Elmer/Ice course

22-23 April 2013 – Edmonton

Olivier GAGLIARDINI [1]

Introduction

(1) LGGE - Grenoble - France

Program

Day 1 : April 22 2013 (9:00am-5:00pm)

- A brief introduction of the main tools
  - What is Elmer/Ice?
  - How to get a mesh
  - Solver input file
  - How to visualise results

- A step by step exercise using ISMIP tests B and C
  2D flow line applications, diagnostic and prognostic

Day 2: April 23 2013 (8:30am-3:00pm)

- Short presentation of what you would like to do with Elmer/Ice in the next future (5mn)
- A real world application: Tête Rousse glacier
  3D application, diagnostic and prognostic

Short history of Elmer/Ice

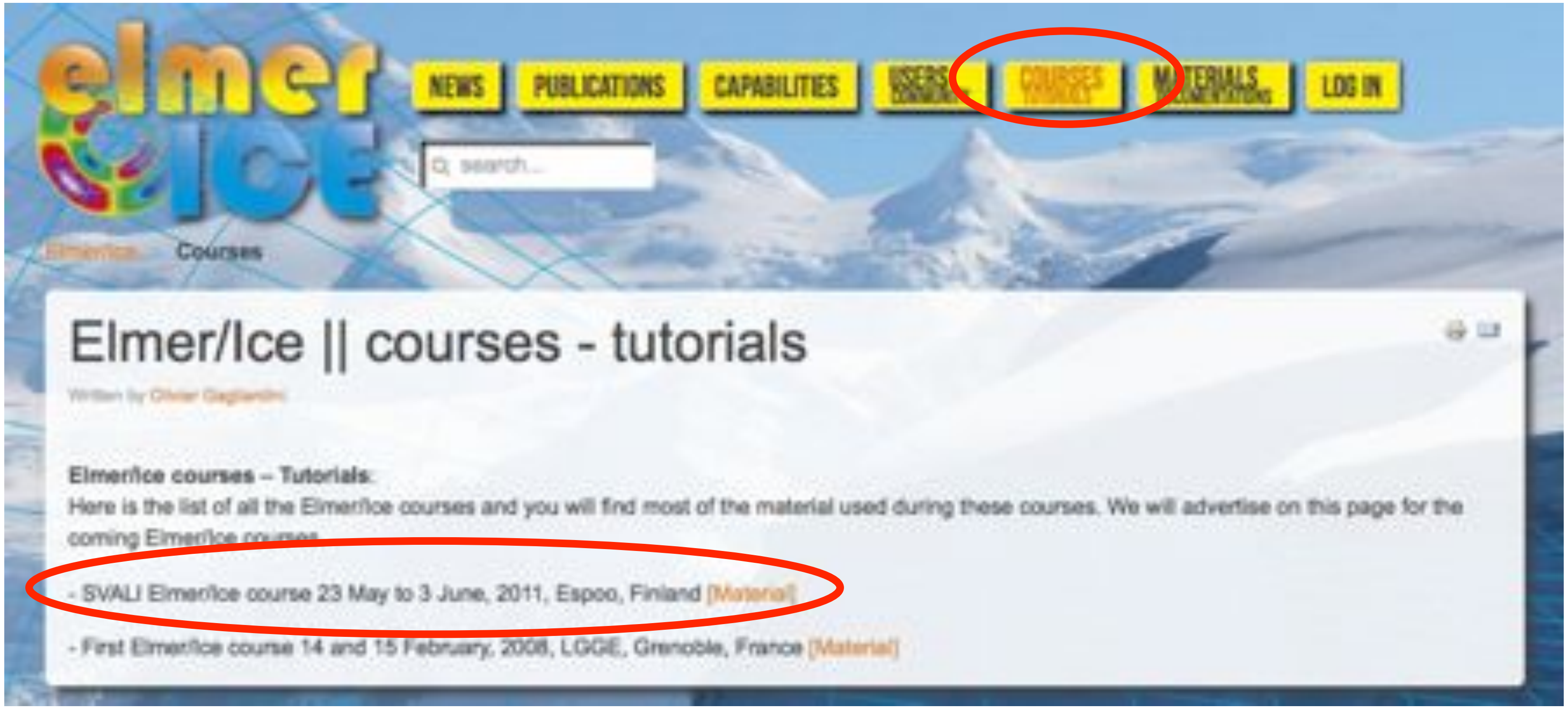
- ✓ EGU2002: OG was looking for a 3D FE code to model the flow of strain-induced anisotropic polar ice – meet TZ
- ✓
- ✓ March 2003: OG visited CSC for few days: AIFlowSolver and FabricSolver partly implemented
- ✓ August 2005 – One year visit of OG at CSC (Anisotropy, cavity, glaciers, ISMIP tests, ...)
- ✓ February 2008 – First Elmer/Ice Course - Grenoble
- ✓ June 2011 – Second Elmer/Ice Course – Finland
- ✓ 2012 – Elmer/Ice has now a website, a logo and a mailing list
- ✓ 2012 – Elmer/Ice comes as a Elmer Package – New wiki
- ✓ 2012 – Elmer/Ice course at UBC/SFU
- ✓ 2013 – Elmer/Ice courses at Univ. Washington and Univ. Alberta
- ✓ 9 April 2013 – First Elmer/Ice users meeting

Elmer/Ice website

http://elmerice.elmerfem.org/

Elmer/Ice website

http://elmerice.elmerfem.org/



Much more material available than what I will present today

Elmer/Ice wiki http://elmerice.elmerfem.org/wiki/doku.php


[[start]]

ELMER/ICE WIKI

Trace: • start

Show pagesource

Recent changes Sitemap Login

Search

- Home
- Problems
- Solvers
- User Functions
- Meshing Tools
- Tips and Tricks
- Meetings and Courses
- Compilation of Elmer/Ice
- Links

Welcome to the Elmer/Ice wiki

Elmer is an open-source, parallel, Finite Element code, mainly developed by the CSC-IT Center for Science Ltd. in Finland. Elmer/Ice builds on Elmer and includes developments related to glaciological problems. On this page you will find documentation and examples of the various solvers and user functions developed for **glaciological** applications using Elmer/Ice. Building Elmer/Ice on top of an existing Elmer installation is explained in the Compilation Section.

