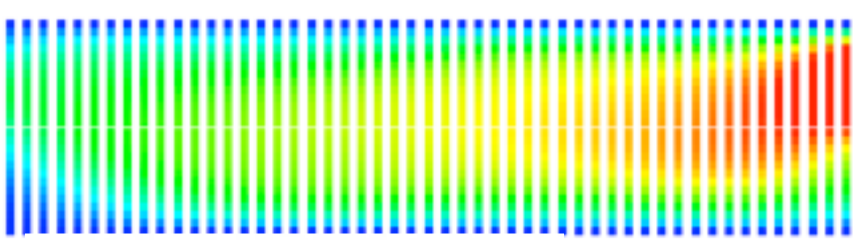
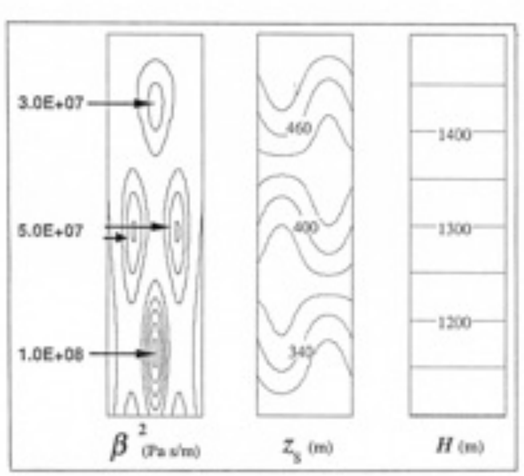
Elmer/Ice Grenoble 2017

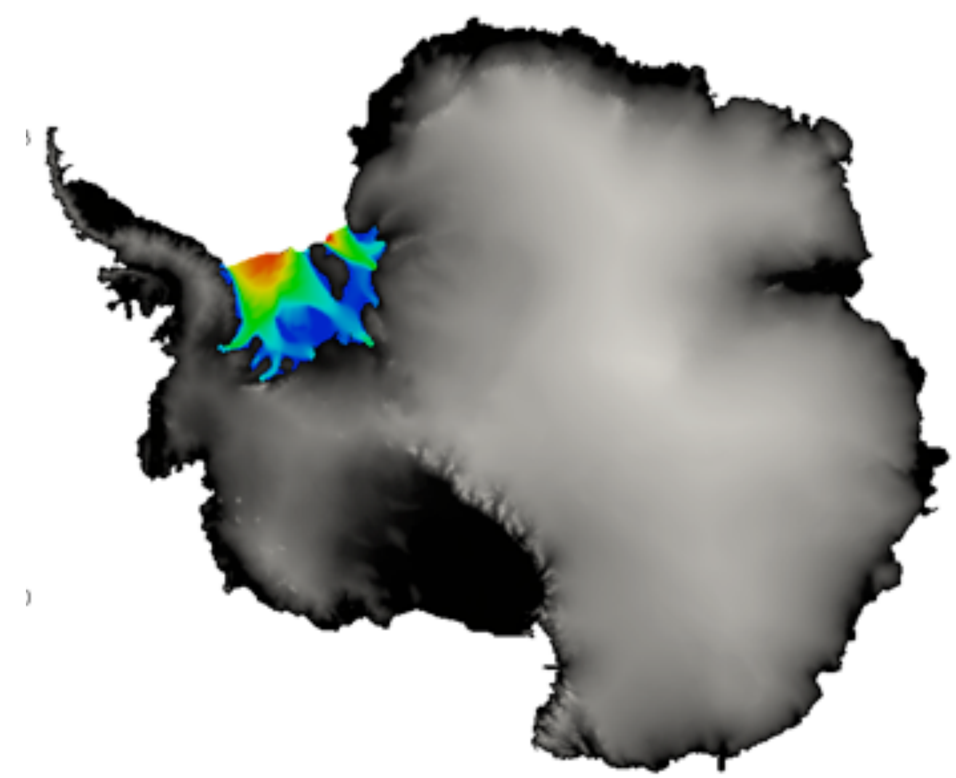
Inverse methods in Elmer/Ice
Applications with the SSA





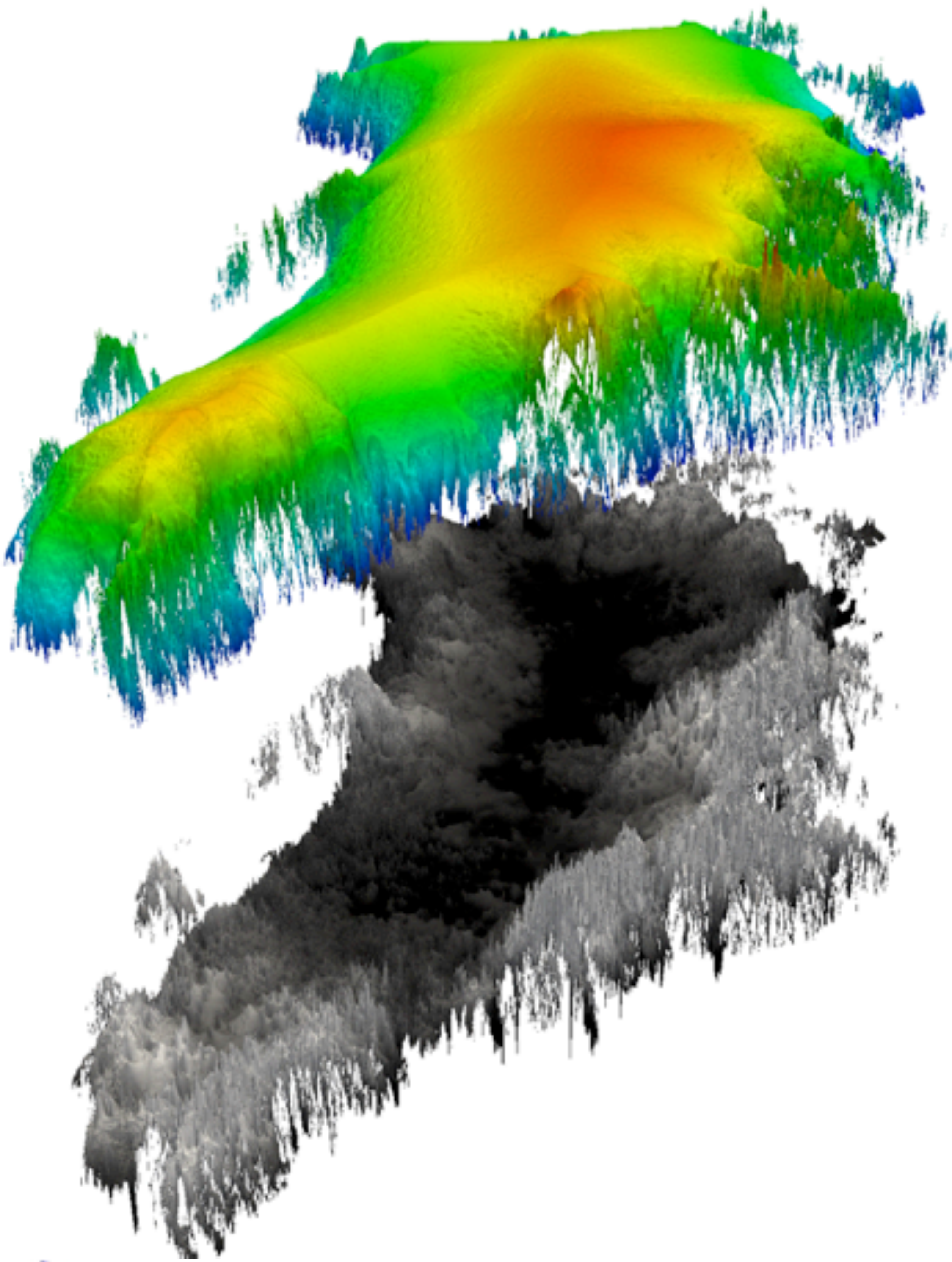
Fabien Gillet-Chaulet

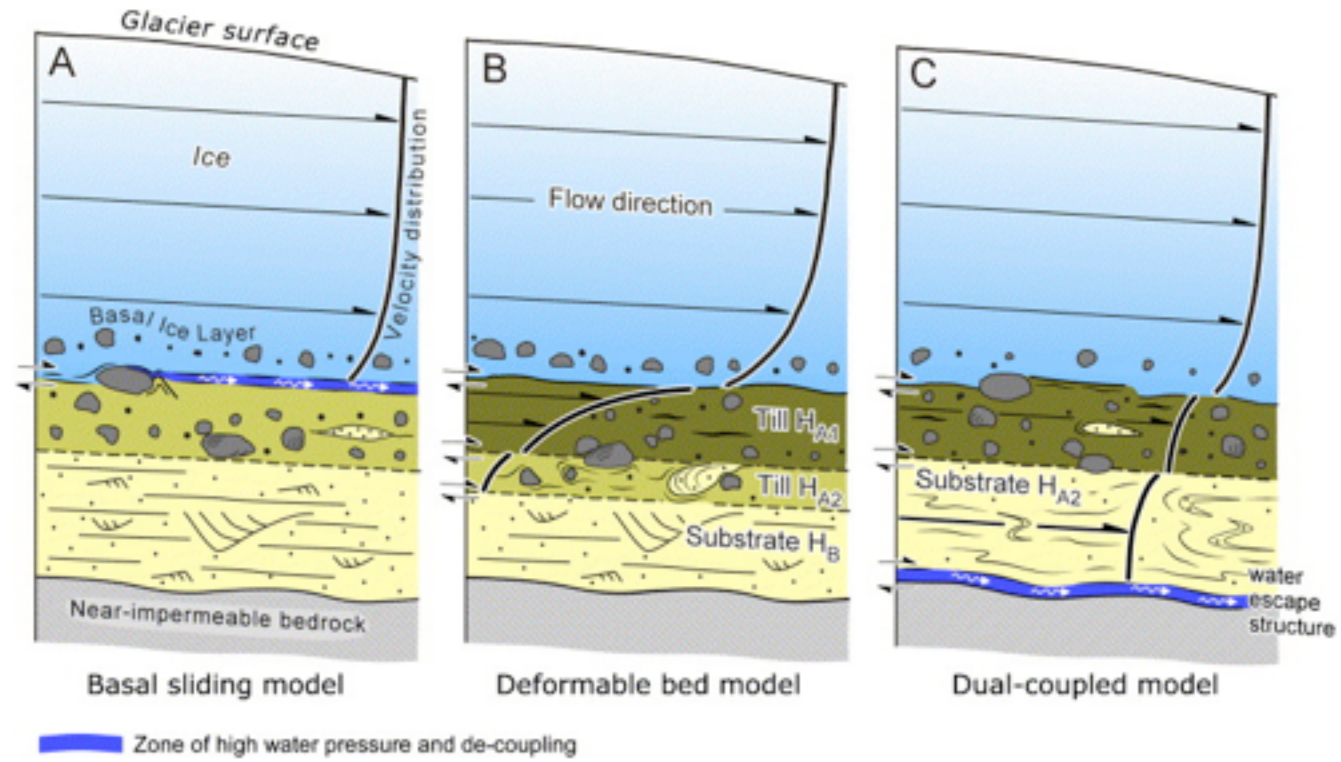
IGE - Grenoble - France

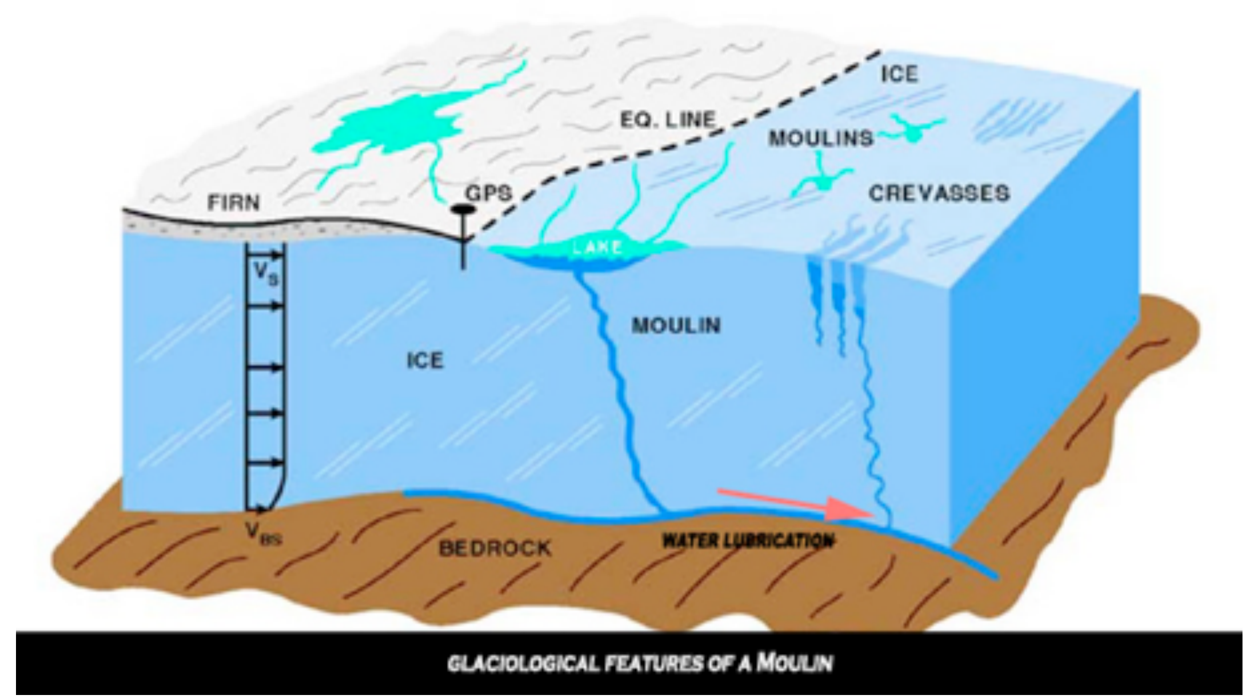


Uncertain parameterisations

e.g. friction of the ice on the bedrock highly variable in space and time
Usually prescribed as a friction law Tau=f(u)

