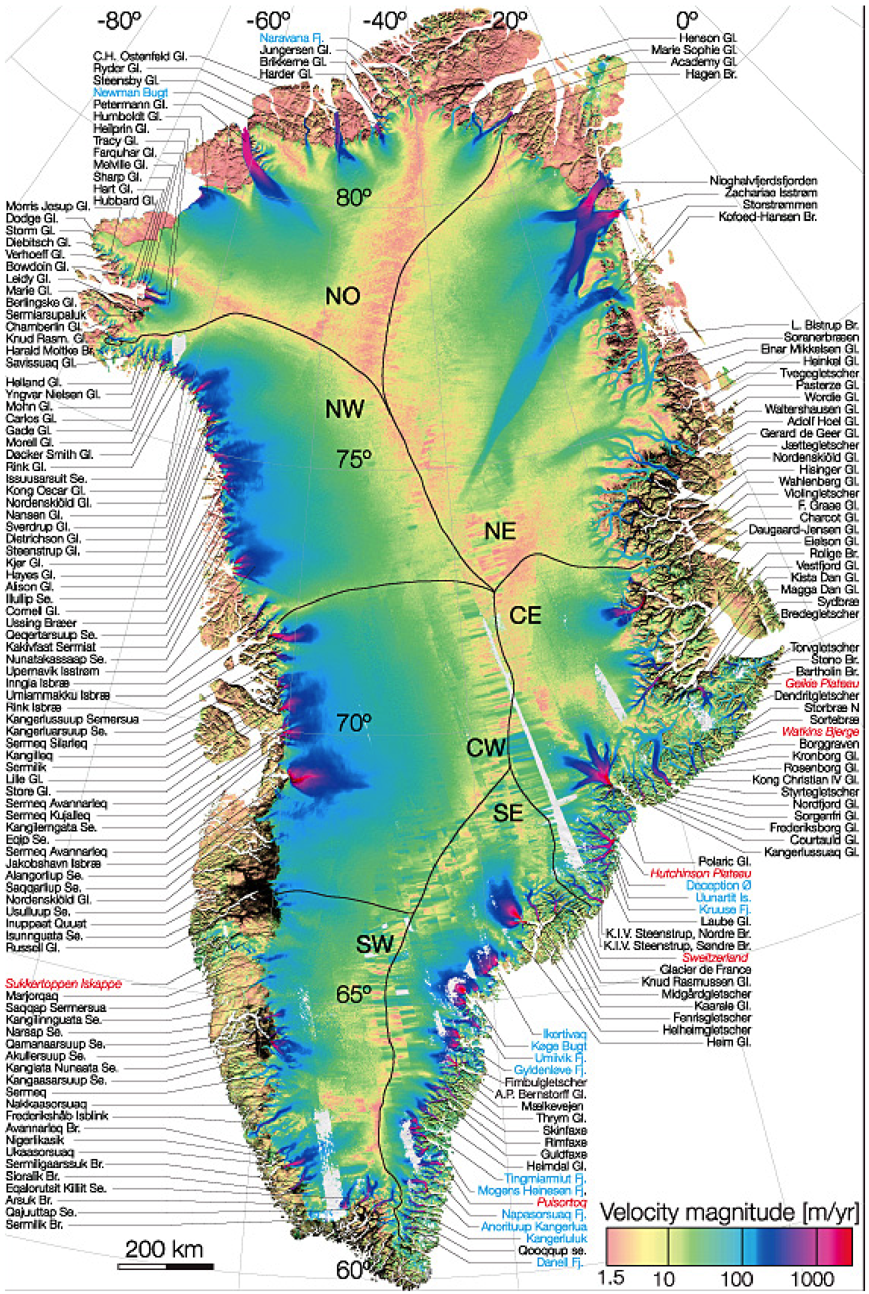
Anisotropic mesh adaptation using YAMS

GILLET-CHAULET Fabien

Laboratoire de Glaciologie et Géophysique de l'Environnement

CNRS/UJF-Grenoble

Large domain and highly heterogeneous flow