The Problems Section presents the various categories of glaciological problems that can be solved using Elmer/Ice.

The Solvers Section and the User Functions Section describe the glaciology related solvers and user functions, respectively, that can be used to solve these problems.

Tools that can be used to mesh glacier and ice-sheet geometry are presented in the Meshing Section.

The Tips and Tricks Section gives some useful demo of MATC, Post-treatments of results and more.

The Courses Material Section contains presentation as well as material proposed in the framework of the Elmer/Ice courses dispensed since 2008.

Some useful links are given in the Links Section.

Scientific publications presenting glaciological applications with Elmer/Ice are listed in the Elmer/Ice website.

start.txt · Last modified: 2012/12/03 17:45 by tzwinger

Elmer/Ice mailing list

To subscribe to the Elmer/Ice list *elmerice@elmerfem.org*, just sent an email to *majordomo@elmerfem.org*, with in the body the text:

subscribe elmerice

If you do not know how to use mailing lists run by majordomo you may sent a mail with "help" in the message body.

Elmer/Ice versus Elmer

Elmer is an open-source, parallel, Finite Element code, mainly developed by the CSC-IT Center for Science Ltd. in Finland.

Elmer/Ice builds on Elmer and includes developments related to glaciological problems.

Elmer/Ice includes a variety of dedicated solvers and user functions for glaciological applications.

Elmer/Ice Package

All the Solvers, User Functions and Meshers presented on the Elmer/Ice wiki comes as an Elmer/Ice package on the Elmer distribution (in `elmerfem/elmerice`)

To compile the package, go in `elmerice` directory

```
$ make compile
$ make install
```

To use it (in the SIF file):

```
Procedure = File "ElmerIceSolvers" "NameSolver”
```

or

```
Procedure = File "ElmerIceUSF" "NameUSF”
```

Important links

Elmer at CSC (documentation, how to install, …)
http://www.elmerfem.org/
http://www.csc.fi/english/pages/elmer

Elmer Forum
http://elmerfem.org/forum/

Elmer/Ice webpage
http://elmerice.elmerfem.org/

Elmer/Ice wiki
http://elmerice.elmerfem.org/wiki/doku.php?id=start

Important notices

In this course

- I will not teach finite element method (can give references)

- I will focus on some technical aspects of using Elmer for glaciological applications

What I expect from this course ?

- some fruitful collaborations to begin !

Elmer/Ice capabilities

- Full-Stokes equation but also SIA, SSA, Diagnostic or transient
- Various rheology (Glen's law, firn/snow and two anisotropic flow laws)
- Temperature solver accounting for the upper limit at melting point
- Evolution equations for density, fabric, …
- Dating, evaluation of strain-rate and stress fields
- Various friction laws (Weertman, effective-pressure dependent friction law)
- Grounding line dynamics as a contact problem
- Inverse methods (linear adjoint and Arthern and Gudmundsson 2010 methods)
- Tools to mesh glaciers (YAMS, extrusion of footprint)
- Highly parallel Stokes solver

Elmer/Ice applications

More than 30 publications using Elmer/Ice since 2004

- ISMIP, MISMIP, MISMIP-3d
- 2D and 3D Grounding line dynamics
- Ice2sea and SeaRISE contributions (Greenland)
- Inverse methods (Variegated, Vestfonna ice-cap, GIS)
- Flow of anisotropic ice