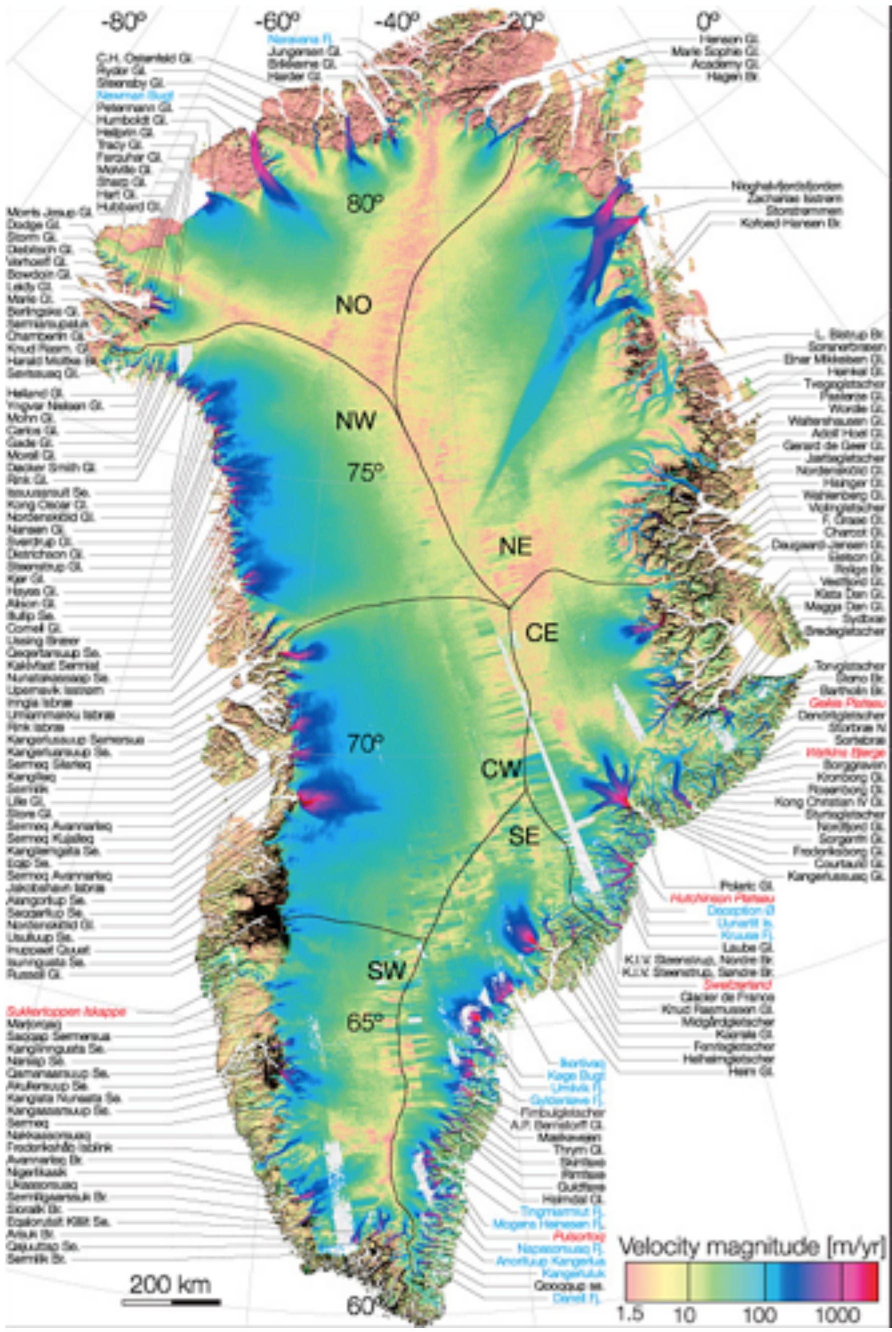




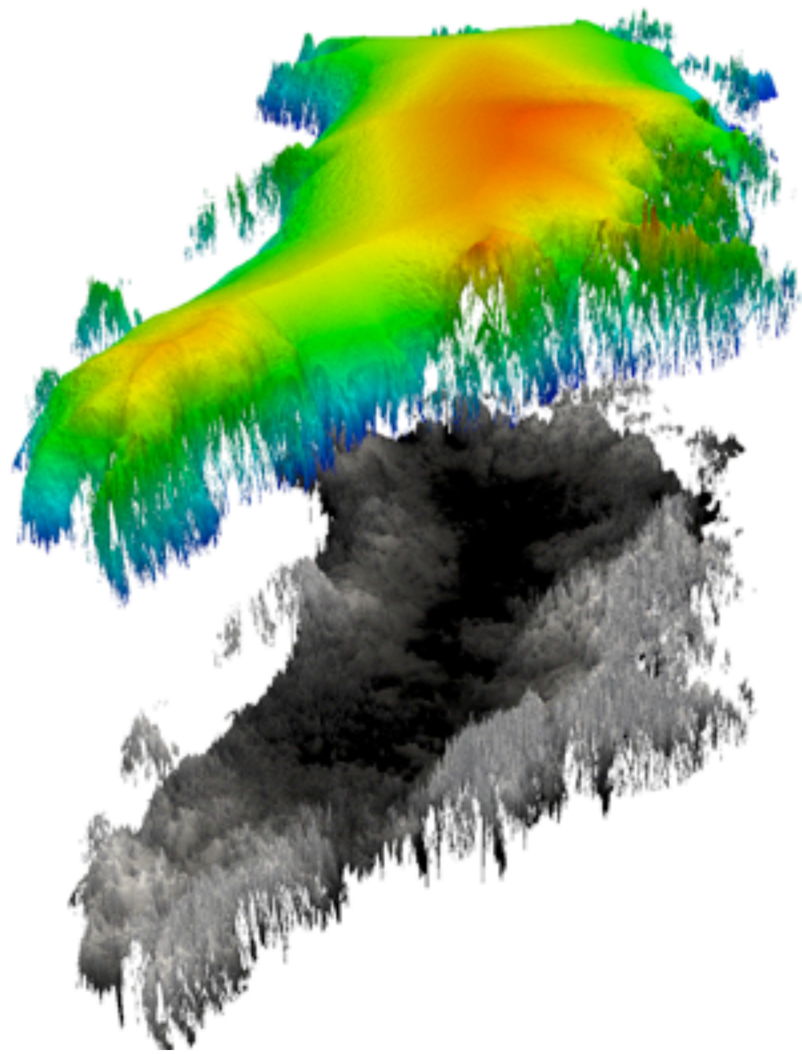


More and more available observations

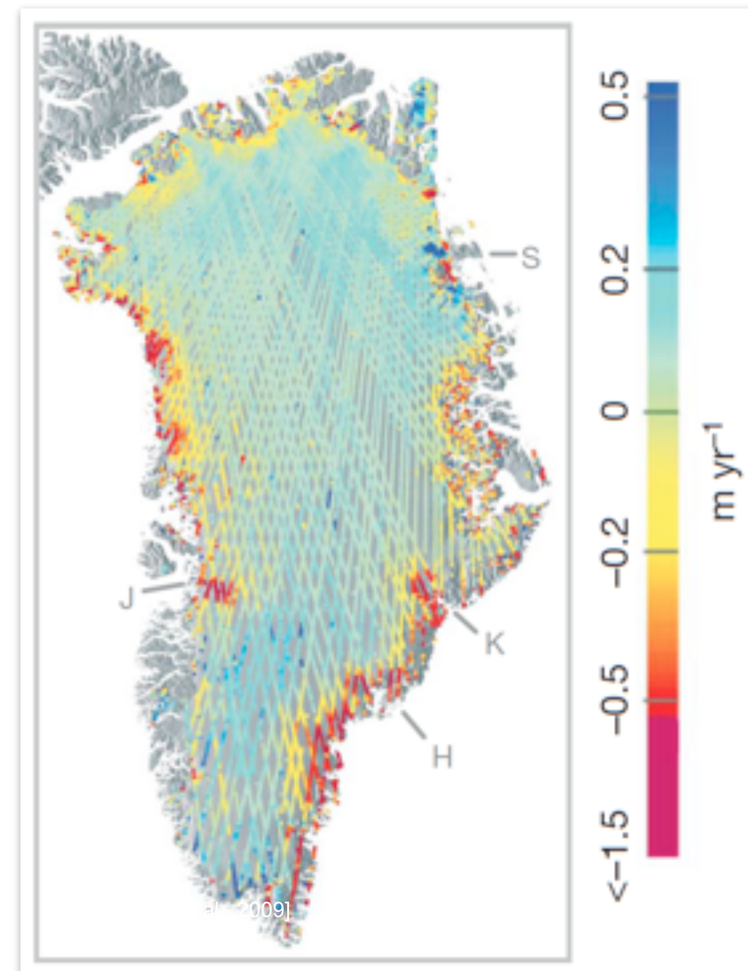
Surface velocities



Topography



ds/dt

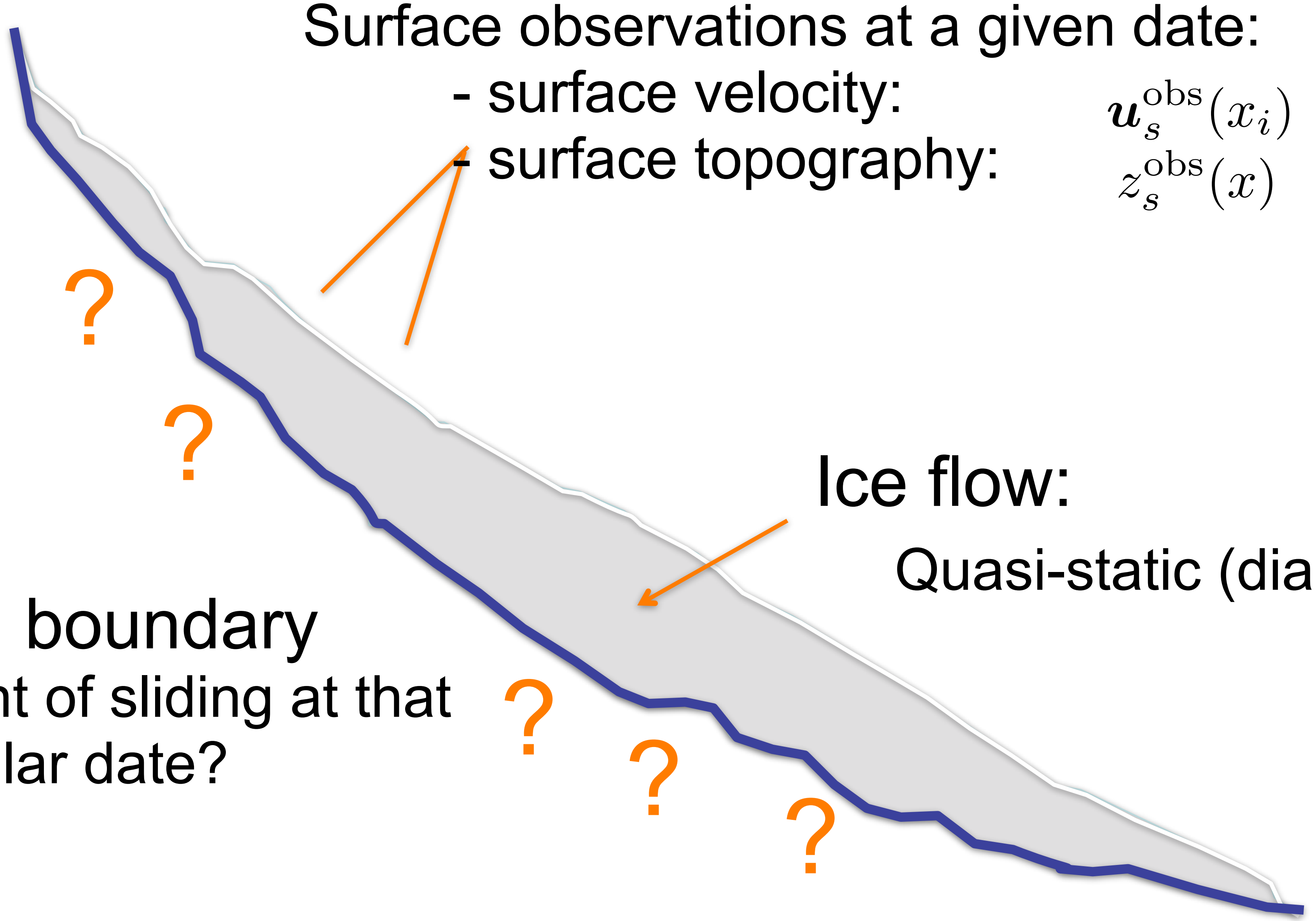


Specificity of ice flow

Very low Reynolds -> no history in the velocity

Surface observations at a given date:

- surface velocity: $\boldsymbol{u}_s^{\text{obs}}(x_i)$
- surface topography: $z_s^{\text{obs}}(x)$



Ice flow:

Quasi-static (diagnostic)

Basal boundary

Amount of sliding at that particular date?

Reconstruction of the basal conditions from surface measurements

Inverse methods in Elmer: balance equations

- **STOKES: 2 inverse methods** implemented in Elmer/Ice:
  - **Robin inverse method** (arthern and Gudmundsson, 2010)
  - **Adjoint method** (Mac Ayeal, 1993; Morlighem et al., 2010; Petra et al., 2012)

Characteristics:
=> restricted to **diagnostic** (no time evolution)
=> **slip coefficient** (Linear sliding law)
=> **ice viscosity**
=> could also do Neumann and Dirichlet BC (Adjoint method)

- **SSA:**
  - **Adjoint Method: in elmerice branch since 2016**
  - **Not yet documented on the wiki**
  - **Test case presented at the beginner course in Oslo in 2016**

- **Efficient minimisation library** (quasi-Newton algorithm)

Variational data assimilation

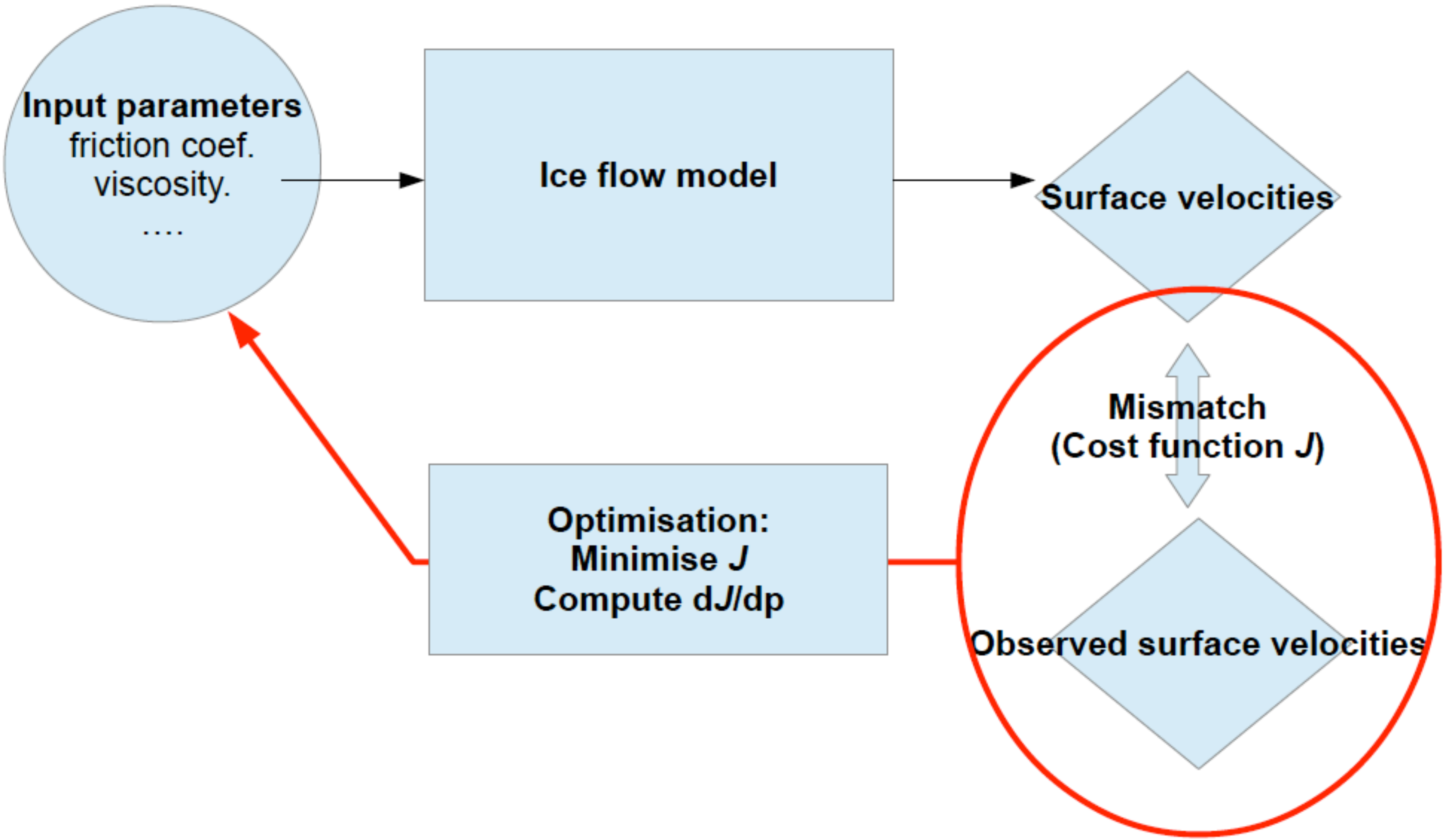
See pioneer paper from Mac Ayeal!!

Journal of Glaciology, Vol. 39, No. 131, 1993

A tutorial on the use of control methods in ice-sheet modeling

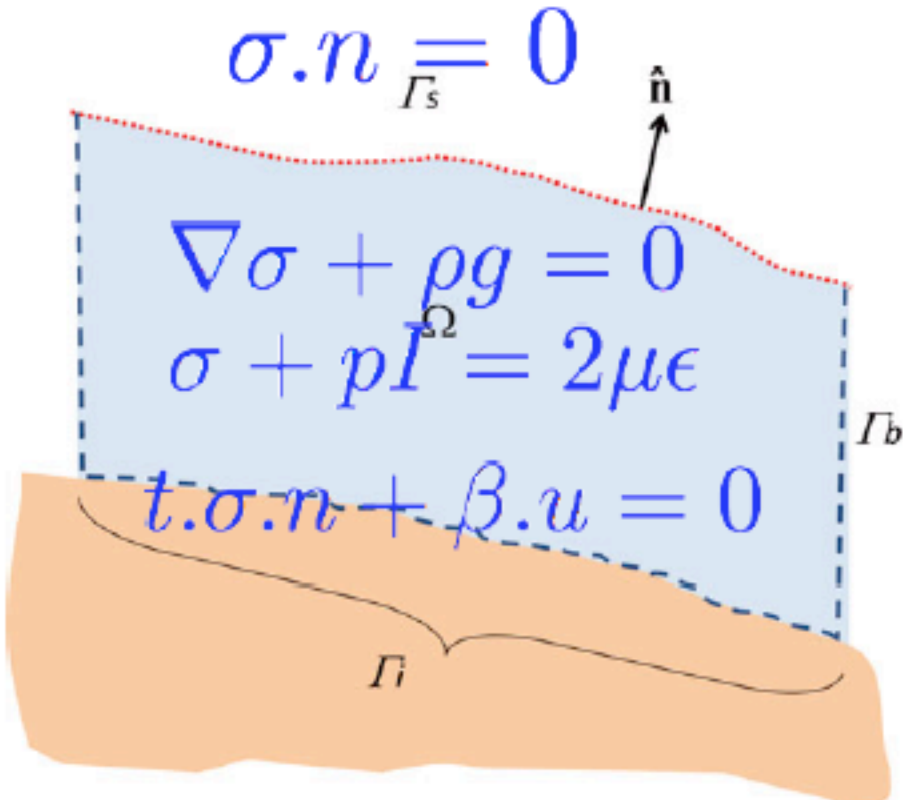
DOUGLAS R. MACAYEAL
Department of the Geophysical Sciences, The University of Chicago, Chicago, Illinois 60637, U.S.A.

