

Elmer/Ice advanced Workshop

30 Nov – 2 Dec 2015

Lower-order Stokes model

Fabien Gillet-Chaulet

LGGE - Grenoble - France

- **Shallow Shelf / Shallow stream Solver**
- **Thickness Solver**
- **Shallow Ice Solver**
- **Current / planned development**

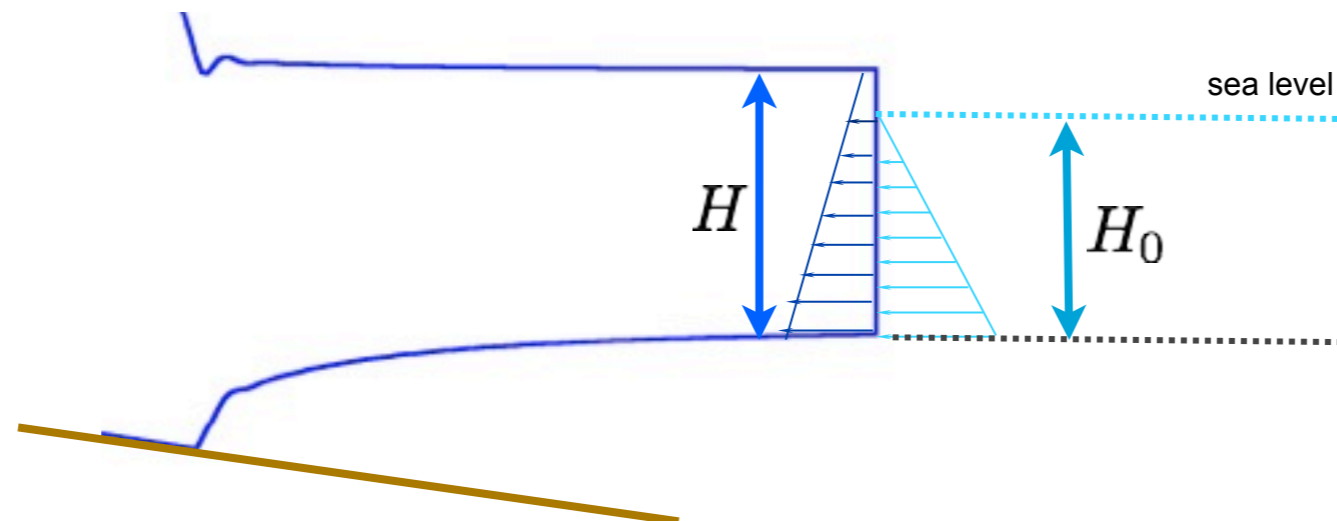
Shallow Shelf Approximation/Shallow Stream Approximation

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:

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



Shallow Shelf Approximation/Shallow Stream Approximation

Field equations:

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

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:

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

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 $rho

! 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 ...
    
```

Friction laws:

- Linear:

$$\tau_b = \beta u$$

- Weertman:

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

- Coulomb:

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

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

End

Shallow Shelf Approximation/Shallow Stream Approximation

Field equations:

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

Boundary Conditions:

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

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

SSA uses mean viscosity and density:

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

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

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
```

=> J. Brondex (LGGE) is working on new solutions for this step and to compute the 3D velocity field
(=> coupling with damage and temperature)

- Shallow Shelf / Shallow stream Solver
- **Thickness Solver**
- Shallow Ice Solver
- Current / planned development

Thickness Solver

Field equations:

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

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 stabilisation** 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 solver for example)

Thickness Solver

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

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 Z_s and Z_b ($H=Z_s-Z_b$)

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

Do it yourself:

```
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 "dummy"

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 Z_b and $Z_s=Z_b+H$

1. From H compute Z_b and Z_s
look for definition of Exported variables in «Body Force»

2. From Z_b and Z_s compute u

3. From u compute H

Coupling SSA solver / Thickness solver

SSASolver uses Z_s and Z_b ($H=Z_s-Z_b$)

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

Do it yourself:

```
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 "dummy"

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
```

I will put a *FlotationSolver* in the distrib soon:

- From H apply Flotation to compute Z_b
- If bedrock is given check if floating or grounded
- compute grounded mask (-1: floating, +1: grounded, 0: groundig line)
- $Z_s = H + Z_b$
- *optionally*: compute dZ_s/dt and dZ_b/dt

Check volume and fluxes using SaveScalars

Solver X