see http://elmerice.elmerfem.org/publications

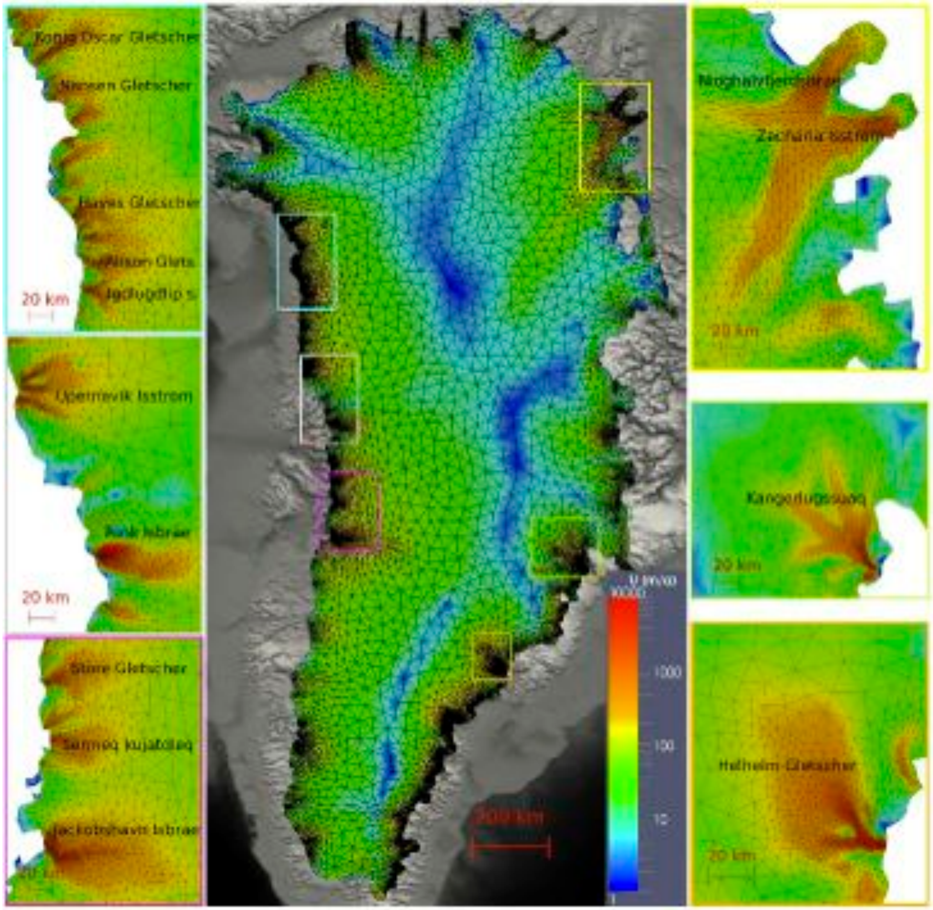
GMD paper

Capabilities and performance of Elmer/Ice, a new generation ice-sheet model

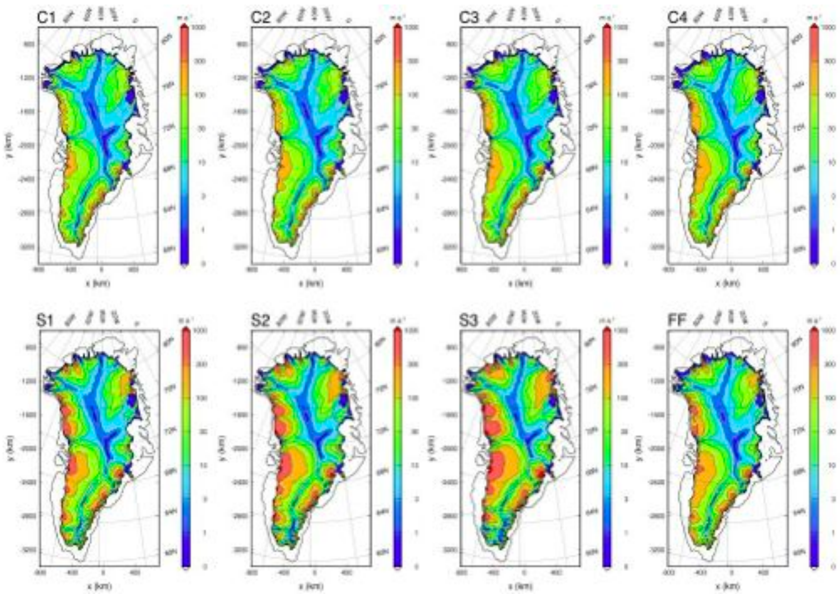
O. Gagliardini[1,2], T. Zwinger[3], F. Gillet-Chaulet[1], G. Durand[1], L. Favier[1], B. de Fleurian[1], R. Greve[4], M. Malinen[3], C. Martín[5], P. Råback[3], J. Ruokolainen[3], M. Sacchettini[1], M. Schäfer[6], H. Seddik[4], and J. Thies[7]

Few recent examples

Grenland within ice2sea
@Fabien Gillet-Chaulet, LGGE



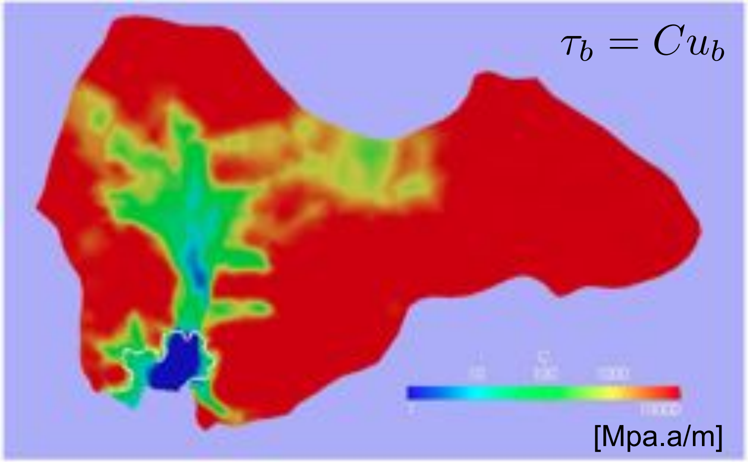
Grenland within SeaRise
@Hakime Seddik, ILTS