Variational data assimilation



Adjoint method *(Mac Ayeal, 1993)*

Direct problem



1. Define a cost function

$$J = f(u)$$

$$e.g. \quad J = \int_{\Gamma_S} \frac{1}{2}(u - u^{obs})^2 d\Gamma$$

2. Insure that *u* is solution of your problem

$$J' = J(u) + \Lambda(\nabla\sigma + \rho g)$$

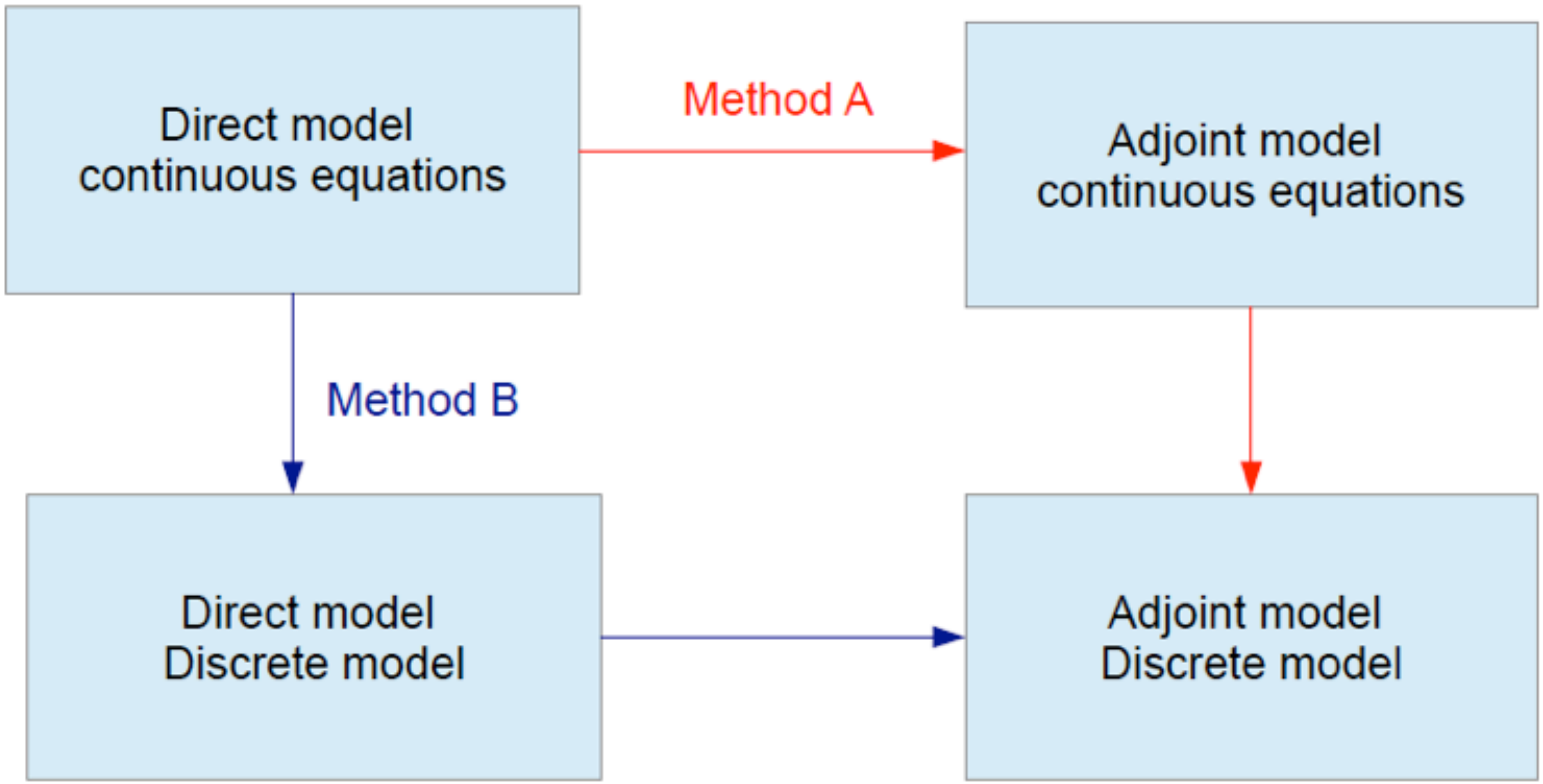
3. Minimisation of *J'* requires that all variations are 0

$d_\Lambda J' = 0$ => direct problem equation is satisfied

$d_u J' = 0$ => adjoint equations

=> gradient of *J* w.r. To input parameters *p* $\quad d_p J = f(\Lambda, u)$

Getting the adjoint



Usually Method A ≠ Method B

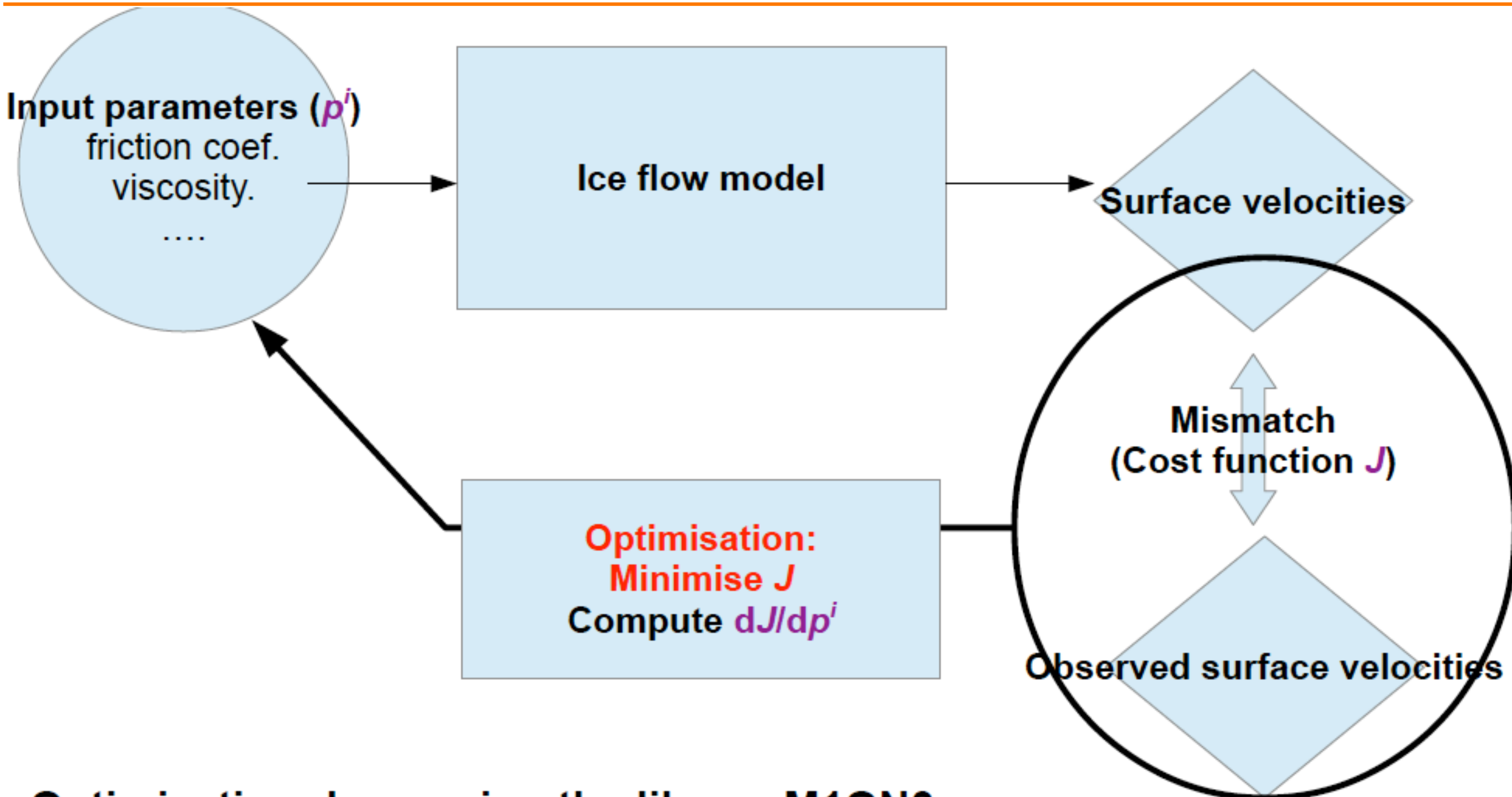
Method B should be preferred

Can be done using automatic differentiation

Pointer arrays not yet supported

=> crucial parts have been derived by hand

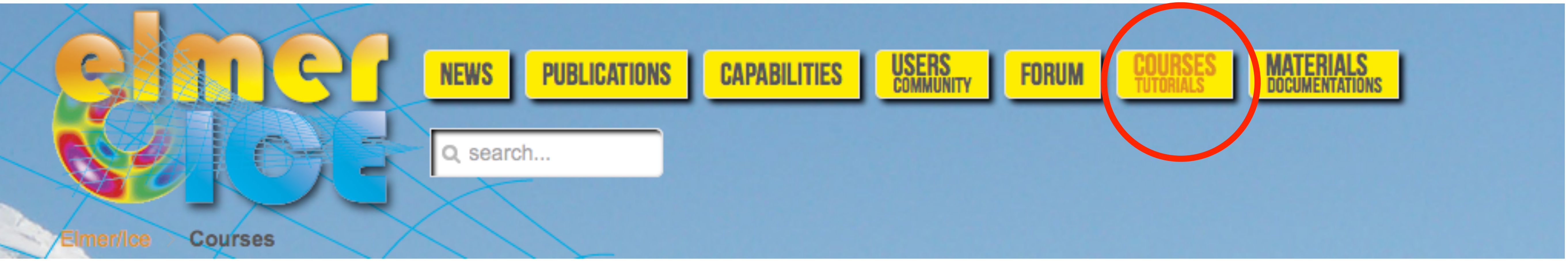
Optimisation algorithm: M1QN3



Optimisation done using the library M1QN3:

- Limited memory quasi-newton algorithm
- Implemented in reverse communication (i.e. called by Elmer within a solver)
- Iterative procedure: Input: p^i, J^i, dJ/dp^i – Output p^{i+1}
- https://who.rocq.inria.fr/Jean-Charles.Gilbert/modulopt/optimization-routines/m1qn3/m1qn3.html

Stokes: *Nothing new since the CSC - 2013 Advanced Course*



CSC - Espoo - 4-6 November 2013

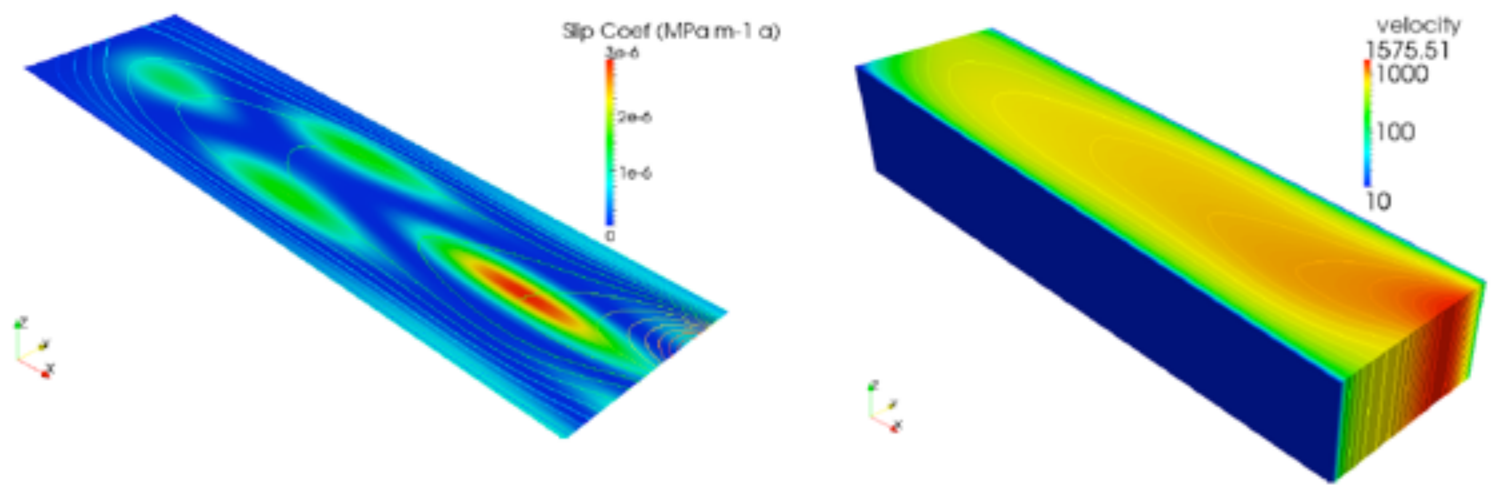
A 3-day Elmer/Ice advanced workshop was organised at CSC (Espoo, Finland) from the 4th to the 6th of November 2013. The course was held by Fabien Gillet-Chaulet (LGGE), Mika Malinen (CSC), Peter Råback (CSC) and Thomas Zwinger (CSC).

Title	Presentation	Material
Introduction to Elmer	pdf	-
Elmer Glaciological Modelling	pdf	-
Simple Hydro Toymodel	pdf	tar file
Structured Meshes	pdf	tar file
Enhanced pre-processing	pdf	USB stick
Block pre-conditioner	pdf	USB stick
Enhanced post-processing	pdf	ZIP archive
Make-file for YAMS on Ubuntu 64bit	-	tar archive
Mesh Adaptation using YAMS (see also these notes)	pdf	tar archive
Inverse methods	pdf	tar archive

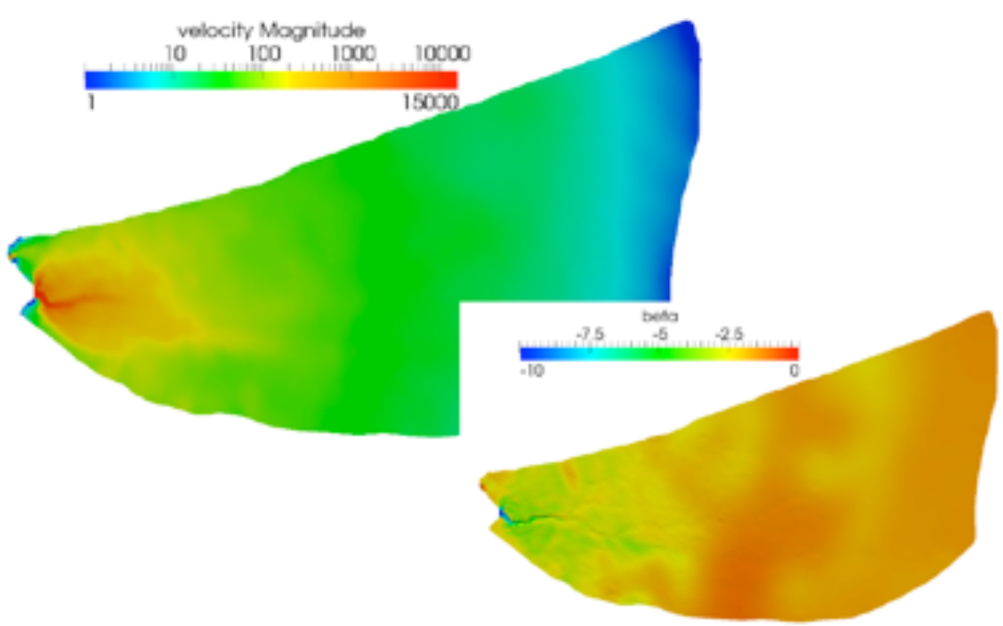
See the CSC-2013-Advanced Course Material:

- step by step construction of a «twin experiment» (set-up based on Mac Ayeal, 1993)

Ideal observed surface velocities



- Application to Jacobshavn Isbrae drainage basin



See also the Elmer/Ice reference paper

Geosci. Model Dev., 6, 1299–1318, 2013
www.geosci-model-dev.net/6/1299/2013/
doi:10.5194/gmd-6-1299-2013

Geoscientific Model Development
Open Access

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]

[1]Laboratoire de Glaciologie et Géophysique de l'Environnement, UJF-Grenoble, CNRS – UMR5183, Saint-Martin-d'Hères, France
[2]Institut Universitaire de France, Paris, France
[3]CSC-IT Center for Science Ltd., Espoo, Finland
[4]Institute of Low Temperature Science, Hokkaido University, Sapporo, Japan
[5]British Antarctic Survey, Cambridge, UK
[6]Arctic Centre, University of Lapland, Rovaniemi, Finland
[7]Uppsala University, Uppsala, Sweden

Adjoint of the SSA

Field equations:

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

See applications:

- **Assimilation of viscosity and friction:**
  - ***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*, 2016**

- **Assimilation of bedrock topography and friction:**
  - **Mosbeux, C., Gillet-Chaulet, F., Gagliardini, O., 2016. Comparison of adjoint and nudging methods to initialise ice sheet model basal conditions. Geosci. Model Dev. 2016**

- **Assimilation of friction at different dates:**
  - **Gillet-Chaulet, F., Durand, G., Gagliardini, O., Mosbeux, C., Mouginot, J., Rémy, F., Ritz, C., 2016. Assimilation of surface velocities acquired between 1996 and 2010 to constrain the form of the basal friction law under Pine Island Glacier. Geophys. Res. Lett. 2016**

SSA: *See Oslo - 2016 Course*

NEWS | PUBLICATIONS | CAPABILITIES | USERS COMMUNITY | FORUM | COURSES TUTORIALS | MATERIALS DOCUMENTATIONS

search...