```
Exec Solver = After Timestep
```

```
Equation = "Save Scalars"
```

```
Procedure = File "SaveData" "SaveScalars"
```

```
Filename = File "Scalars_"$name$.dat"
```

```
Variable 1 = "Time"
```

```
! int H = Volume
```

```
Variable 2 = "H"
```

```
Operator 2 = "int"
```

```
! int dh/dt = dVolume/dt
```

```
Variable 3 = "dhdt"
```

```
Operator 3 = "int"
```

```
! int SMB
```

```
Variable 4 = "smb"
```

```
Operator 4 = "int"
```

```
! SMB_H=Artificial additional Mass flux due to limits on H
```

```
Variable 5 = "h residual"
```

```
Operator 5 = "sum"
```

```
! OUT Flow
```

```
Variable 6 = "SSAVelocity"
```

```
Operator 6 = "convective flux"
```

```
Coefficient 6 = "Flux"
```

```
! => Dvolume/dt ~ SMB + SMB_H - OUT
```

```
End
```

Material 1

```
!! For Save scalar to compute mass flux (=H*SSA_UV)
```

```
Flux = Equals H
```

```
End
```

Boundary Condition 1

```
Target Boundaries = 1
```

```
Save Scalars = Logical True
```

```
Calving Front = Logical True
```

```
End
```

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`

- Shallow Shelf / Shallow stream Solver
- Thickness Solver
- **Shallow Ice Solver**
- Current / planned development

Shallow Ice Approximation

Field equations:

$$\begin{aligned}\frac{\partial u}{\partial z} &= -2A(\rho g)^n (S - z)^n \left[\sqrt{\left(\frac{\partial S}{\partial x}\right)^2 + \left(\frac{\partial S}{\partial y}\right)^2} \right]^{n-1} \frac{\partial S}{\partial x}, \\ &= -(\rho g / \eta)^n (S - z)^n \left[\sqrt{\left(\frac{\partial S}{\partial x}\right)^2 + \left(\frac{\partial S}{\partial y}\right)^2} \right]^{n-1} \frac{\partial S}{\partial x},\end{aligned}$$

$$\begin{aligned}\frac{\partial v}{\partial z} &= -2A(\rho g)^n (S - z)^n \left[\sqrt{\left(\frac{\partial S}{\partial x}\right)^2 + \left(\frac{\partial S}{\partial y}\right)^2} \right]^{n-1} \frac{\partial S}{\partial y}, \\ &= -(\rho g / \eta)^n (S - z)^n \left[\sqrt{\left(\frac{\partial S}{\partial x}\right)^2 + \left(\frac{\partial S}{\partial y}\right)^2} \right]^{n-1} \frac{\partial S}{\partial y},\end{aligned}$$

$$\frac{\partial w}{\partial z} = -\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y},$$

$$\frac{\partial p}{\partial z} = -\rho g,$$

These equations can be seen as degenerated Poisson equations:

$$\frac{\partial^2 U}{\partial z^2} = \Psi(x, y, z), \quad \text{with boundary conditions} \quad \frac{\partial U}{\partial z} = \Gamma(x, y) \text{ for } z = \Omega_f, \text{ and } U = \bar{U} \text{ for } z = \Omega_u. \quad (1)$$

The SIA Solver in the *Elmer/Ice* distrib solves 4 times Eq. (1) for the 4 unknowns (u,v,w,p)

=> work with fully unstructured meshes

=> we need to work on efficient solutions for structured meshes

Shallow Ice Approximation

SIF entries:

```
! Dummy solver just here to declare SIAFlow
! as a true variable (not an exported variable)
! to allow access to previous values
Solver 2
Equation = "SIA Variable"
Procedure = File "ElmerIceSolvers" "SIAVariable"
Variable = "SIAFlow"
Variable DOFs = 4 ! 4 in 3D (u,v,w,p), 3 in 2D (u,v,p)
End

Solver 3
Equation = "SIA"
Procedure = File "ElmerIceSolvers" "SIASolver"
Variable = -nooutput "SIAvar"
Variable DOFs = 1

Linear System Solver = "Direct"
Linear System Direct Method = umfpack

Steady State Convergence Tolerance = Real 1.0e-3
End

!!! bedrock
Boundary Condition 5
Target Boundaries = 5
SIAFlow 1 = Real 0.0e0
SIAFlow 2 = Real 0.0e0
SIAFlow 3 = Real 0.0e0
End

!!! free surface
Boundary Condition 6
Target Boundaries = 6
Save Line = Logical True
SIAFlow 4 = real 0.0 !(p=0)
Depth = real 0.0
End
```

An example using the SIASolver applied to experiment A160 of ISMIP-HOM benchmarks can be found in `[ELMER_TRUNK]/elmerice/Tests/SIA`.

- Shallow Shelf / Shallow stream Solver
- Thickness Solver
- Shallow Ice Solver
- **Current / planned development**

Current/planned developments

Inverse methods:

- AdjointSolver for SSA => constrain friction, mean viscosity, Z_b , Z_s from observation
 - *Fürst et al.*, Assimilation of Antarctic velocity observations provides evidence for uncharted pinning points, *The Cryosphere*, 2015
 - *Fürst et al.*, Passive shelf ice: the safety band of Antarctic ice shelves, *Nature Climate Change*, accepted
- AdjointSolver for Thickness => constrain u , smb from observations of H
(see Morlighem *et al.*, 2011, a mass conservation approach for mapping glacier ice thickness)

SSA*:

- modify viscosity to take into account vertical shearing
(see Cornford *et al.*, 2013, adaptive mesh, finite volume modeling of marine ice sheets)

Efficient hybrid model SSA+SIA

Efficient coupling with Temperature and Damage

Current/planned developments

Adaptative mesh refinement around the grounding line:

- serial mesh-splitting strategy already implemented