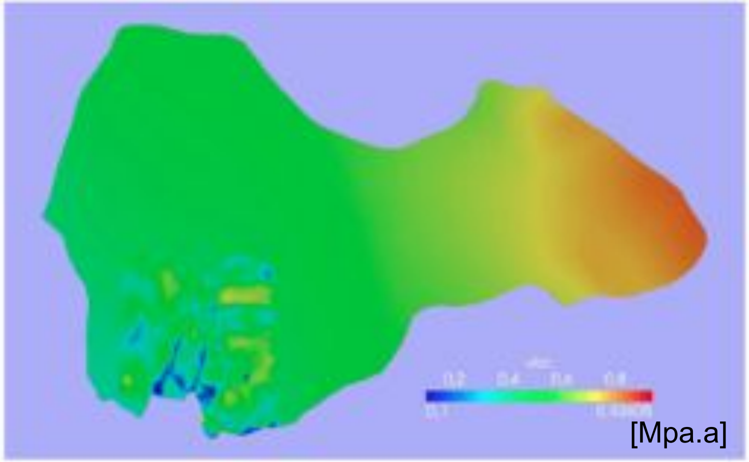
Few recent examples

Grounding line 3D   @Lionel Favier, LGGE

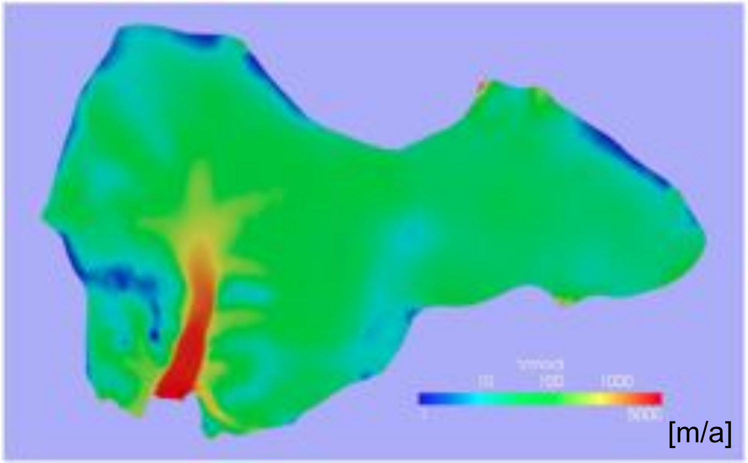
Inverted **basal friction** parameter



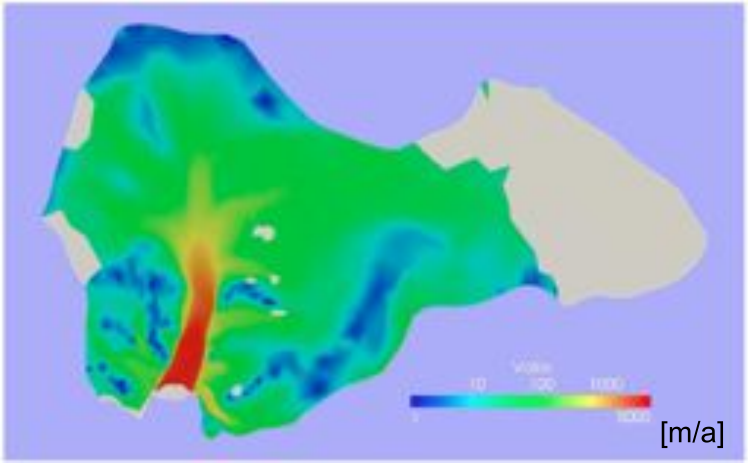
Inverted surface **effective viscosity**



Inverted surface velocity

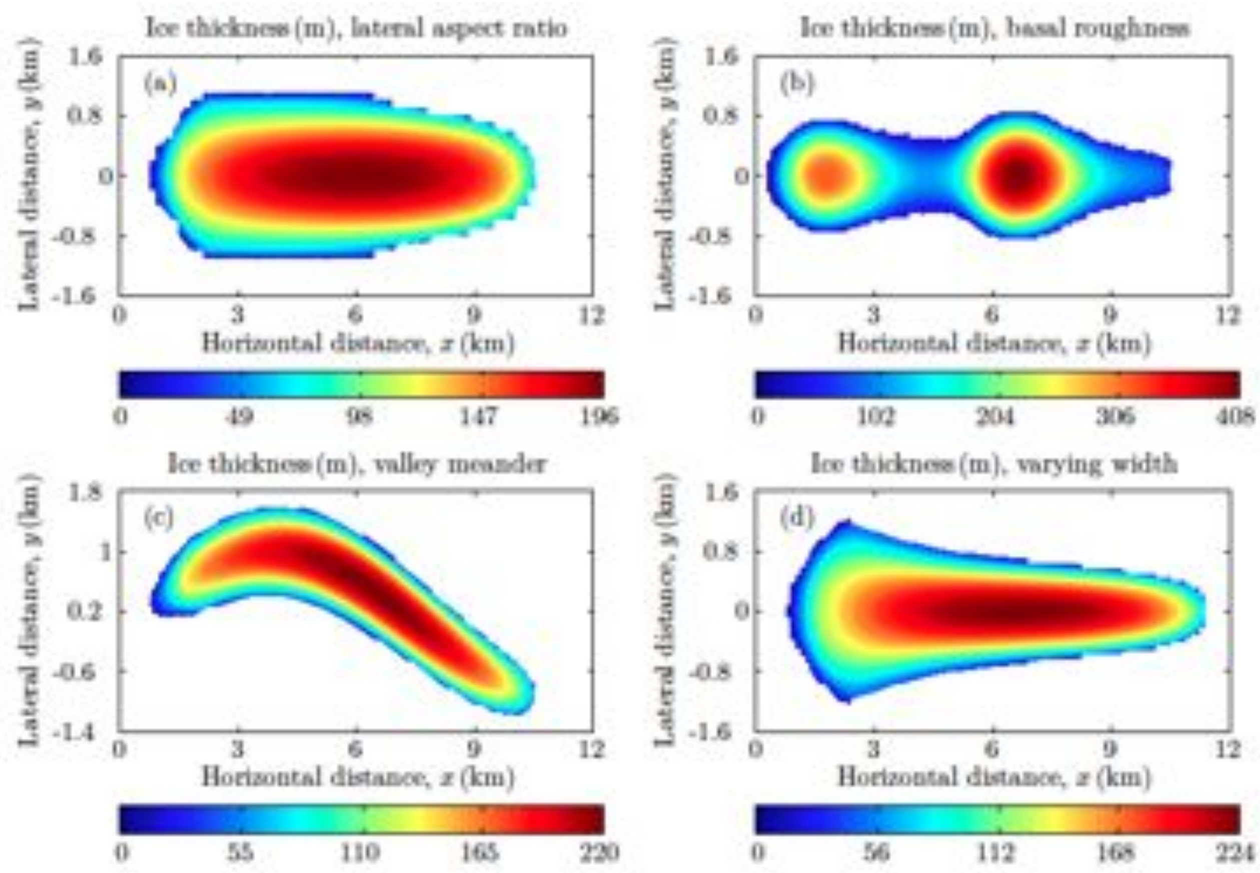


Observed surface velocity (Rignot et al., 2011)