Elmer/Ice » Courses

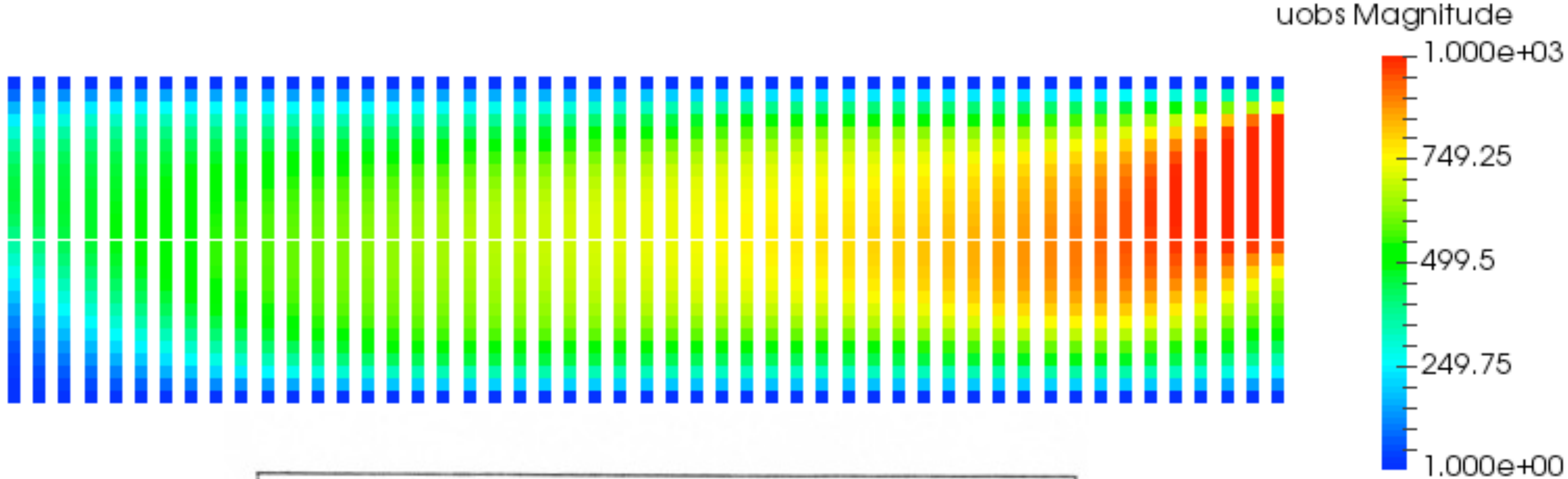
UiO - Oslo - 31st October, 1st and 2nd November 2016

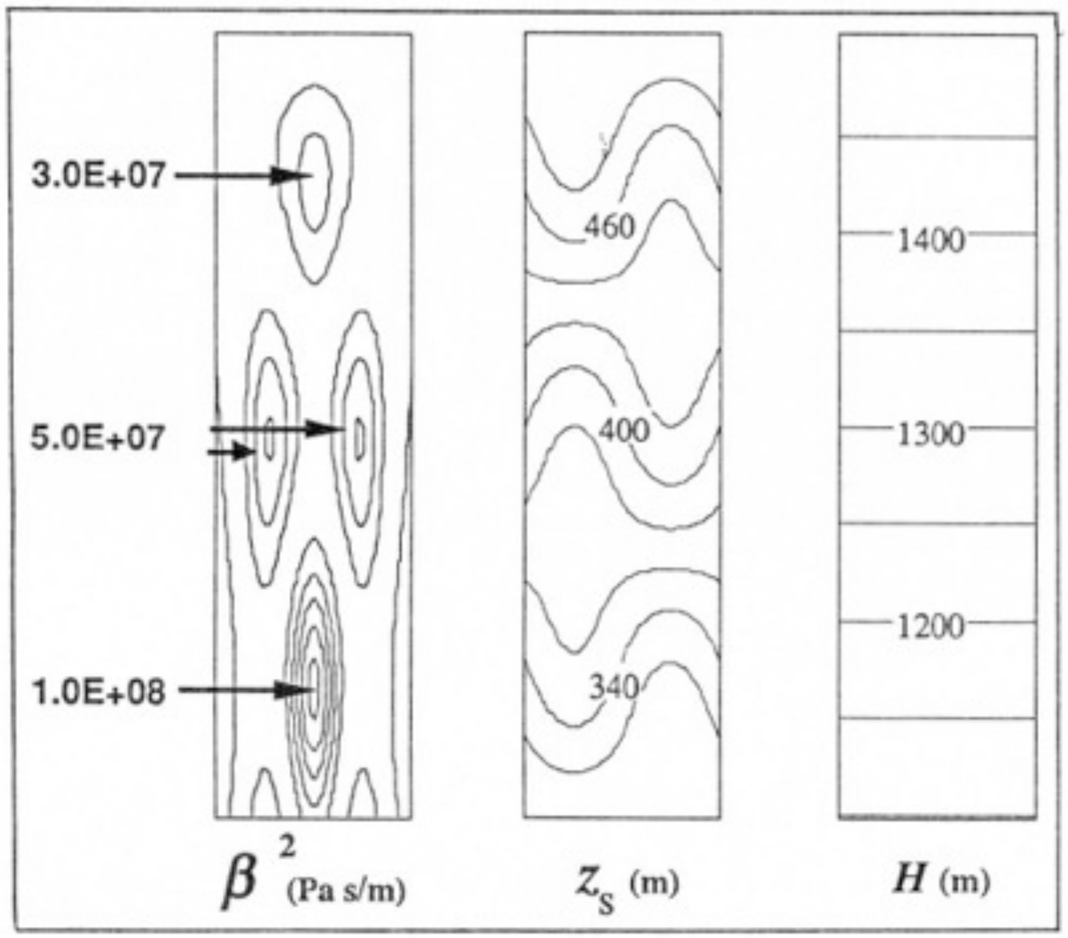
The Department of Geosciences of the University of Oslo in cooperation with LGGE (France) and CSC (Finland) is organizing a 3-day beginner Elmer/Ice course on the 31st of October, 1st and 2nd of November 2016. This course is sponsored by the Labex OSUG@2020 and eScience tools for investigating climate change (eSTICC).

Title	Presentation	Material
Introduction	pdf	-
Toy flow-line model	pdf	tar file
Tête Rousse	pdf	tar file
Ice flow and temperature coupling	pdf	tar file
Inverse modelling and SSA	pdf	tar file

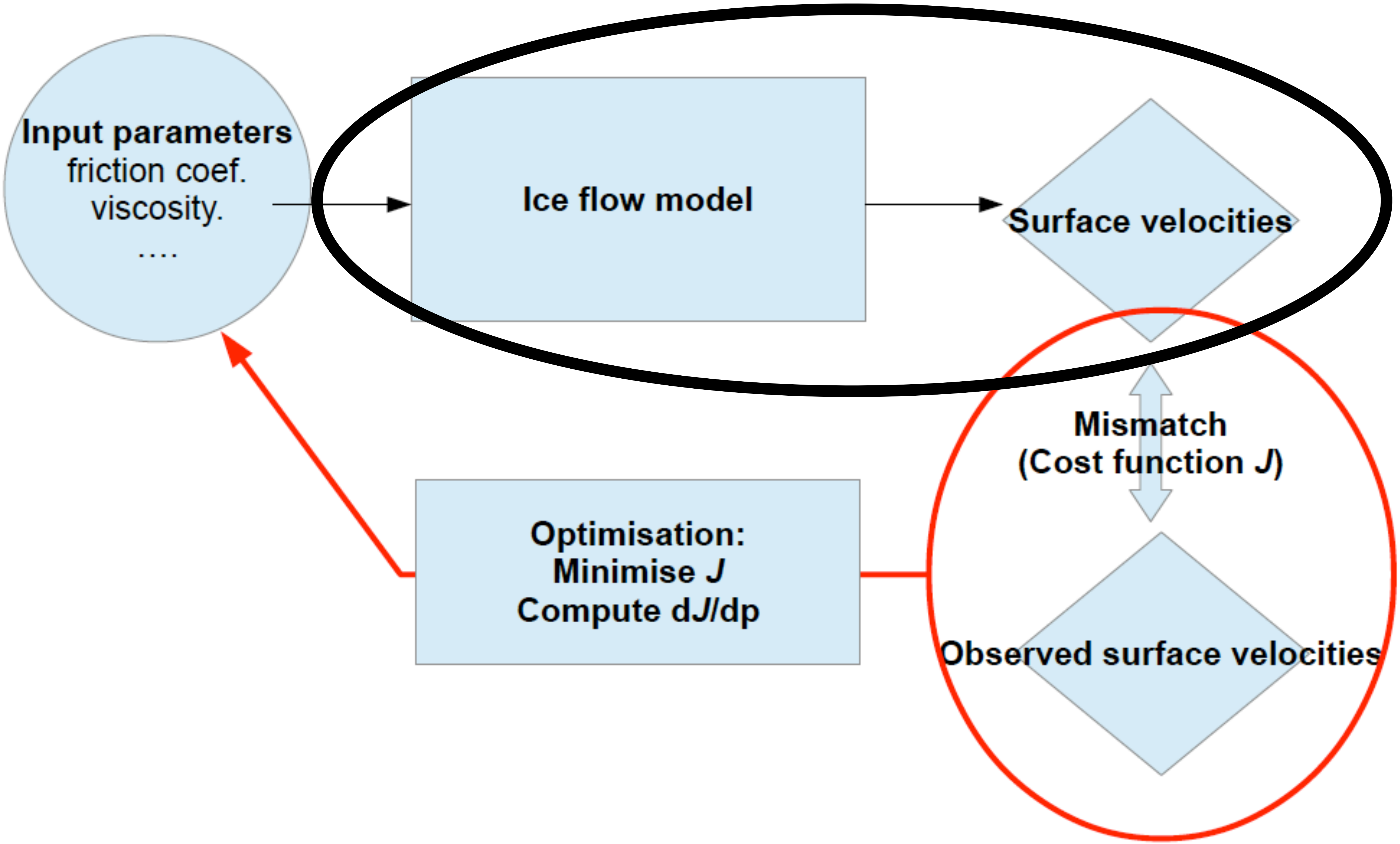
Download the list of participants, program and useful informations here.

Repeat Mac Ayeal Twin Experiment





Variational data assimilation



Initial guess

the guess (square root of)

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Initial Condition 1
 BetaS = Variable coordinate 1, Coordinate 2
     REAL MATC "betaSquare(tx)"
! alpha is the optimised variable
 alpha = Real 1.0e-3

 Zb = Variable coordinate 1, Coordinate 2
     REAL MATC "zb(tx)"

 Zs = Variable coordinate 1, Coordinate 2
      REAL MATC "zs(tx)"

 SSAVelocity 1 = Real 0.0
 SSAVelocity 2 = Real 0.0
End
```

the truth

```
$ function betaSquare(tx) {\
  Lx = 200.0e3;\
  Ly = 50.0e03;\
  yearinsec = 365.25*24*60*60;\
  F1=sin(3.0*pi*tx(0)/Lx)*sin(pi*tx(1)/Ly);\
  F2=sin(pi*tx(0)/(2.0*Lx))*cos(4.0*pi*tx(1)/Ly);\
  beta=5.0e3*F1+5.0e03*F2;\
  _betaSquare=beta*beta/(1.0e06*yearinsec);\
}
```

Compute model velocity

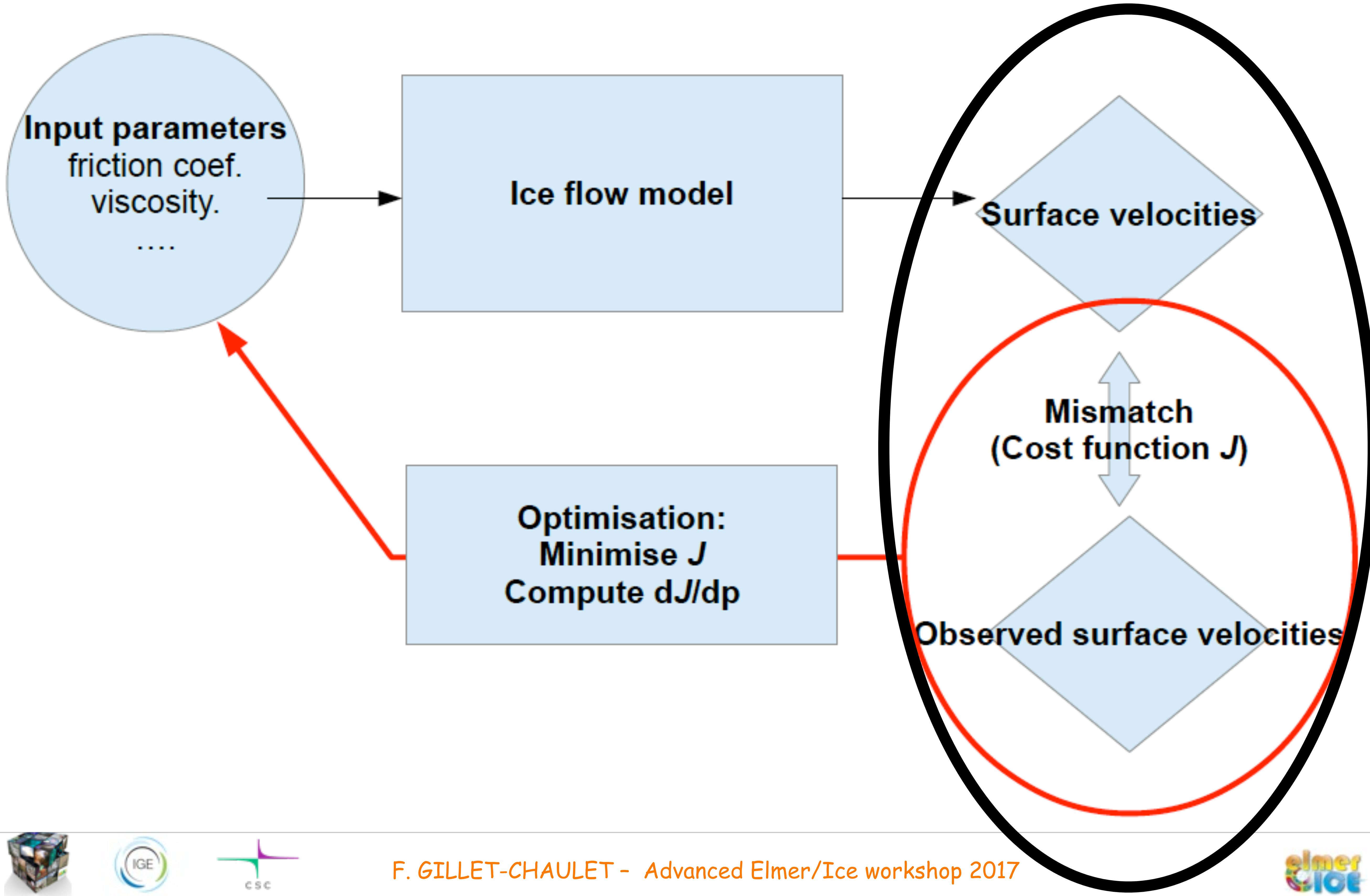
```
Material 1

  Viscosity Exponent = Real $1.0e00/3.0e00
  Critical Shear Rate = Real 1.0e-10

  SSA Mean Density = Real $rhoi
  SSA Mean Viscosity = Real $ 1.8e8*1.0e-6*(2.0*yearinsec)^(-1.0/3.0)

  SSA Friction Law = String "linear"
  ! The friction parameter is the square of the optimised variable to insure > 0
  SSA Friction Parameter = Variable alpha
      REAL MATC "tx*tx"
End
```

```
Solver 1
  Equation = "SSA"
  Variable = -dofs 2 "SSAVelocity"

  Procedure = "AdjointSSASolvers" "AdjointSSA_SSASolver"

!! Mandatory for the adjoint
  Calculate Loads = Logical True

  Linear System Solver = Direct
  Linear System Direct Method = mumps

  Nonlinear System Max Iterations = 50
  Nonlinear System Convergence Tolerance  = 1.0e-10
  Nonlinear System Newton After Iterations = 40
  Nonlinear System Newton After Tolerance = 1.0e-06
  Nonlinear System Relaxation Factor = 1.00

  Steady State Convergence Tolerance = Real 1.0e-12

  Exported Variable 1 = Zb
  Exported Variable 2 = Zs
  Exported Variable 3 = BetaS
  Exported Variable 4 = CostValue
  Exported Variable 5 = DJDBeta
  Exported Variable 6 = -dofs 2 "Velocityb"
End
```

The direct solver used to make the adjoint (may not be up-to-date wr the latest ssa solver)

For the accuracy of the adjoint use Newton method for non-linear iterations

Variational data assimilation

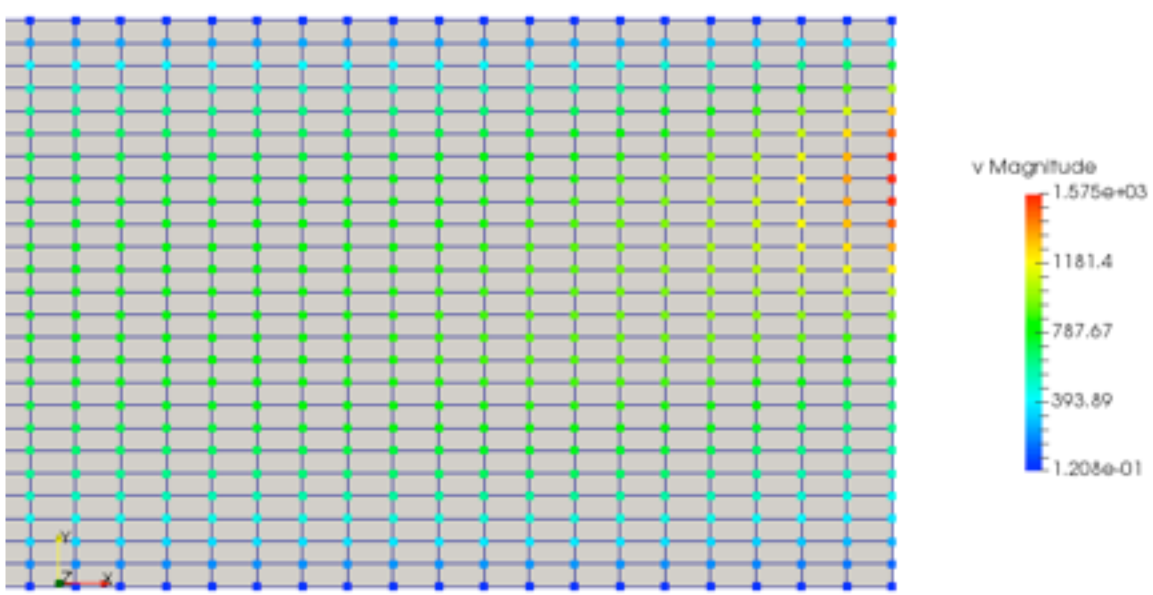


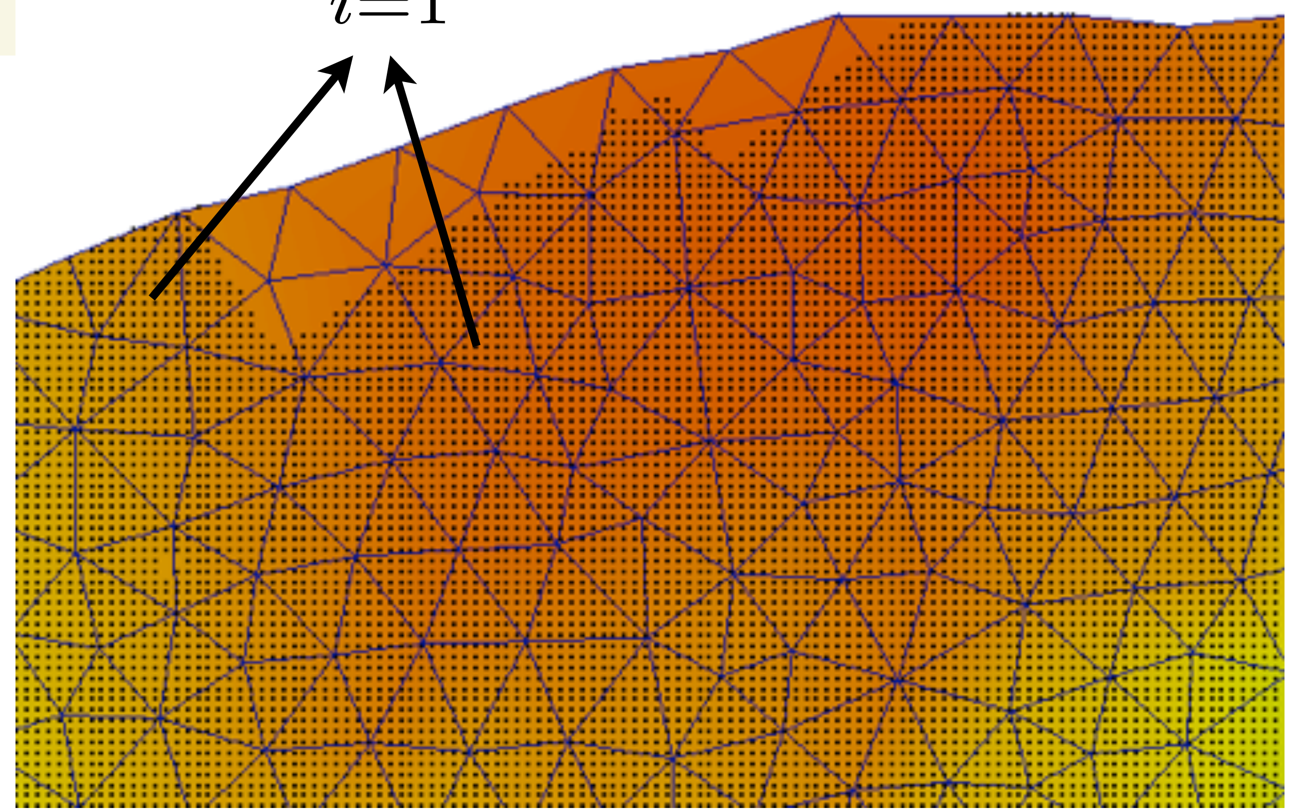
Compute the cost function