Meshing the 2D Greenland ice sheet footprint with GMSH

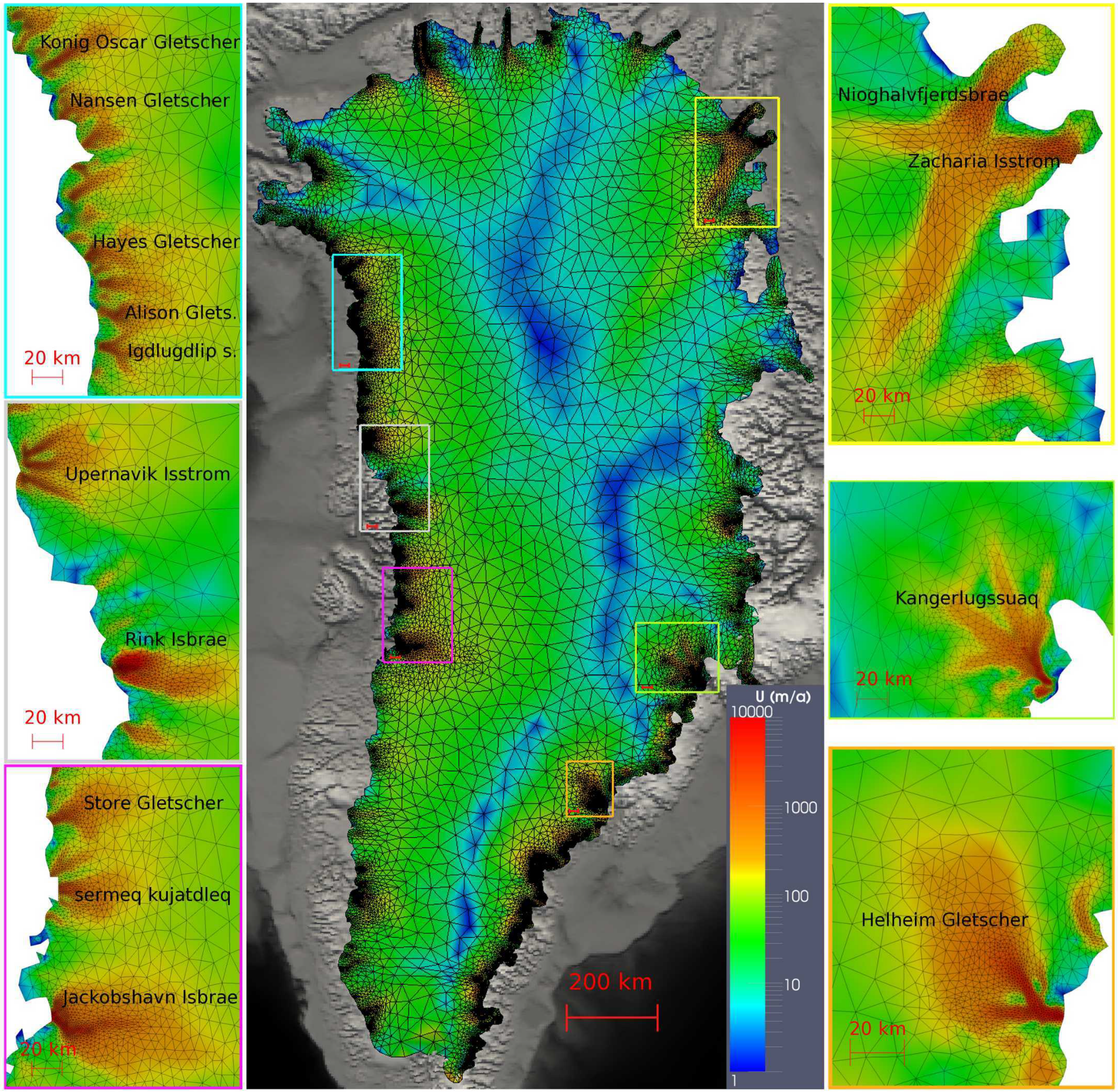
Mesh resolution (km)	Number of Nodes
50	4766
25	10304
15	21119
10	36001
5	114719
2.5	420262
1	2402834