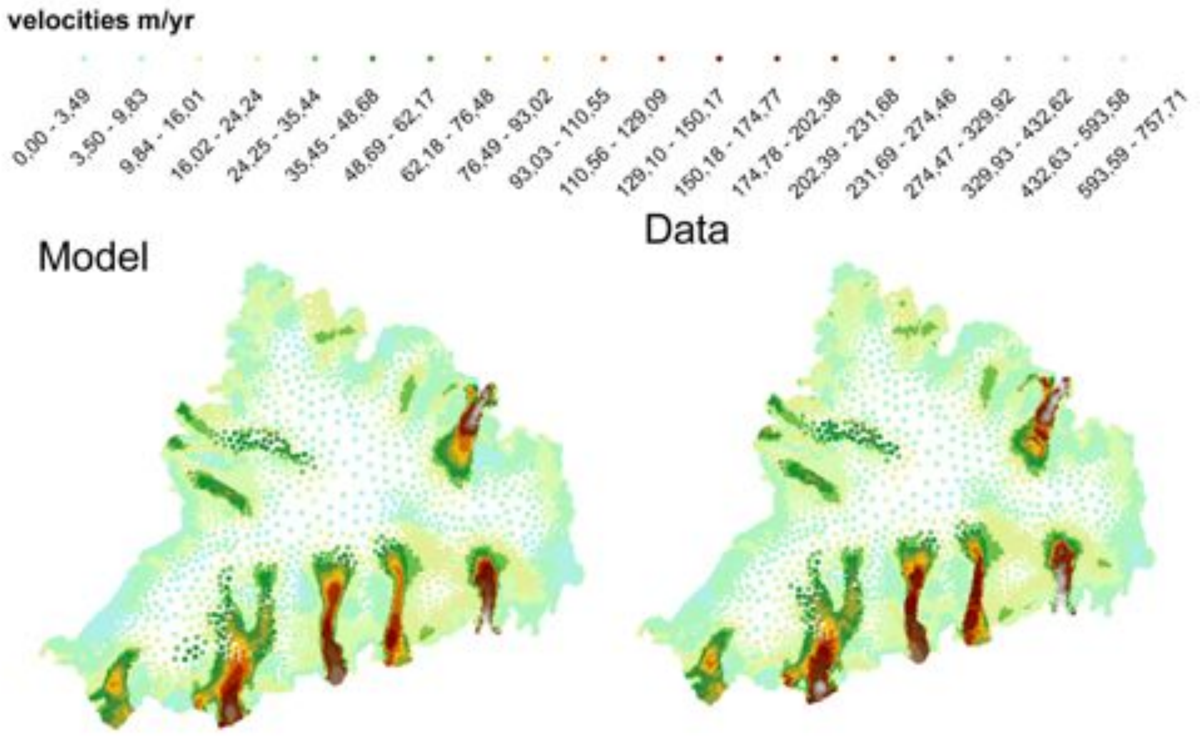
Few recent examples

Volume/Area relation @Surendra Adhikari, Univ. Calgary



Few recent examples

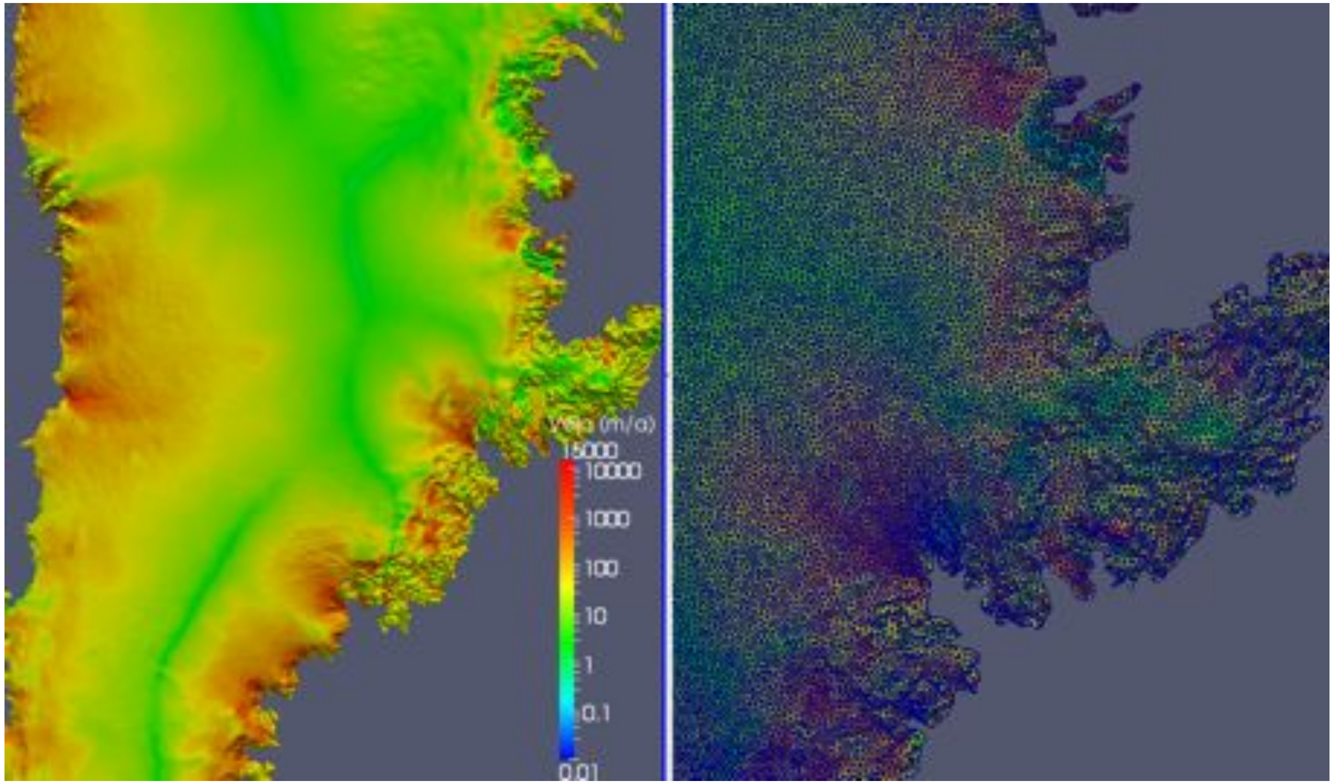
Vestfonna ice cap basal friction @Martina Schäfer, Univ. Lapland