```
Solver 2
  Equation = "Cost"

   !!  Solver need to be associated => Define dumy variable
    Variable = -nooutput "CostV"
    Variable DOFs = 1

    procedure = "AdjointSSASolvers" "AdjointSSA_CostDiscSolver"

   Problem Dimension = Integer 2 !2D mesh and 2D SSA Solution
   Cost Variable Name = String "CostValue"  ! Name of Cost Variable
   Lambda = Real 1.0
 ! save the cost as a function of iterations (iterations,Cost,rms=sqrt(2*Cost/Ndata)
   Cost Filename = File "Cost_$name$.dat"

   Observed Variable Name = String "SSAVelocity"
 ! ASCII File with data: x,y,u,v
   Observation File Name = File "../DATA/MacAyeal_VELOCITIES.txt"

end
```

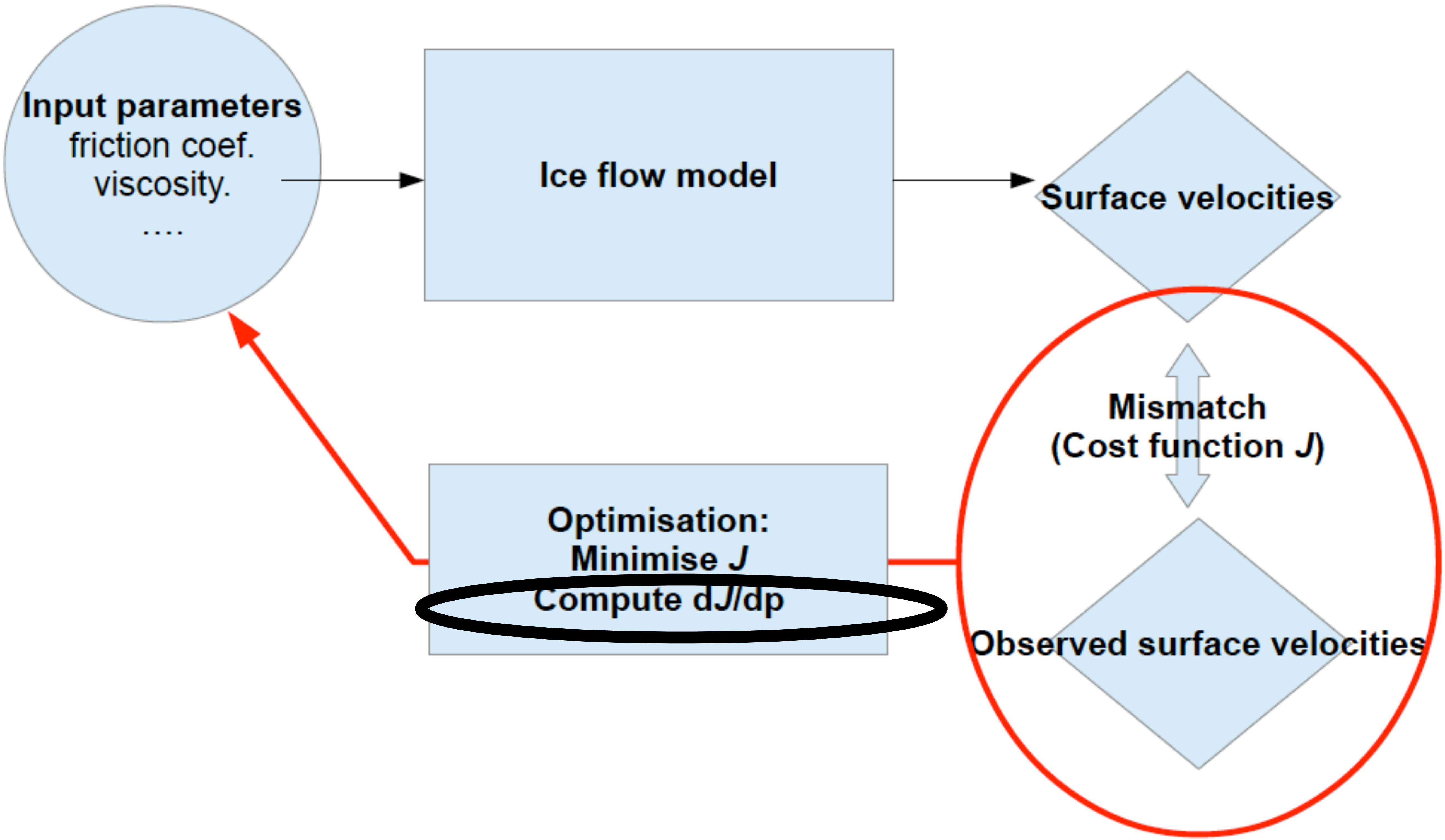
```
#date,time,1/3/2016,10:35:52
#lambda, 0.10000000E+01
# iter, Lambda*J0, rms
  0.10000E+01   0.31721506E+06   0.21590507E+02
  0.20000E+01   0.89979177E+05   0.11498917E+02
  0.30000E+01   0.15123345E+05   0.47142201E+01
  0.40000E+01   0.79399313E+04   0.34158147E+01
  0.50000E+01   0.59819521E+04   0.29648819E+01
  0.60000E+01   0.37836248E+04   0.23579792E+01
  0.70000E+01   0.26658066E+04   0.19792482E+01
```

$$\sum_{i=1}^{obs} \|\boldsymbol{u}^{mod} - \boldsymbol{u}^{obs}\|^2\Big|_{obs}$$





Variational data assimilation



Compute the gradient

```
!!!!  Adjoint Solution
Solver 3
  Equation = "Adjoint"
  Variable = Adjoint
  Variable Dofs = 2

  procedure = "AdjointSSASolvers" "AdjointSSA_AdjointSolver"

!Name of the flow solution solver
  Flow Solution Equation Name = string "SSA"

   Linear System Solver = Direct
   Linear System Direct Method = mumps
End

!!!!!  Compute Derivative of Cost function / Beta
Solver 4
  Equation = "DJDBeta"

  !!  Solver need to be associated => Define dumy variable
    Variable = -nooutput "DJDB"
    Variable DOFs = 1

    procedure = "AdjointSSASolvers" "AdjointSSA_GradientSolver"

    Flow Solution Name = String "SSAVelocity"
    Adjoint Solution Name = String "Adjoint"
    Compute DJDBeta = Logical True   ! Derivative with respect to the Friction parameter
    DJDBeta Name = String "DJDBeta"
end
```

Boundary conditions

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Boundary Condition 1
  Name = "Side Walls"
  Target Boundaries(2) = 1 3

  SSAVelocity 1 = Real 0.0
  SSAVelocity 2 = Real 0.0

  Adjoint 1 = Real 0.0
  Adjoint 2 = Real 0.0
End

Boundary Condition 2
  Name = "Inflow"
  Target Boundaries = 4

   SSAVelocity 1 = Variable Coordinate 2
      REAL MATC "4.753e-6*yearinsec*(sin(2.0*pi*(Ly-tx)/Ly)+2.5*sin(pi*(Ly-tx)/Ly))"
   SSAVelocity 2 = Real 0.0

  Adjoint 1 = Real 0.0
  Adjoint 2 = Real 0.0
End

Boundary Condition 3
  Name = "OutFlow"
  Target Boundaries = 2

   SSAVelocity 1 = Variable Coordinate 2
      REAL MATC "1.584e-5*yearinsec*(sin(2.0*pi*(Ly-tx)/Ly)+2.5*sin(pi*(Ly-tx)/Ly)+0.5*sin(3.0*pi*(Ly-tx)/Ly))"
   SSAVelocity 2 = Real 0.0

  Adjoint 1 = Real 0.0
  Adjoint 2 = Real 0.0
End
```

Change of variable

We have computed the gradient wr to the friction parameter; physically this parameter should remain positive => make change of variable $\beta = \alpha^2$ or $\beta = 10^\alpha$

```
Solver 5
  Equation = "UpdateExport"
    Procedure = File "ElmerIceSolvers" "UpdateExport"
  Variable = -nooutput "dumy"

!used here to update DJDalpha from DJDbeta (see correponding line in Body Force section)
  Exported Variable 1 = -dofs 1 alpha
  Exported Variable 2 = -dofs 1 DJDalpha
End
```

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

! change of variable; DJDBeta is the derivative of the Cost fn w.r.t. the slip coeff.
!                     as slip coeff=alpha^2 => DJDalpha=DJDBeta * DBeta / Dalpha
  DJDalpha = Variable DJDBeta , alpha
     REAL MATC "2.0*tx(0)*tx(1)"
End
```

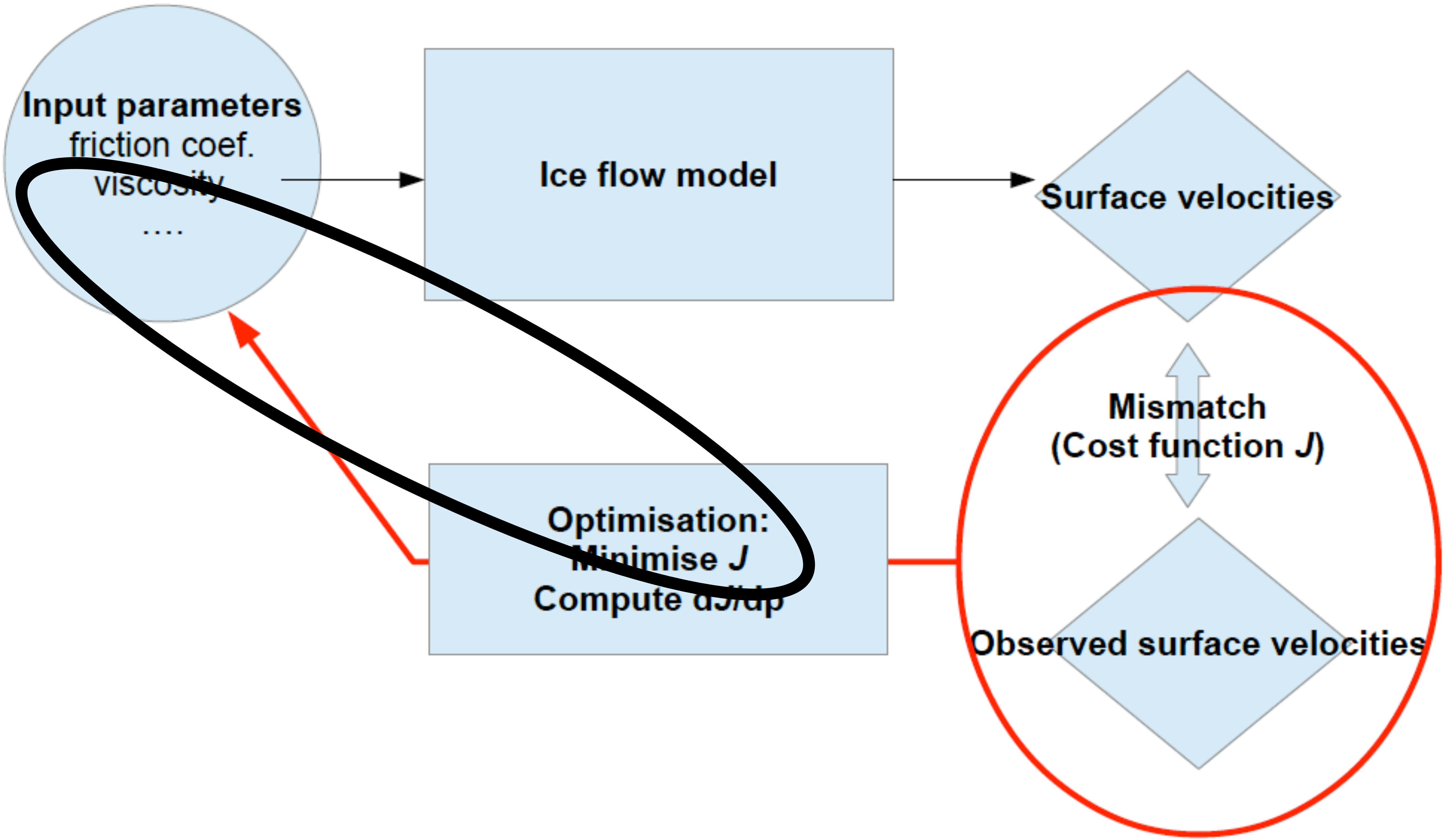
Regularisation

```
!!!!!  Compute Regularistaion term
!   Regularisation by default is: Lambda * int_{Pb dimension} 0.5 * (d(var)/dx)**2
!   A priori regularisation can also be used ( A priori Regularisation=True) :
!                                 Lambda * int_{Pb dimension} 0.5 *(1/sigma**2)*(var-var{a_priori})**2
!
!     OUTPUT are : J and DJDvar
Solver 6
  Equation = "DJDBeta_Reg"

  !!  Solver need to be associated => Define dumy variable
    Variable = -nooutput "DJDBReg"
    Variable DOFs = 1

    procedure = "AdjointSSASolvers" "AdjointSSA_CostRegSolver"

    Problem Dimension=Integer 2
    Cost Filename=File "CostReg_$name$.dat"
    Optimized Variable Name= String "alpha"
    Gradient Variable Name= String "DJDalpha"
    Cost Variable Name= String "CostValue"
    Lambda= Real $Lambda
    Reset Cost Value= Logical False  !=> DJDapha already initialized in solver DJDBeta; switch off initialisation to 0 at the
beginning of this solver
    A priori Regularisation= Logical False
end
```

Most real inverse problem are ill-posed
=> add a-priori knowldedge to regularise the problem
=> not needed here as the obs. are perfect!

```
#date,time,1/3/2016,10:35:52
#lambda, 0.00000000E+00
# iter, Jreg
  0.10000E+01   0.96510928E-35
  0.20000E+01   0.26697707E-06
  0.30000E+01   0.91582437E-06
  0.40000E+01   0.12325799E-05
  0.50000E+01   0.14043708E-05
  0.60000E+01   0.16939676E-05
  0.70000E+01   0.22054452E-05
```

Variational data assimilation

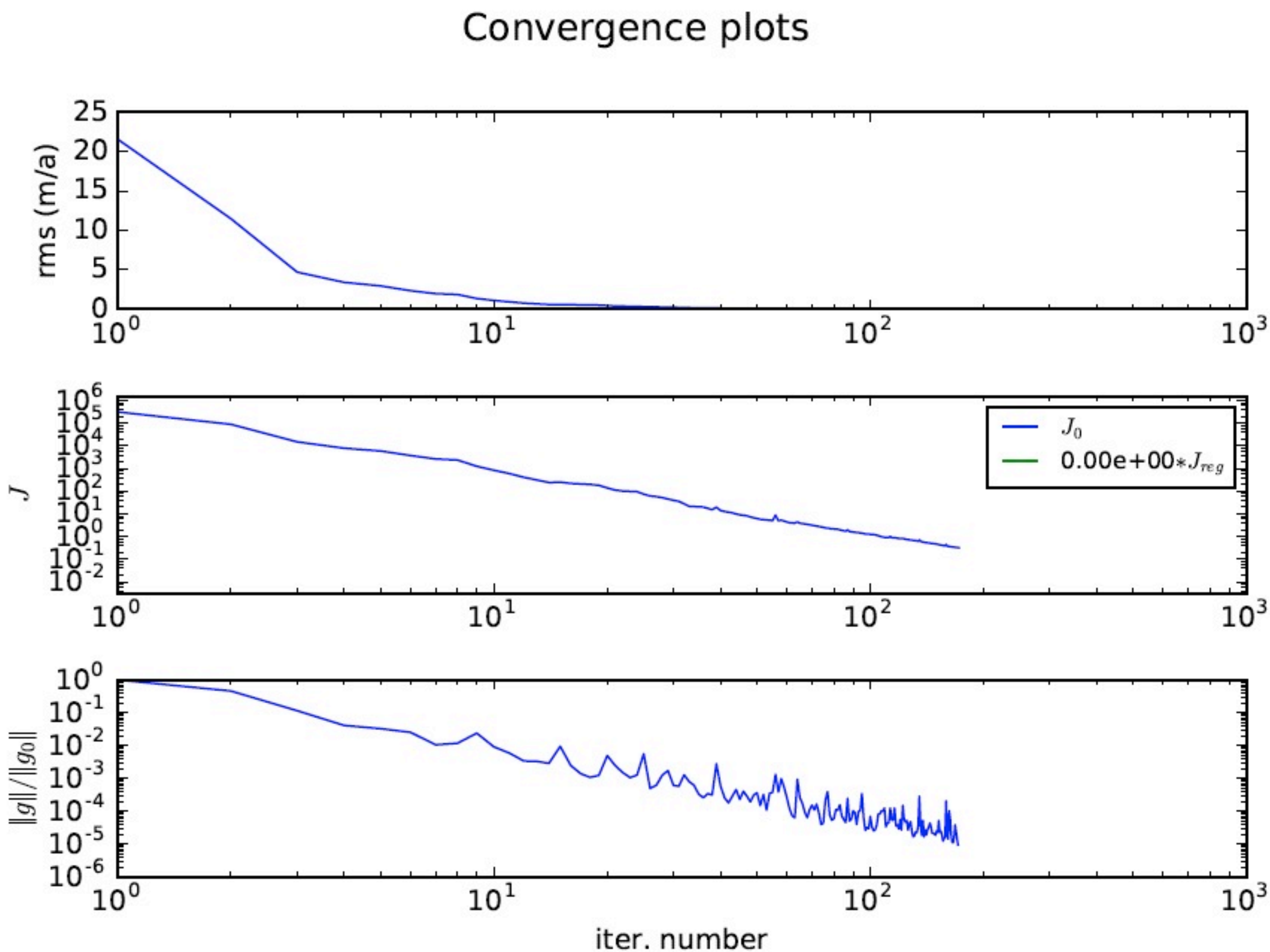


Optimisation

