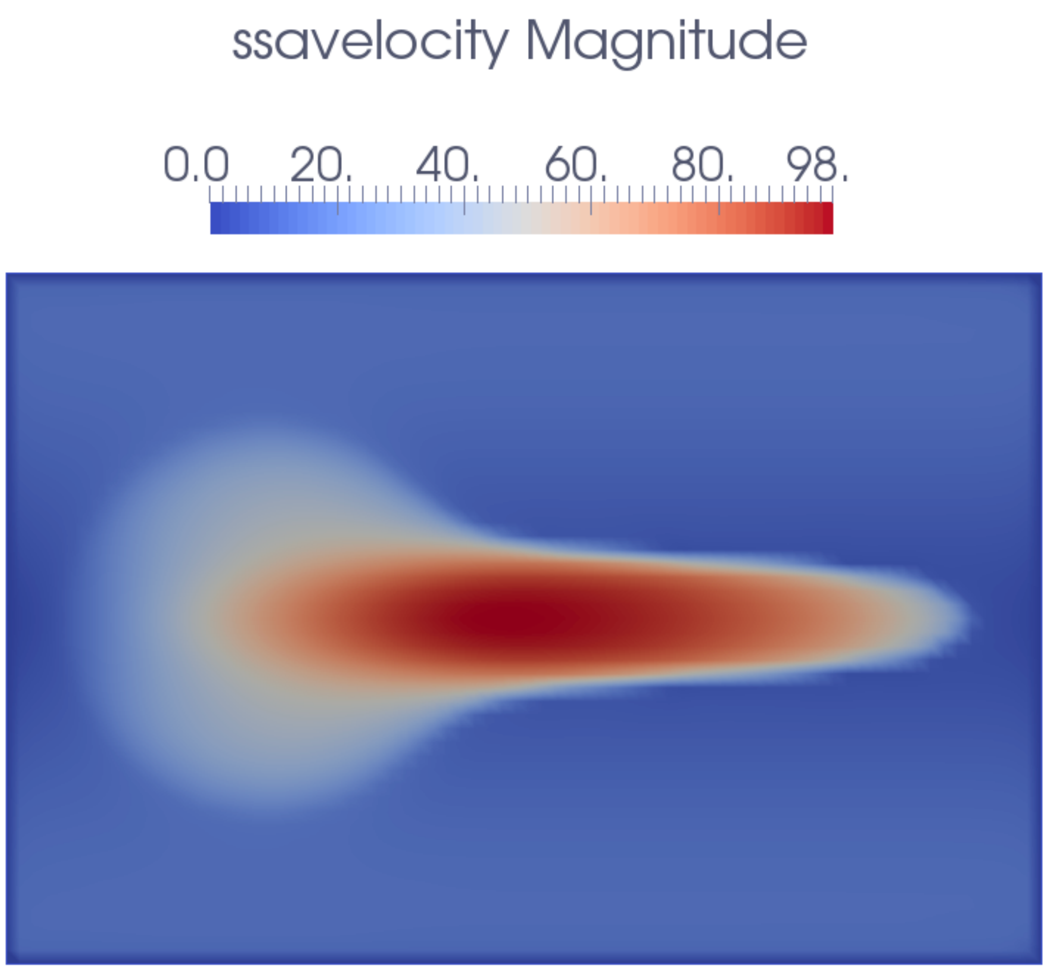
Run the simulation

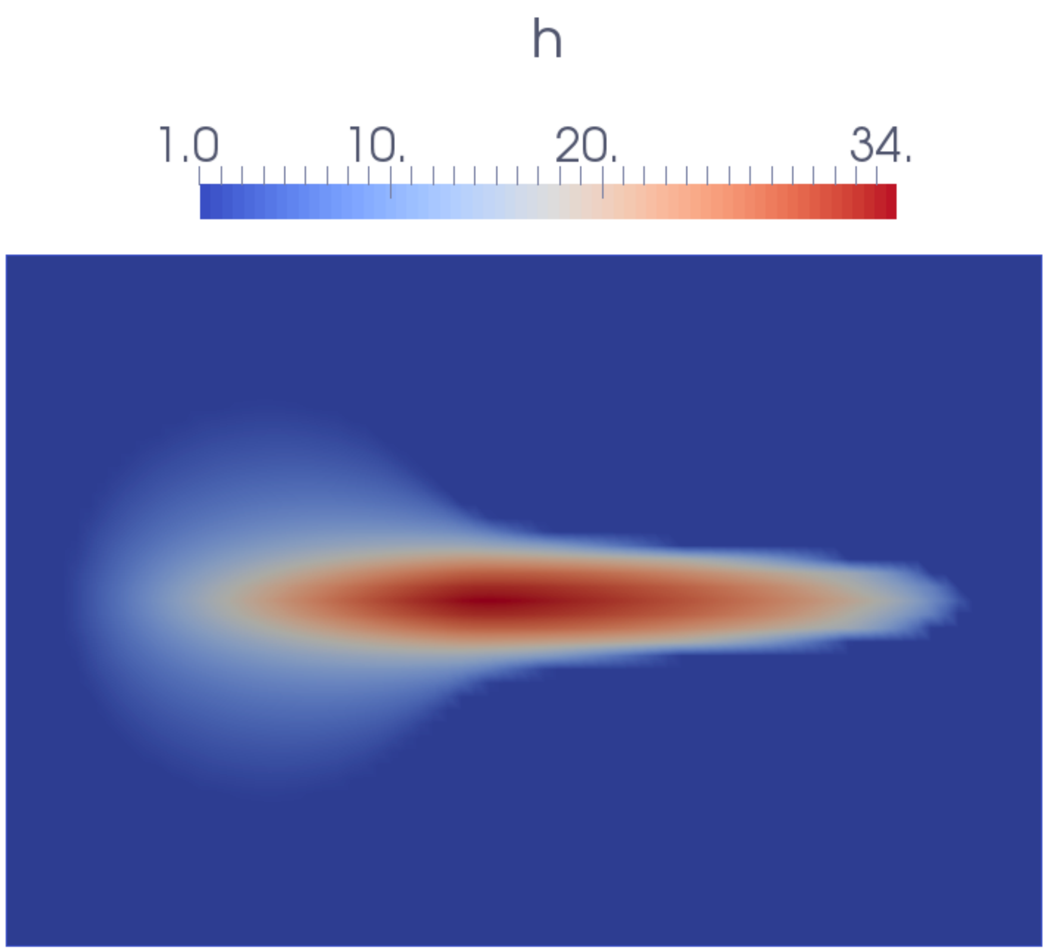
To compile the user function (Makefile):

> *make*

Run the simulation:

> *ElmerSolver glacier3d_SSA.sif*





Play around...

Some ideas ....

- ✓ change the basal friction coefficient, change the form of the friction law
- ✓ start a perturbation run from this steady state (SMB(t) or friction(t))
- ✓ change the bed geometry
- ✓ change the mesh to triangular unstructured mesh
- ✓ have a look in the Stokes directory to run the same problem with Stokes
- ✓ ...