Few recent examples

High parallel computing @Fabien Gillet-Chaulet, LGGE

1 900 000 nodes on 400 partitions
~7 000 000 dofs

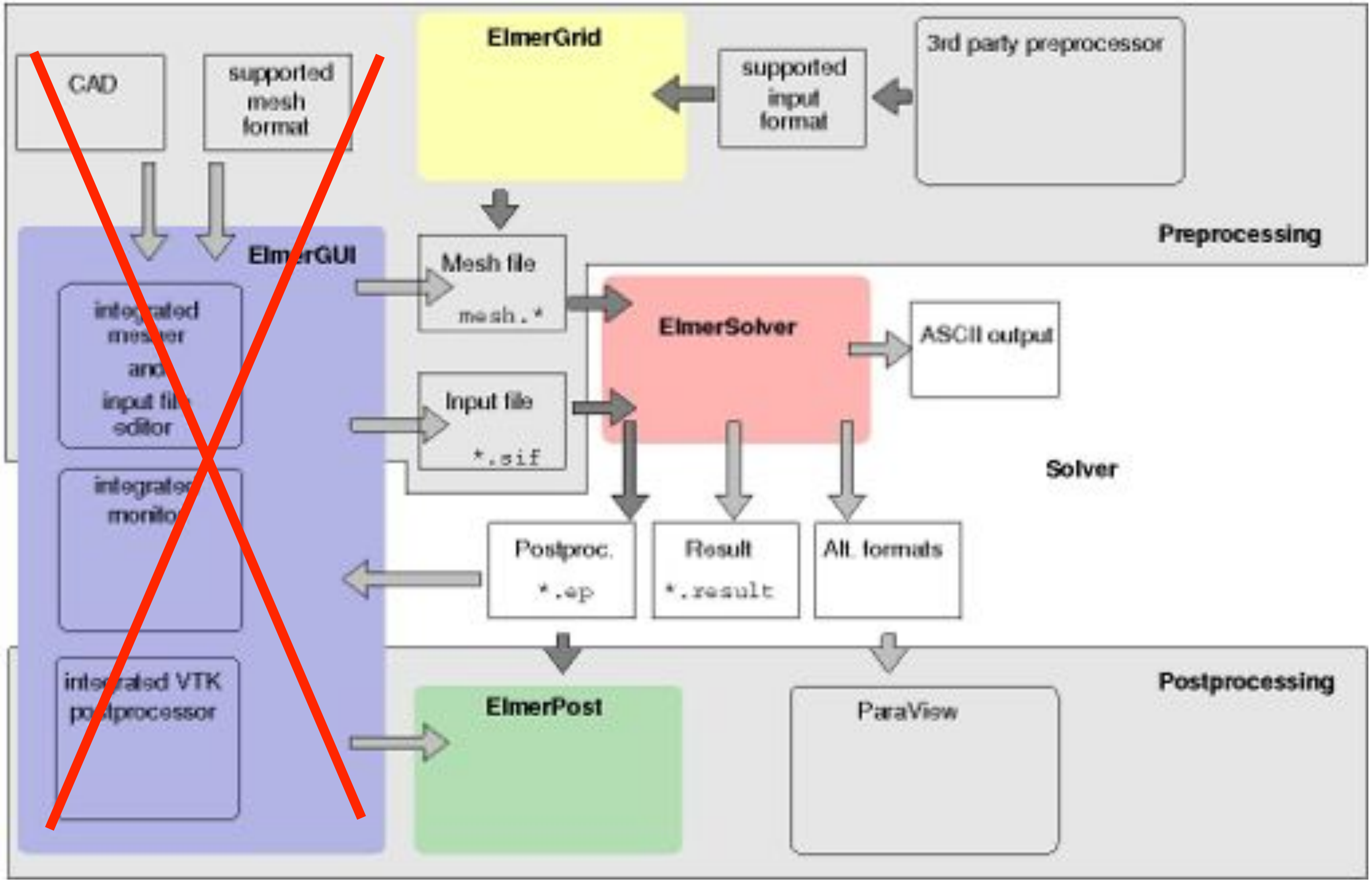


Current or planned developments

- **Calving law (damage mechanics)**
- **Hydrology model to infer basal water pressure**
- Moving margins / remeshing / adaptive mesh
- Coupling with an ocean model / Implementation of a plume model
- Accounting for refreezing in the temperature equations
- Inversion of bedrock topography
- Lower order Stokes models

How does it work ?