```
!!!!! Optimization procedure : Parallel only
Solver 7
  Equation = "Optimize_m1qn3"
  !!  Solver need to be associated => Define dumy variable
    Variable = -nooutput "UB"
    Variable DOFs = 1

  procedure = "ElmerIceSolvers" "Optimize_m1qn3Parallel"

  Cost Variable Name = String "CostValue"
  Optimized Variable Name = String "alpha"
  Gradient Variable Name = String "DJDalpha"
  gradient Norm File = File "GradientNormAdjoint_$name$.dat"


 ! M1QN3 Parameters
  M1QN3 dxmin = Real 1.0e-10
  M1QN3 epsg = Real  1.e-5
  M1QN3 niter = Integer 200
  M1QN3 nsim = Integer 200
  M1QN3 impres = Integer 5
  M1QN3 DIS Mode = Logical False
  M1QN3 df1 = Real 0.5
  M1QN3 normtype = String "dfn"
  M1QN3 OutputFile = File  "M1QN3_$name$.out"
  M1QN3 ndz = Integer 20

end
```

```
#10/31/2016      10:35:52
  0.10000E+01    0.84790499E+08
  0.20000E+01    0.38736142E+08
  0.30000E+01    0.97311488E+07
  0.40000E+01    0.34591176E+07
  0.50000E+01    0.27595369E+07
  0.60000E+01    0.21367979E+07
  0.70000E+01    0.89028275E+06
  0.80000E+01    0.99144675E+06
  0.90000E+01    0.20111705E+07
```

Check the convergence!!



Output information

M1QN3 last 7 lines

```
m1qn3: output mode is  1
    number of iterations:          158
    number of simulations:          171
    realized relative precision on g:  9.33D-06
    f              =  3.24456286D-01
    dfn-norm of g =  7.90817523D+02
```

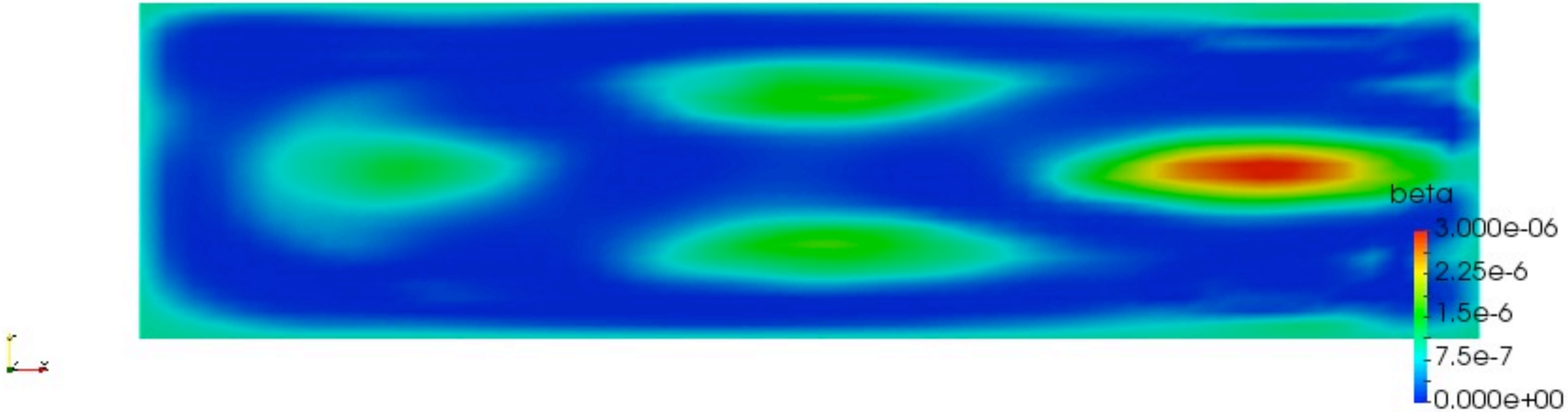
M1QN3 documentation

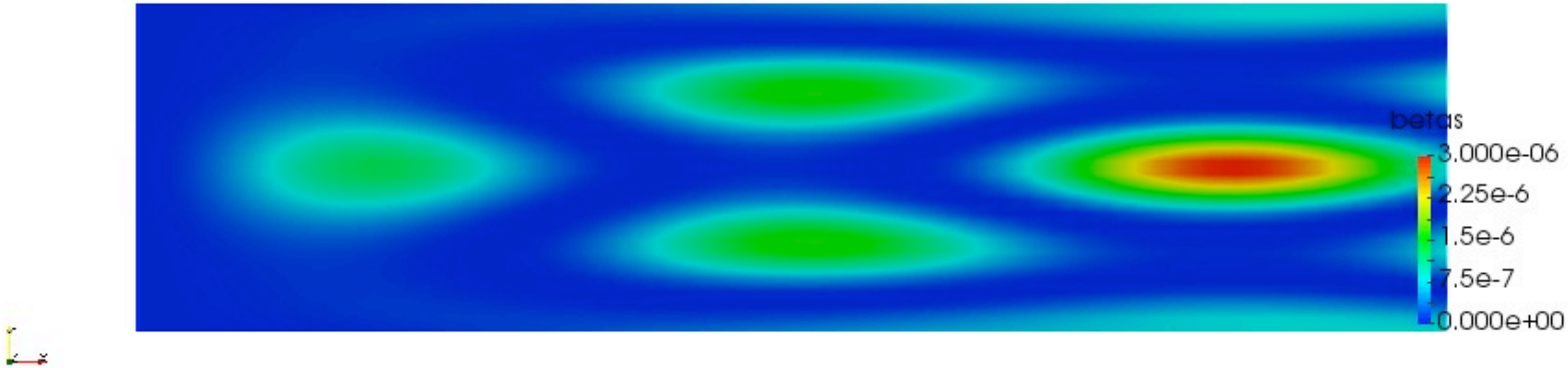
omode (O): Integer variable that specifies the output mode of m1qn3. The following values are meaningful.

- = 0: The simulator asks to stop by returning the value indic = 0.
- = 1: This is the normal way of stopping for m1qn3: the test on the gradient is satisfied (see the meaning of epsg).
- = 2: One of the input arguments is not well initialized. This can be:
  - $n \leq 0$, $niter \leq 0$, $nsim \leq 0$, $dxmin \leq 0.0$ or $epsg \notin]0,1[$,
  - $ndz < 5n+1$ (in SIS mode) or $ndz < 6n+1$ (in DIS mode): not enough storage in memory,
  - the contents of iz is not correct for a warm restart,
  - the starting point is almost optimal (the norm of the initial gradient is less than 10^{-20}).
- = 3: The line-search is blocked on $tmax = 10^{20}$ (see section 4.4 and the documentation on mlis3 in MODULOPT library).
- = 4: The maximal number of iterations is reached.
- = 5: The maximal number of simulations is reached.
- = 6: Stop on dxmin during the line-search (see section 4.4).
- = 7: Either $\langle g, d \rangle$ is nonnegative or $\langle y, s \rangle$ is nonpositive (see section 4.4).

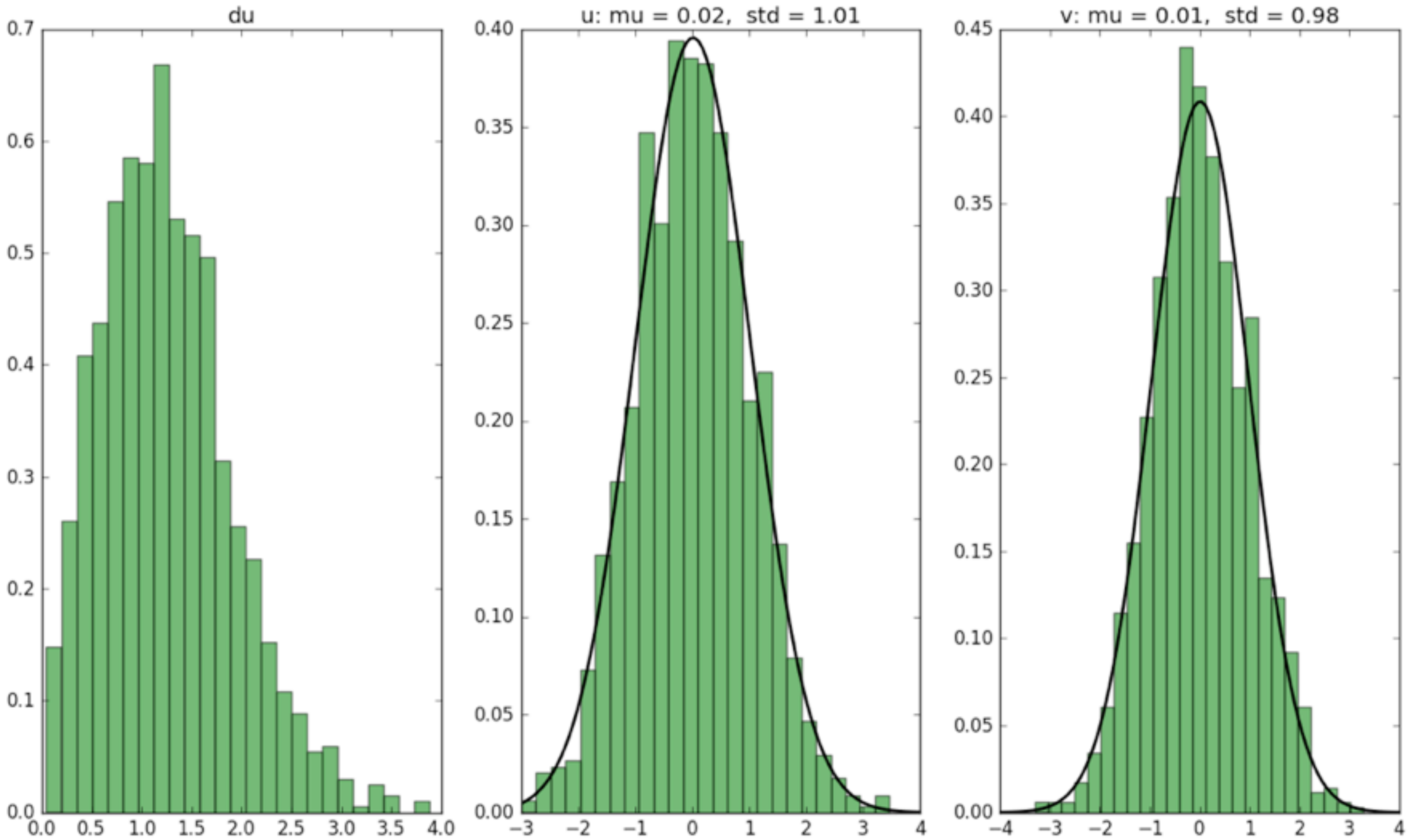
For additional information and comments, see section 4.

Look at the results

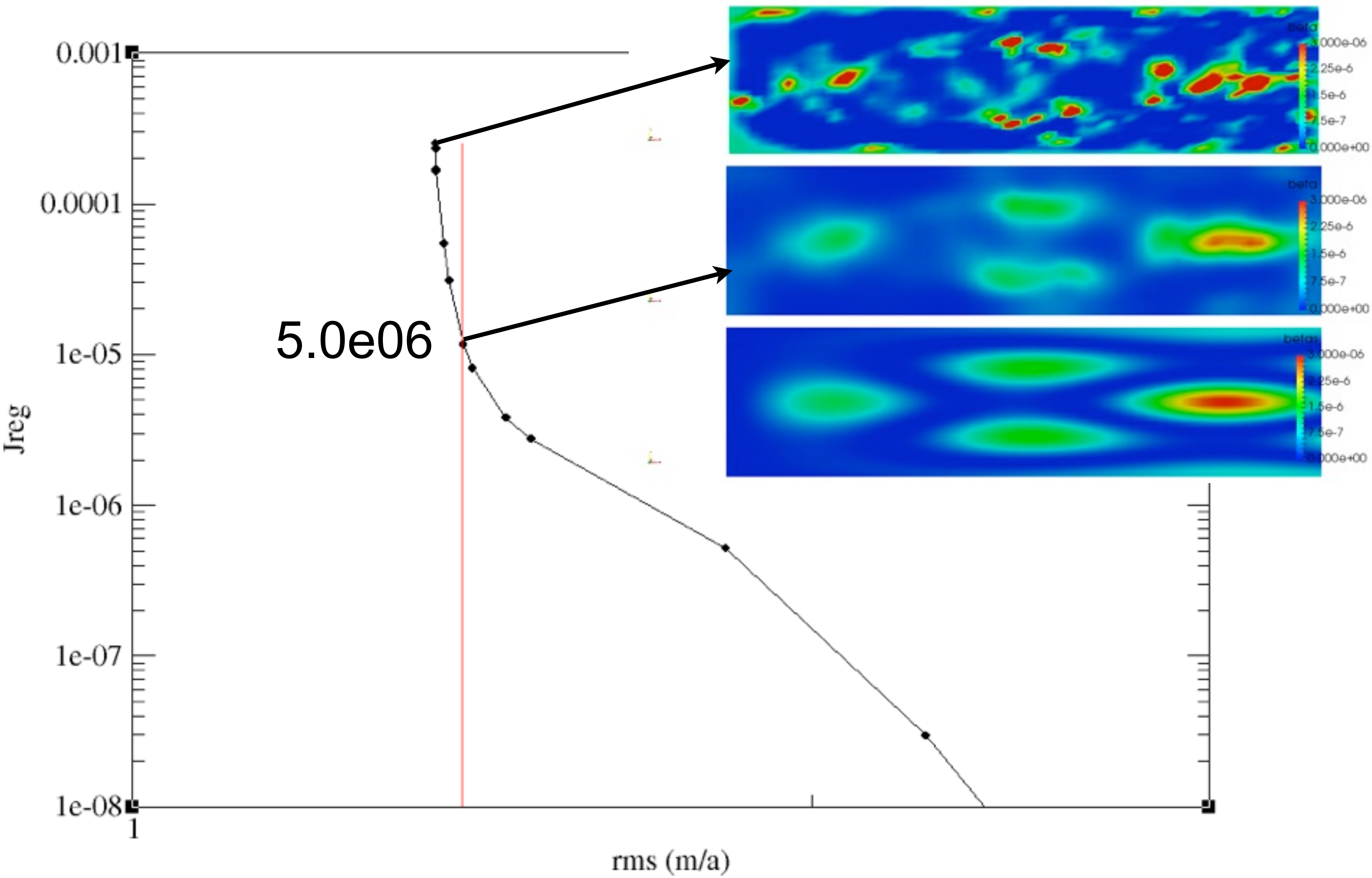




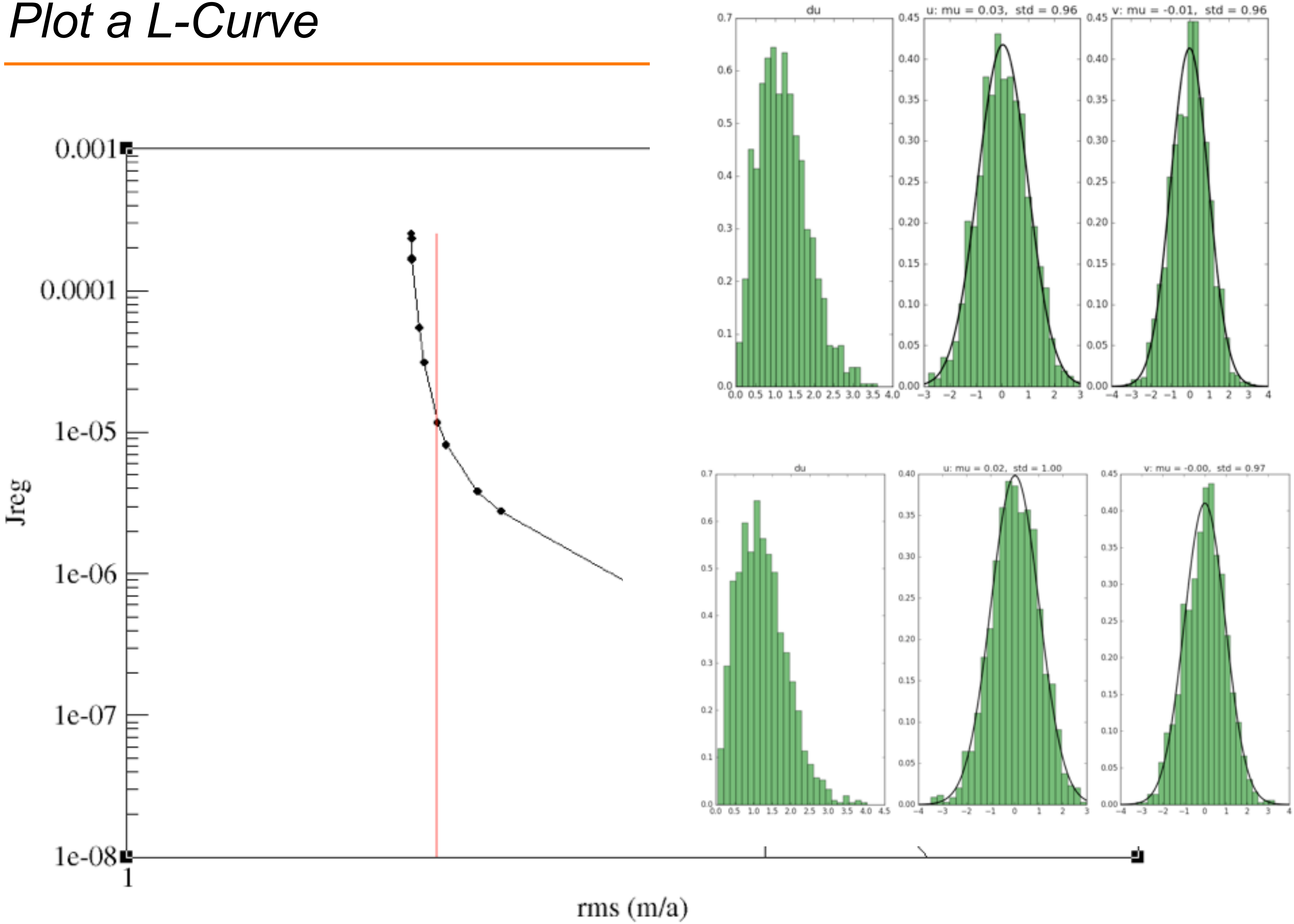
Add noise in the data



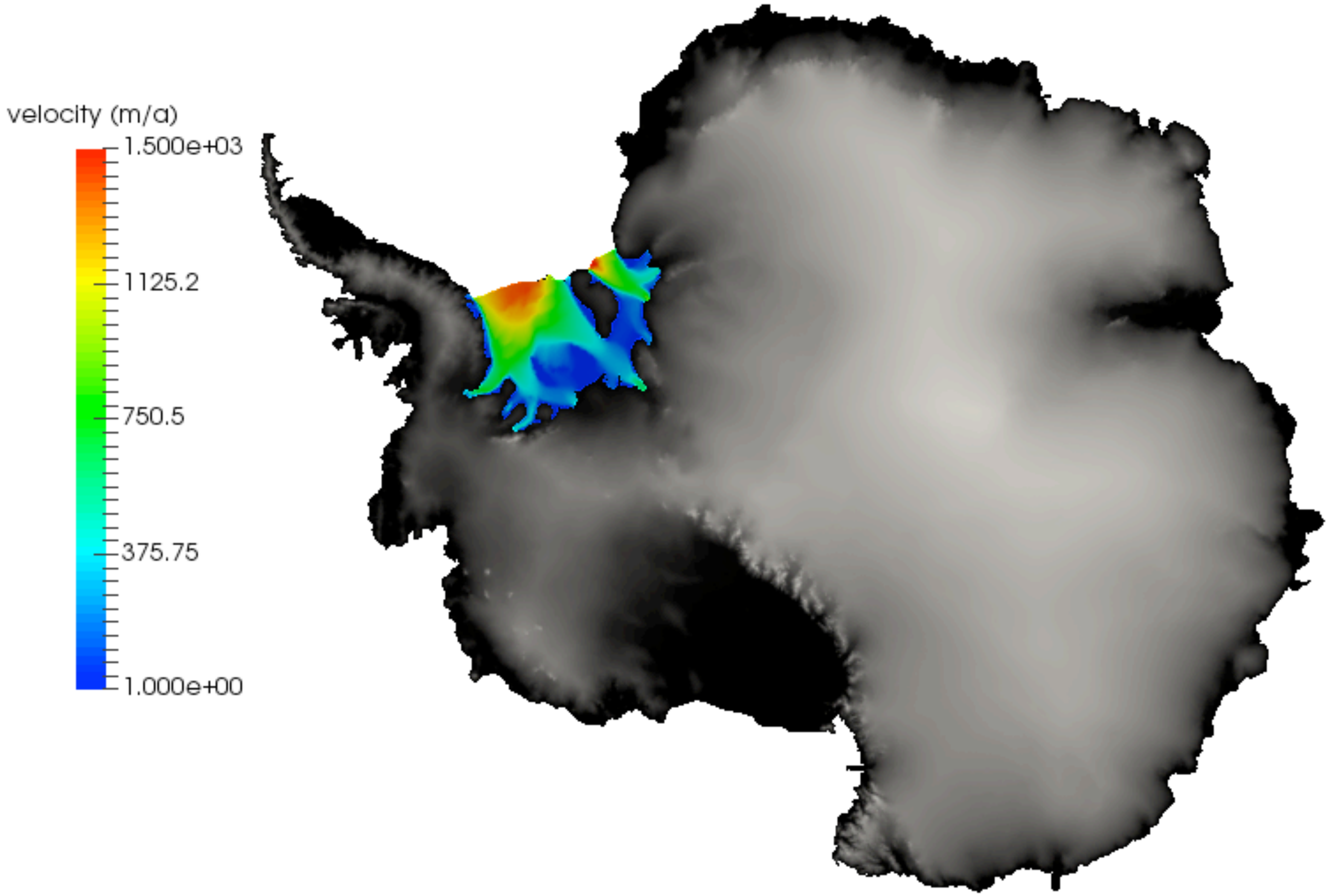
Plot a L-Curve



Plot a L-Curve



Model Ronne-Filchner Ice Shelf



Initialisation

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Simulation
  Coordinate System  = Cartesian
  Simulation Type = Steady State

  Steady State Min Iterations = 1
  Steady State Max Iterations = 200

  Post File = "OPTIM_$name$.vtu"

  Restart File = "IMPORT.result"
  Restart Before Initial Conditions = logical True

  max output level = 3
End
```

I have already imported data required for the computation

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Initial Condition 1
! alpha is the optimised variable
 alpha = Variable Mu
    REAL MATC "sqrt(tx)"
End
```

Optimise mean viscosity

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Material 1

  Viscosity Exponent = Real $1.0e00/3.0e00
  Critical Shear Rate = Real 1.0e-10

  SSA Mean Density = Real $rhoi
  SSA Mean Viscosity = Variable alpha
      REAL procedure "Adjoint_USFs" "Asquare"
  SSA Friction Law = String "linear"
  SSA Friction Parameter = Real 0.0
End
```

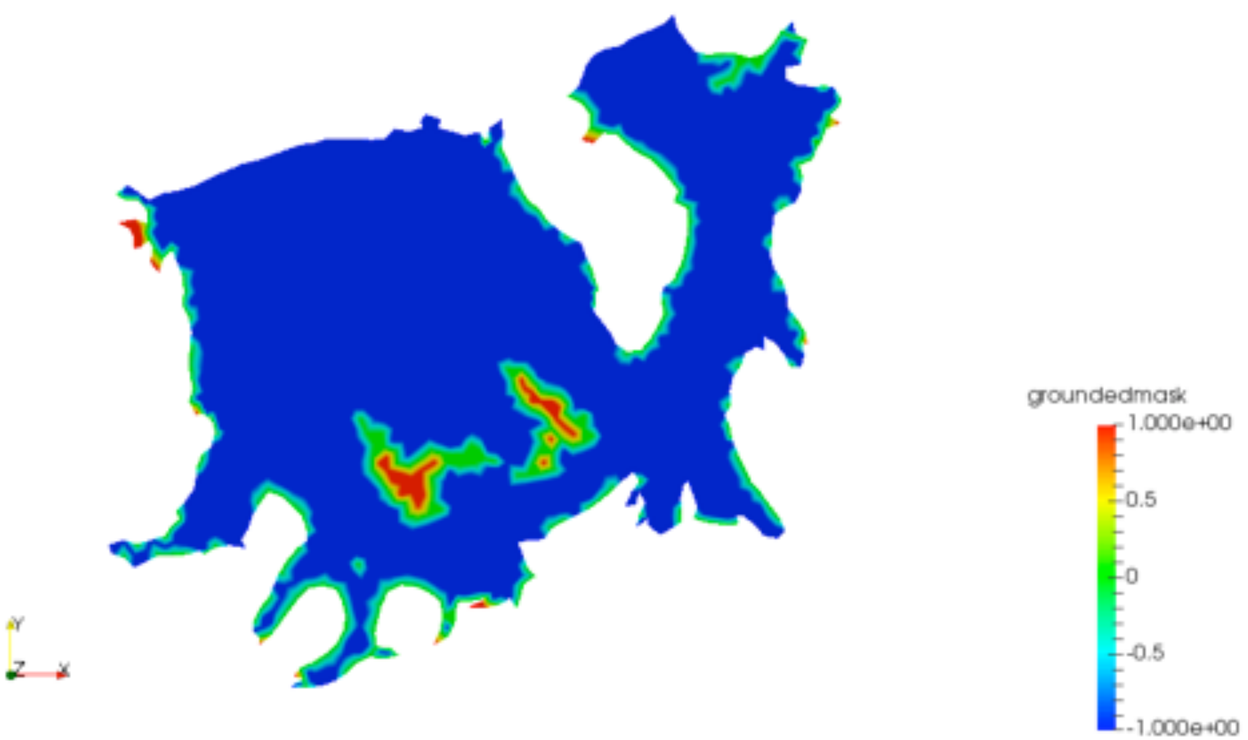
```
!!!!!  Compute Derivative of Cost function / Beta
Solver 4
  Equation = "DJDEta"

  !!  Solver need to be associated => Define dumy variable
    Variable = -nooutput "DJDB"
    Variable DOFs = 1

    procedure = "AdjointSSASolvers" "AdjointSSA_GradientSolver"

    Flow Solution Name = String "SSAVelocity"
    Adjoint Solution Name = String "Adjoint"
    Compute DJDEta = Logical True   ! Derivative with respect to the SSA Mean Viscosity

end
```

Boundary conditions

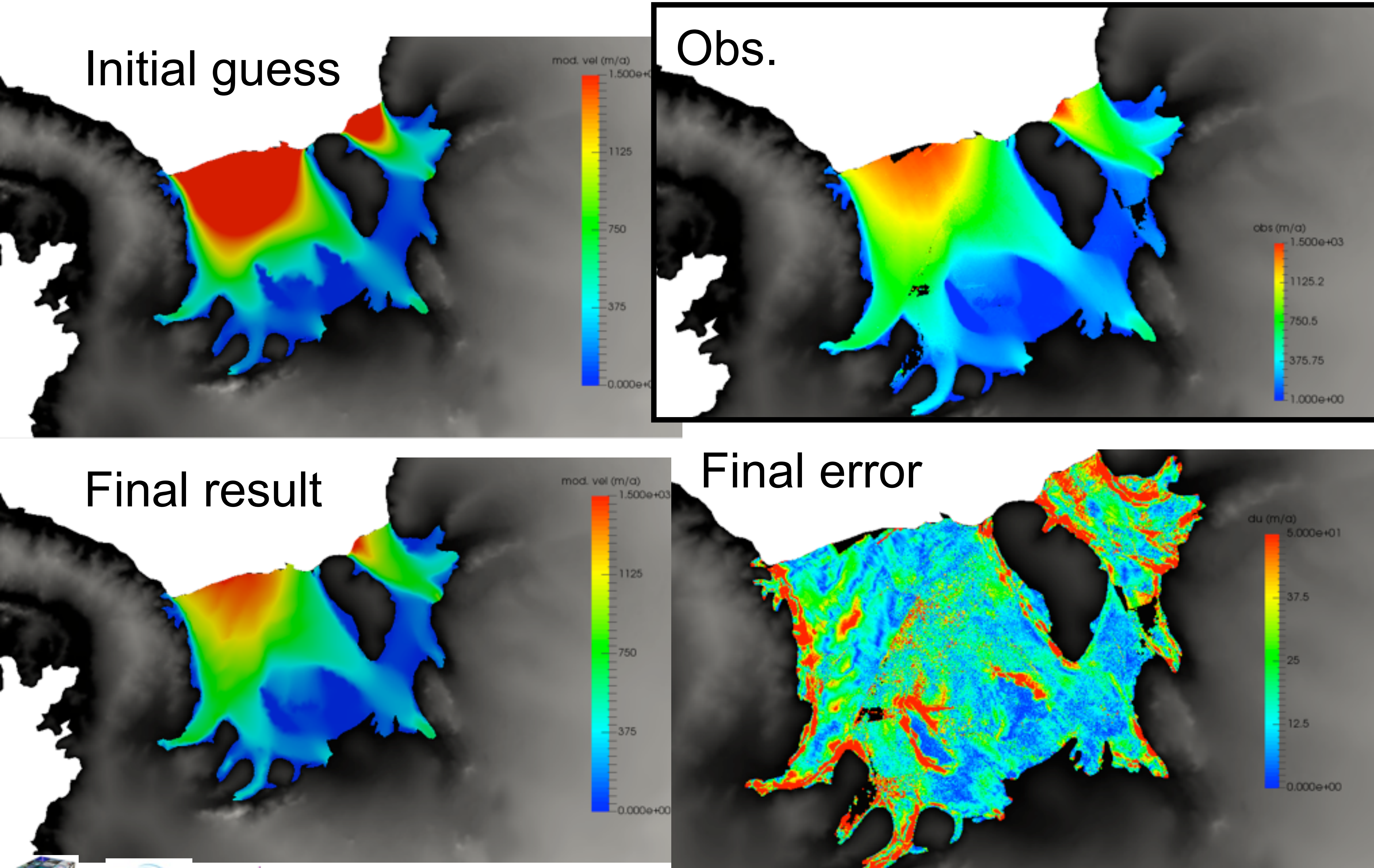
```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Boundary Condition 1
  Target Boundaries(2) = 1 3

  SSAVelocity 1 = Equals Uobs 1
  SSAVelocity 2 = Equals Uobs 2

  Adjoint 1 = Real 0.0
  Adjoint 2 = Real 0.0
End

Boundary Condition 2
  Name = "Ice Front"
  Target Boundaries(2) = 2 4

  calving front = logical true
End
```

Apply Dirichlet condition in grounded parts

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

  DJDalpha = Variable DJDEta , alpha
     REAL procedure "Adjoint_USFs" "Derivative_Asquare"

  SSAVelocity 1 = Equals Uobs 1
  SSAVelocity 2 = Equals Uobs 2
  SSAVelocity 1 Condition = Variable GroundedMask
    Real procedure "USFs_RonneFilchner" "GM_CONDITION"
  SSAVelocity 2 Condition = Variable GroundedMask
    Real procedure "USFs_RonneFilchner" "GM_CONDITION"

  Adjoint 1 = Real 0.0
  Adjoint 2 = Real 0.0
  Adjoint 1 Condition = Variable GroundedMask
    Real procedure "USFs_RonneFilchner" "GM_CONDITION"
  Adjoint 2 Condition = Variable GroundedMask
    Real procedure "USFs_RonneFilchner" "GM_CONDITION"