Elmer structure



Sequence of a simulation

- file in a solver input file (`mysif.sif`)

- build a mesh in Elmer format, i.e. a directory containing
  `mesh.header, mesh.nodes, mesh.element, mesh.boundary`

- compile object files linked with Elmer of your user functions and solvers (if needed)

- Execute :

```
$ ElmerSolver mysif.sif
```

- Should create a *.ep file (ElmerPost format)

- Visualise :

```
$ ElmerPost
```

We will see

- how to construct a simple mesh
- what is the contains of a sif file
- how to execute
- how to visualise the results

How to get a mesh ?

Different possibilities to get a mesh

- use ElmerGrid alone

- use an other mesher (gmsh, gambit) and then transform it in Elmer format (ElmerGrid can do this for many other mesher formats)

- Glacier particularities :
  - Small aspect ratio (horizontally elongated elements)
  - In 3D, mesh a footprint with an unstructured mesh, and then vertically extrude it (same number of layer everywhere)

will see this later during the course…

ElmerGrid

- command line tool for mesh generation
- native mesh format: `.grd`
- help : just execute : `ElmerGrid`
- possible to import meshes produced by other free or commercial mesh generators (Ansys, Abaqus, Gambit, Comsol, gmsh, …)

Examples :

```
$ ElmerGrid 1 2 my_mesh.grd
$ ElmerGrid 14 2 my_gmsh_mesh.msh
$ ElmerGrid 14 3 my_gmsh_mesh.msh
```

Solver Input File (sif)

Example of sif file

- Comments start with !
- Not case sensitive
- Do not use tabulators for indents

- A section always ends with the keyword `End` or use `::`

- Parameters need to be casted by types:
  `Integer`, `Real`, `Logical`, `String` and `File`

- `Paremetername(n,m)` indicates a n×m array

- Sections are
```
Header
Constants
Simulation
Solver i
Body i
Equation i
Body Force i
Material i
Initial Condition i
Boundary Condition i
```

```
Body Force 1
Heat Source = 1.0
End
```

OR

```
Body Force 1 :: Heat Source = 1.0
```

Example of sif file

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!                                        !!
!! Elmer/Ice Course - Application Step0   !!
!!                                        !!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Updated May 2011

check keywords warn
echo on

Header
  Mesh DB "." "square"
End

Constants
! No constant needed
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Simulation
  Coordinate System  = Cartesian 2D
  Simulation Type = Steady State

  Steady State Min Iterations = 1
  Steady State Max Iterations = 1

  Output File = "ismip_step0.result"
  Post File = "ismip_step0.ep"
  max output level = 100
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Body 1
  Equation = 1
  Body Force = 1
  Material = 1
  Initial Condition = 1
End

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

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Body Force 1
  Flow BodyForce 1 = Real 0.0
  Flow BodyForce 2 = Real -1.0
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

- **Header** declares where to search for the mesh
- If any **constants** needed (i.e. Gas constant)
- **Simulation**
  - Type of coordinate system
  - Steady or Transient
  - Output files (to restart a run) and ElmerPost file
  - Out put level : how verbose is the code
- In **Body** are assigned the Equation, Body Force, Material and Initial Condition
- In **Initial Condition** sets initial variable values
- In **Body Force** specify the body force entering the right side of the solved equation

Example of sif file

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Material 1
  Density = Real 1.0

  Viscosity Model = String "power law"
  Viscosity = Real 1.0
  Viscosity Exponent = Real 0.33333333333333333
  Critical Shear Rate = Real 1.0e-10
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Solver 1
  Equation = "Navier-Stokes"

  Stabilization Method = String Bubbles
  Flow Model = String Stokes

  Linear System Solver = Direct
  Linear System Direct Method = umfpack

  Nonlinear System Max Iterations = 100
  Nonlinear System Convergence Tolerance  = 1.0e-5
  Nonlinear System Newton After Iterations = 5
  Nonlinear System Newton After Tolerance = 1.0e-02
  Nonlinear System Relaxation Factor = 1.00

  Steady State Convergence Tolerance = Real 1.0e-3
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Equation 1
  Active Solvers(1)= 1
End

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Boundary Condition 1
  Target Boundaries = 1
  Velocity 2 = Real 0.0e0
End

Boundary Condition 2
  Target Boundaries = 4
  Velocity 1 = Real 0.0e0
End

Boundary Condition 3
  Target Coordinates(1,2) = Real 0.0 1.0
  Target Coordinates Eps  = Real 1.0e-3
  Pressure = Real 0.0e0
End
```

- In **Material** sets material properties for the body (can be scalars or tensors, and can be given as dependent functions)
- In **Solver** specifies the numerical treatment for these equations (methods, criteria of convergence,…)
- In **Equation** sets the active solvers
- **Boundary Condition**
  - Dirichlet: `Variablename = Value`
  - Neumann: special keyword depending on the solver
  - Values can be given as function

Variable defined as a function

1/ Tables can be use to define a piecewise linear dependency of a variable

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

2/ MATC: a library for the numerical evaluation of mathematical expressions

```
Density = Variable Temperature
MATC "1000*(1-1.0e-4*(tx-273))"