End
```

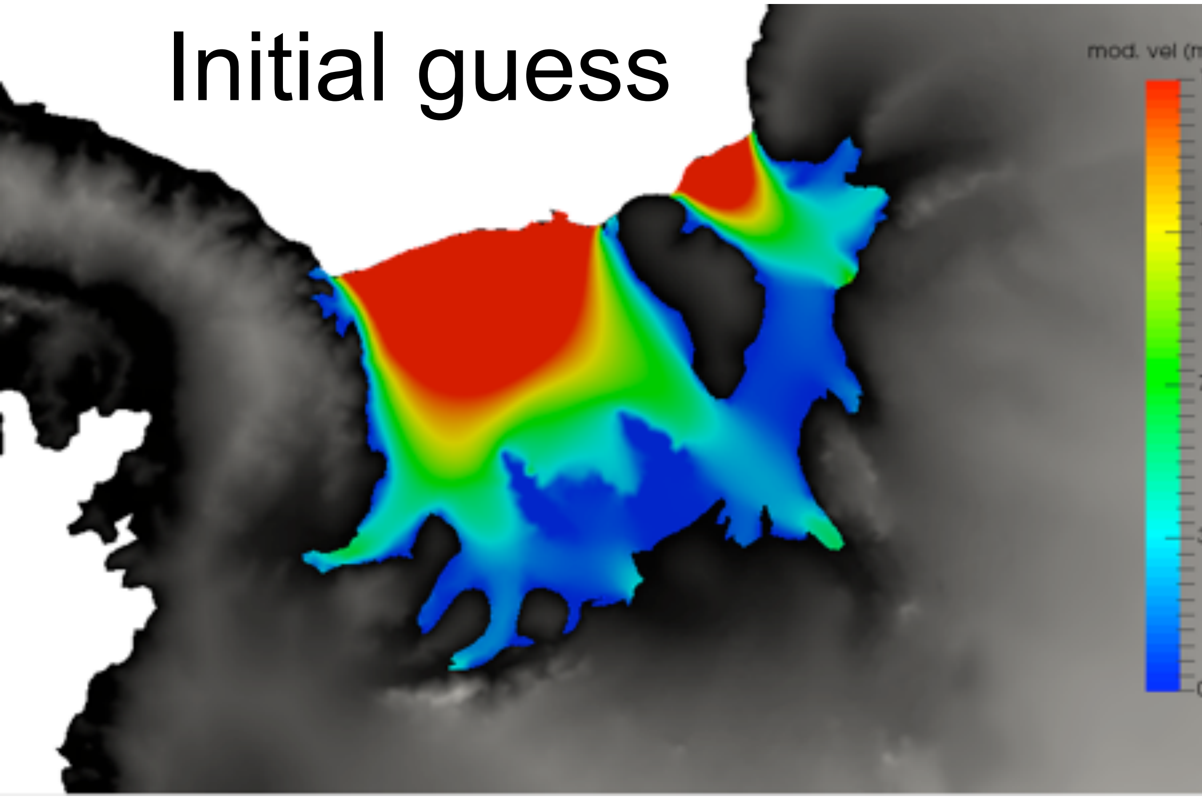


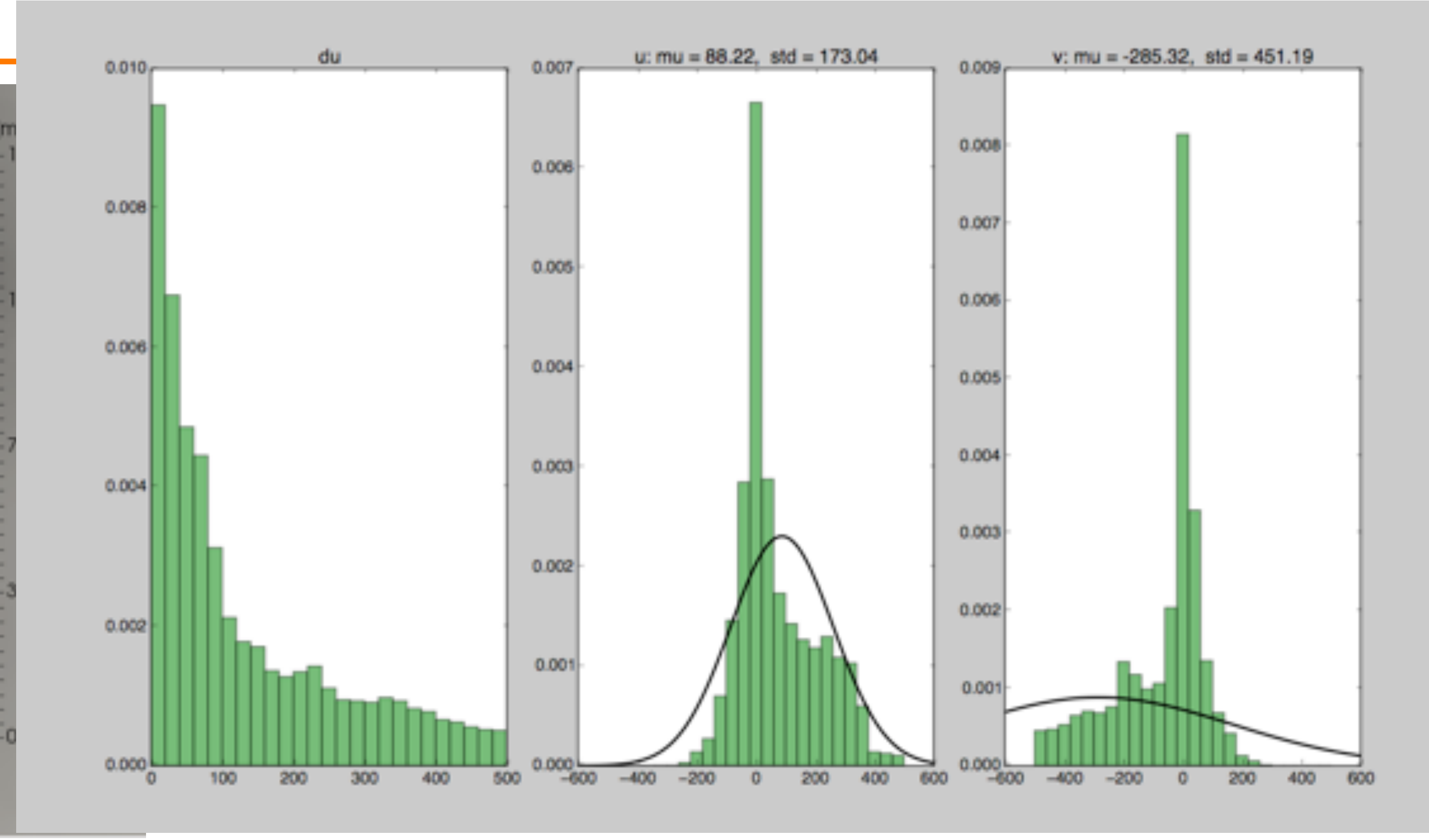
Results


Initial guess
mod. vel (m/a)
1.500e+0
1125
750
375
0.000e+0
Obs.
obs (m/a)
1.500e+03
1125.2
750.5
375.75
1.000e+00
Final result
mod. vel (m/a)
1.500e+03
1125
750
375
0.000e+00
Final error
du (m/a)
5.000e+01
37.5
25
12.5
0.000e+00

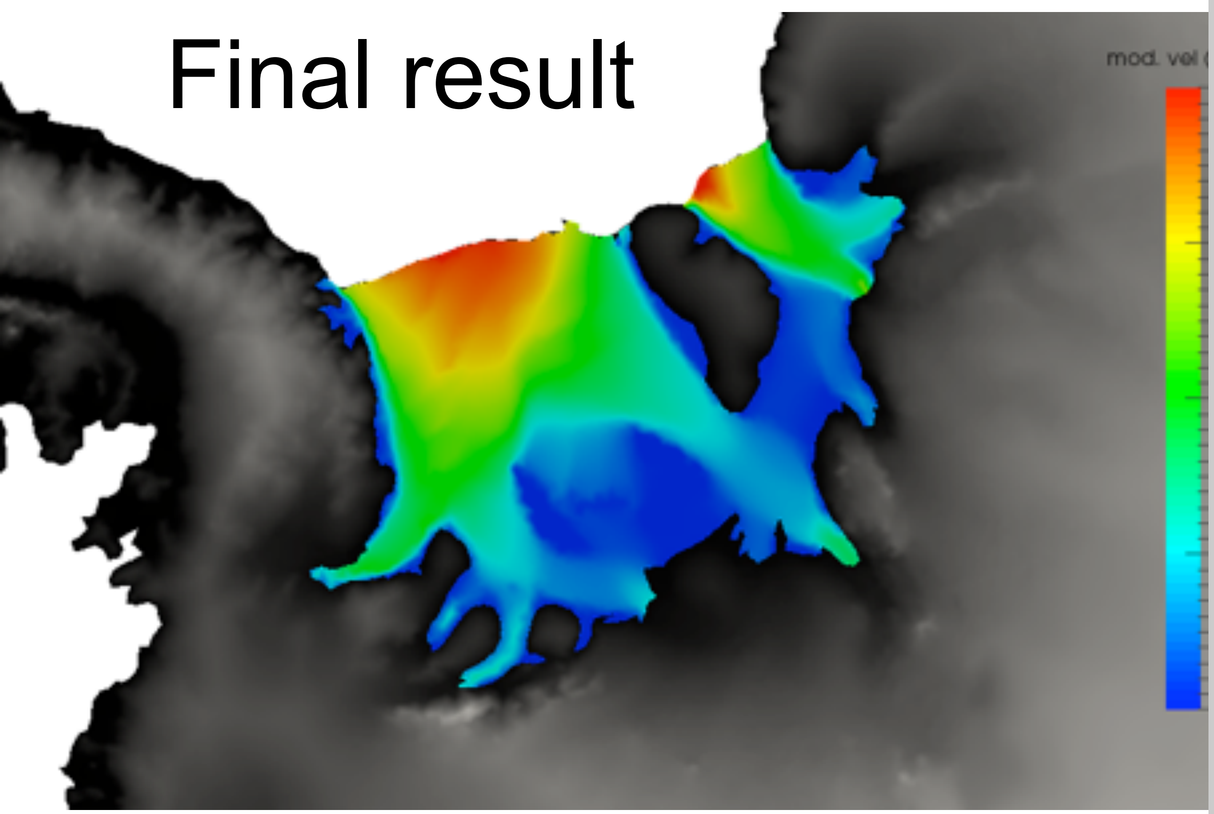
Results

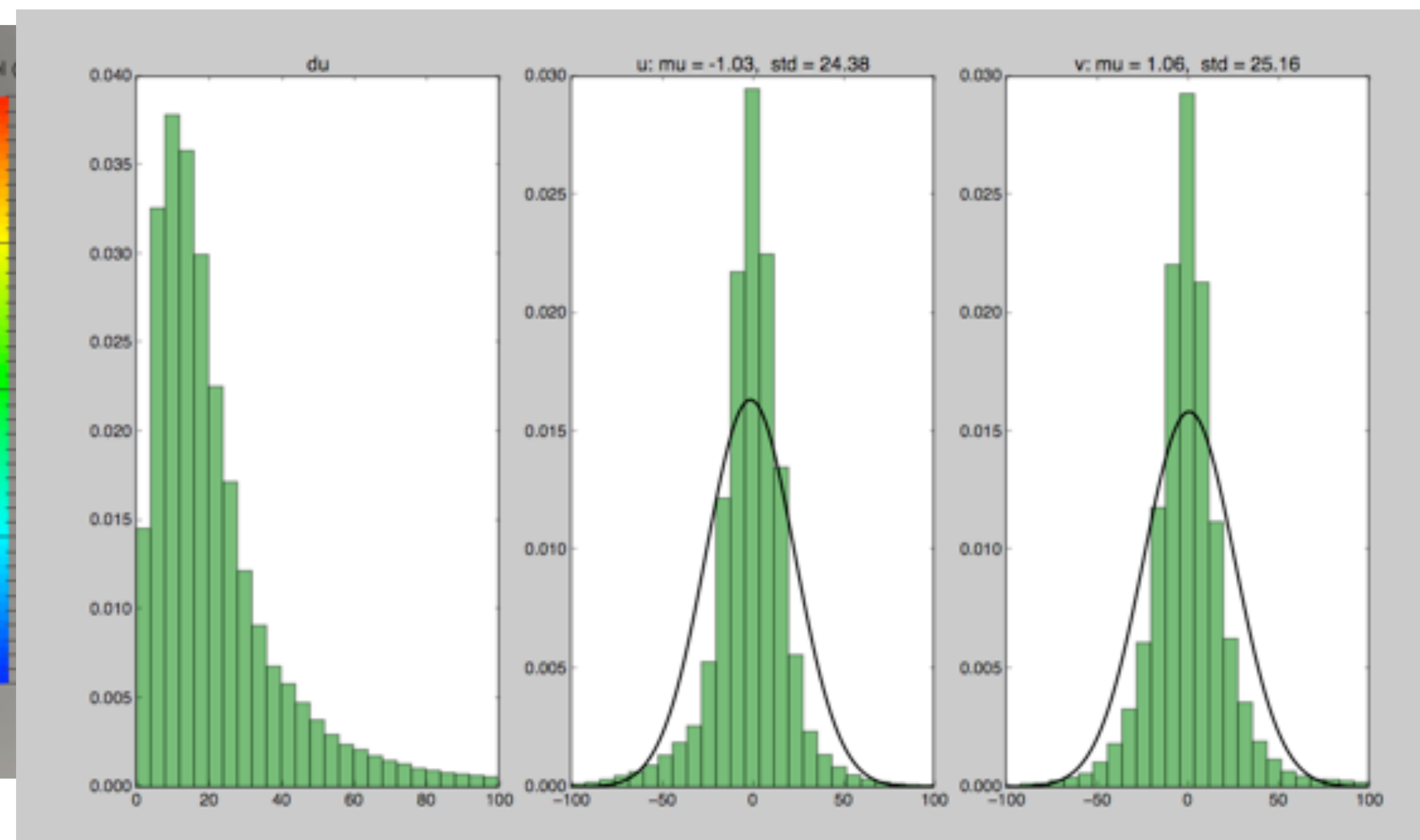
Initial guess



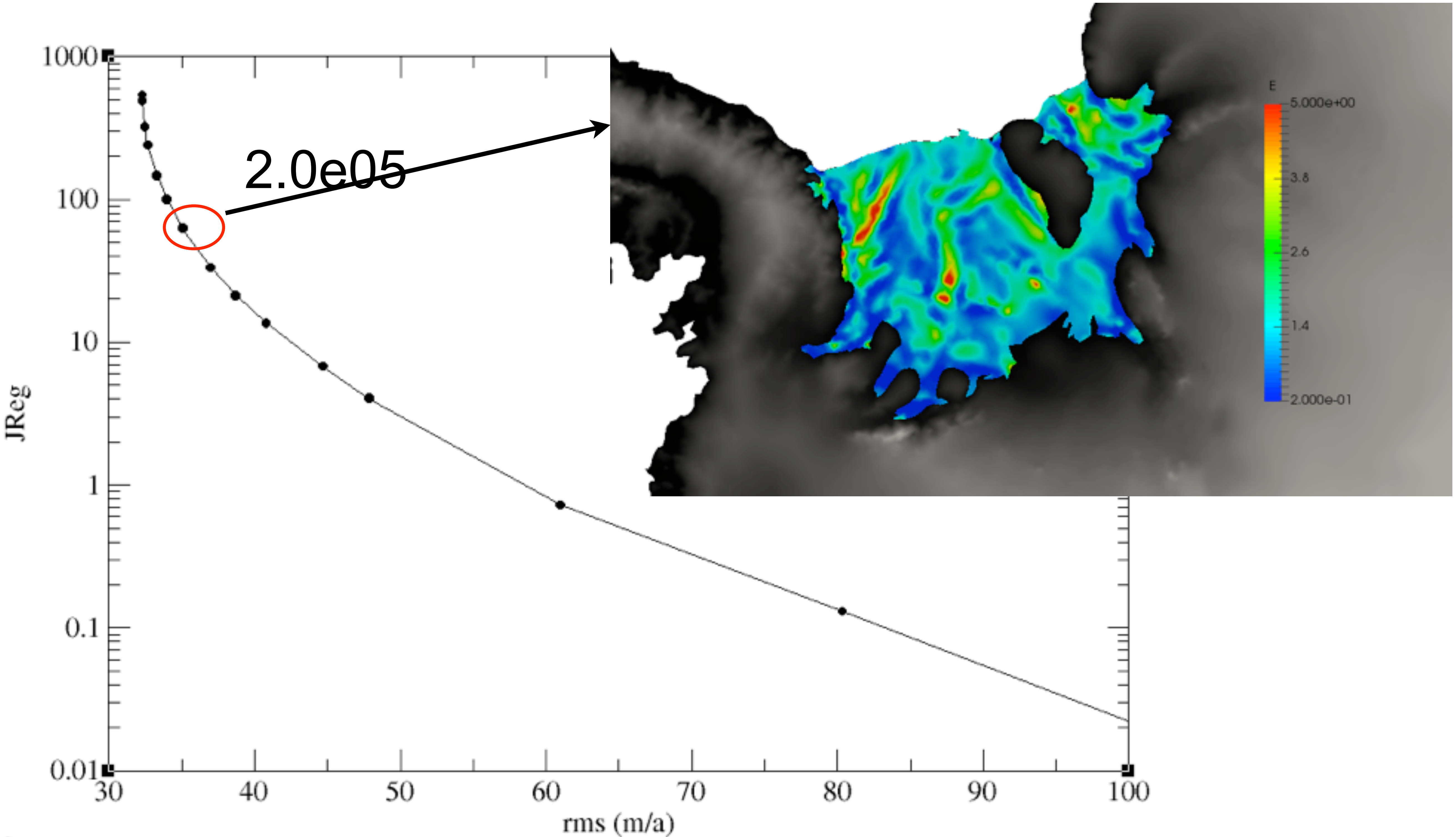


Final result





Results



2.0e05
JReg
rms (m/a)
1000
100
10
1
0.1
0.01
30
40
50
60
70
80
90
100
E
5.000e+00
3.8
2.6
1.4
2.000e-01

Inverse methods in Elmer: Thickness Solver

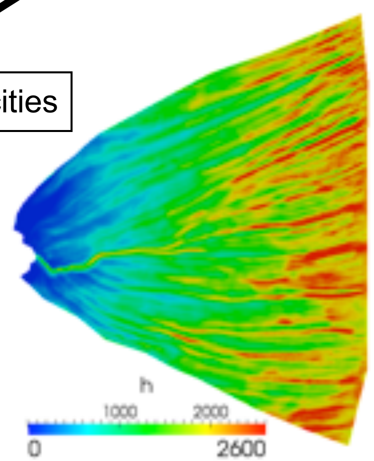
see Morlighem *et al.*, 2011, **a mass consservation approach for mapping glacier ice thickness**

1. Compute the blance thickness

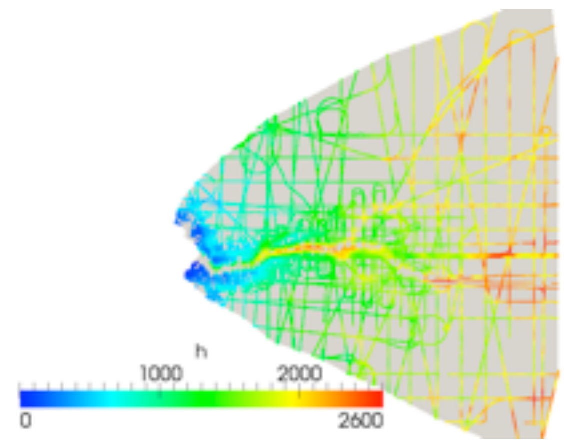
$$\nabla(\bar{u}H) = a_s$$

Effective smb (model smb+dhdt)

Observed surface velocities

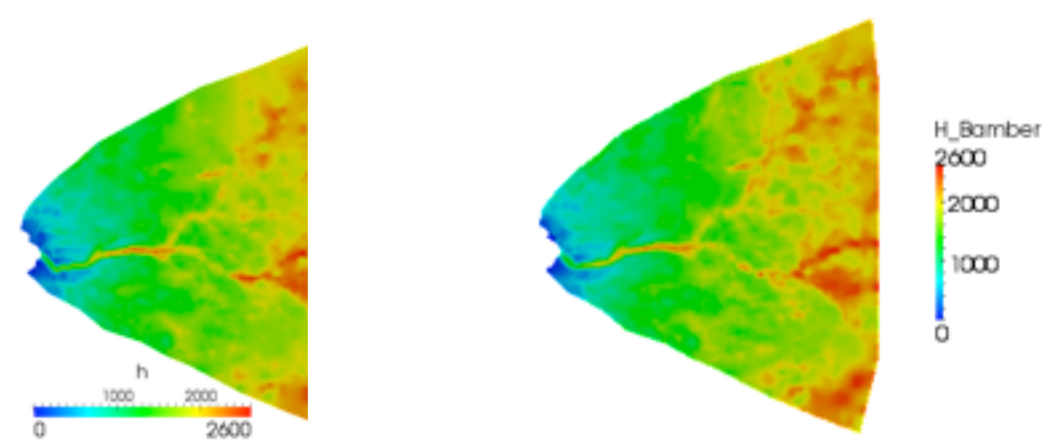


2. Compare with observations



$$J(\bar{u}, a_s) = \sum_{1}^{N_{obs}} 0.5(H - H_{obs})^2 + \lambda J_{Reg}$$

3. Use the adjoint to optimise u and a and reduced mismatch



Inverse methods in Elmer: Thickness Solver

- Work already done
- See presentation by J. Fürst for more details
- Still requires some cleaning/re-arrangement before submission to the Elmer/Ice distrib.