Viscosity Exponent = Real $1.0/3.0
```

3/ Build your own user function

```
Density = Variable Temperature
Procedure "filename" "proc”
```

`filename` should contain a shareable (.so on Unix) code for the user function whose name is `proc`

Example of User Function

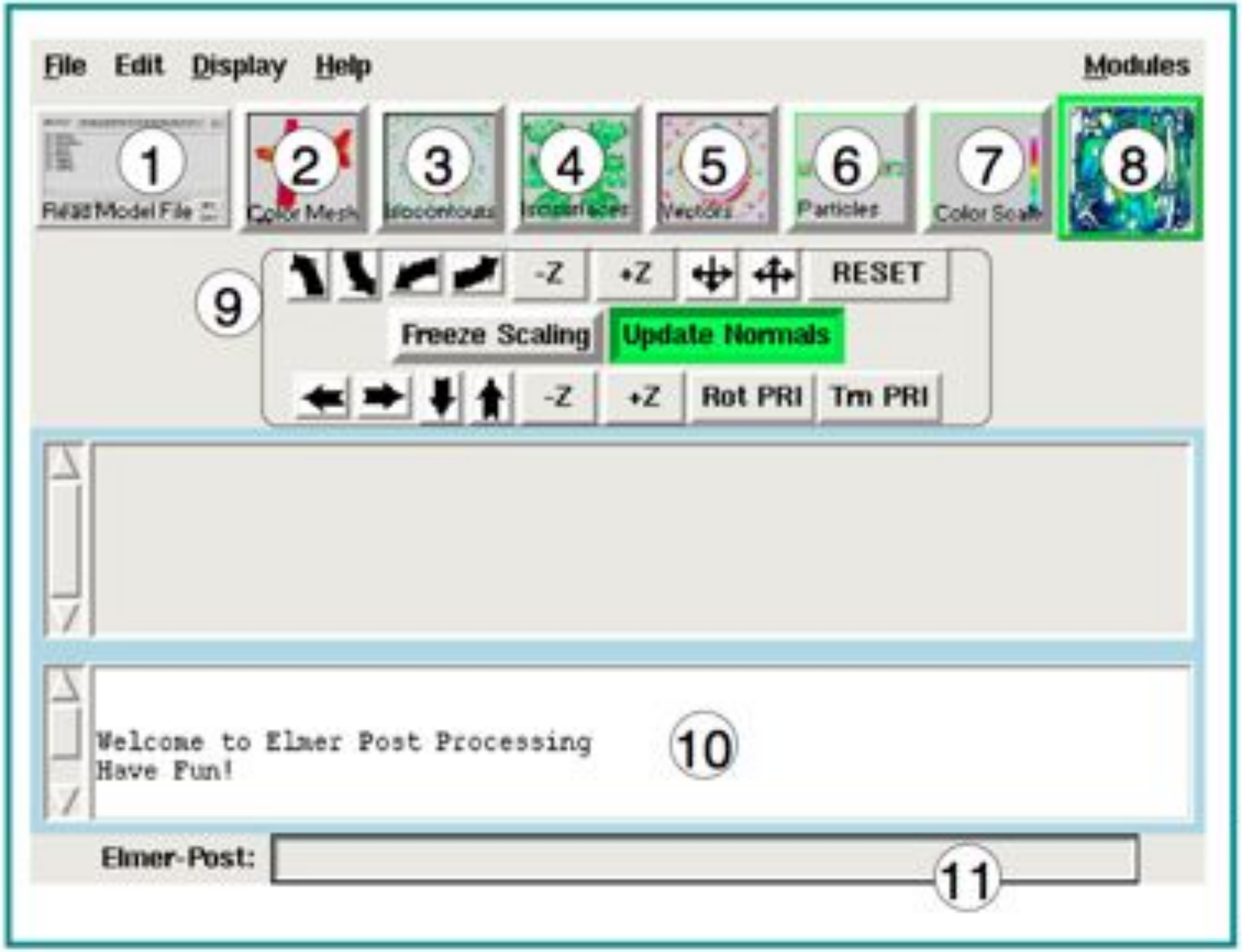
```
FUNCTION proc( Model, n, T ) RESULT(dens)
USE DefUtils
IMPLICIT None
TYPE(Model_t) :: Model
INTEGER :: n
REAL(KIND=dp) :: T, dens

dens = 1000*(1-1.0d-4(T-273.0_dp))
END FUNCTION proc
```

Compilation tools: `elmerf90`

```
$ elmerf90 filename.f90 -o filename
```

How to visualise results

ElmerPost



1. Read result
2. Mesh display
3. Iso-contours
4. Iso-surfaces
5. Vector-field
6. Particles
7. Color-bar
8. Refresh
9. View settings
10. Output
11. Command

Output for other post-processors

	`GID`	GID
	`Gmsh`	Gmsh
`Output Format =`	`Vtk`	VTK legacy
	`Dx Format`	Open DX
	`vtu`	ParaView

```
Solver 1
   Equation = "ResultOutput"
   Procedure = "ResultOutputSolve" "ResultOutputSolver"
   Output File Name = "test"
   Output Format = string "vtu"
   Scalar Field 1 = String "Temperature"
   Vector Field 1 = String "Velocity"
End
```

ASCII Based Output

`SaveScalars`	cpu time, mean, max, min of a variable
`SaveLine`	save a variable along a line (boundary or a given line)
`SaveMaterials`	save a material parameter like a variable

Example:

```
Solver 3
  Exec Solver =  After All
  Procedure = File "SaveData" "SaveLine"
  Filename =  "ismip_surface.dat"
  File Append = Logical False
End


Solver 4
  Exec Solver =  After TimeStep   ! For transient simualtion
  Procedure = File "./MySaveData" "SaveScalars"
  Filename =  "ismip_scalars.dat"
  File Append = Logical True      ! For transient simualtion

  Variable 1 = String "Flow Solution"
  Operator 1 = String "Volume"

  Variable 2 = String "Velocity 1"
  Operator 2 = String "Max Abs"

  Variable 3 = String "Flow Solution"
  Operator 3 = String "Convective flux"

  Variable 4 = String "cpu time"

  Variable 5 = String "cpu memory"
End
```

```
! Upper Surface
Boundary Condition 3
  Target Boundaries = 3
  Save Line = Logical True
  Flux integrate = Logical True
End
```