x20 vertical layers x 4dofs/node =~190.10^6 dofs

Done on 2000 processors using the efficient Parallel Stokes solver …. but still require few hours for 1 non-linear stokes resolution.

Mesh adaptation required to reduce computational cost without reducing accuracy

Application to the Greenland Ice Sheet



source : Gillet-Chaulet et al., 2012

- Anisotropic mesh adaptation
  - minimum 1km in the outlets
  - maximum 30km in the central part

- In 2D : ~26 000 nodes
- In 3D : Extruded in 16 layers
  ~420 000 nodes x 4 ddl

- 48 partitions (processors MPI)

YAMS: a free-software for anisotropic mesh adaptation

- **Mesh adaptation:** mesh size adapted to control/equi-distribute the approximation error

- Here we present how to use **YAMS**

- **References:**
  - Frey and Alauzet, 2005, Anisotropic mesh adaptation for CFD computations, Comput. Methods Appl. Mech. Engrg. 194
  - http://www.ann.jussieu.fr/frey/software.html

5078 P.J. Frey, F. Alauzet / Comput. Methods Appl. Mech. Engrg. 194 (2005) 5068–5082

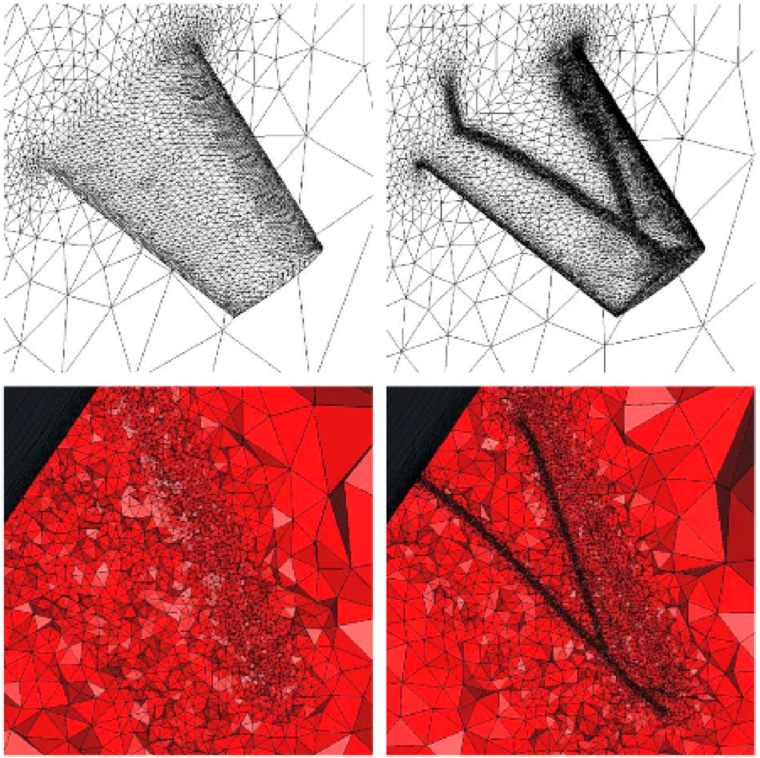


Fig. 2. Onera M6 wing test case: isotropic surface and cut through the volume mesh at iterations 1 and 9 of the adaptation scheme.

YAMS: a free-software for anisotropic mesh adaptation

- **Approximation error** estimate based on the **interpolation error**
- Interpolation error bounded using second derivatives of the computed field.

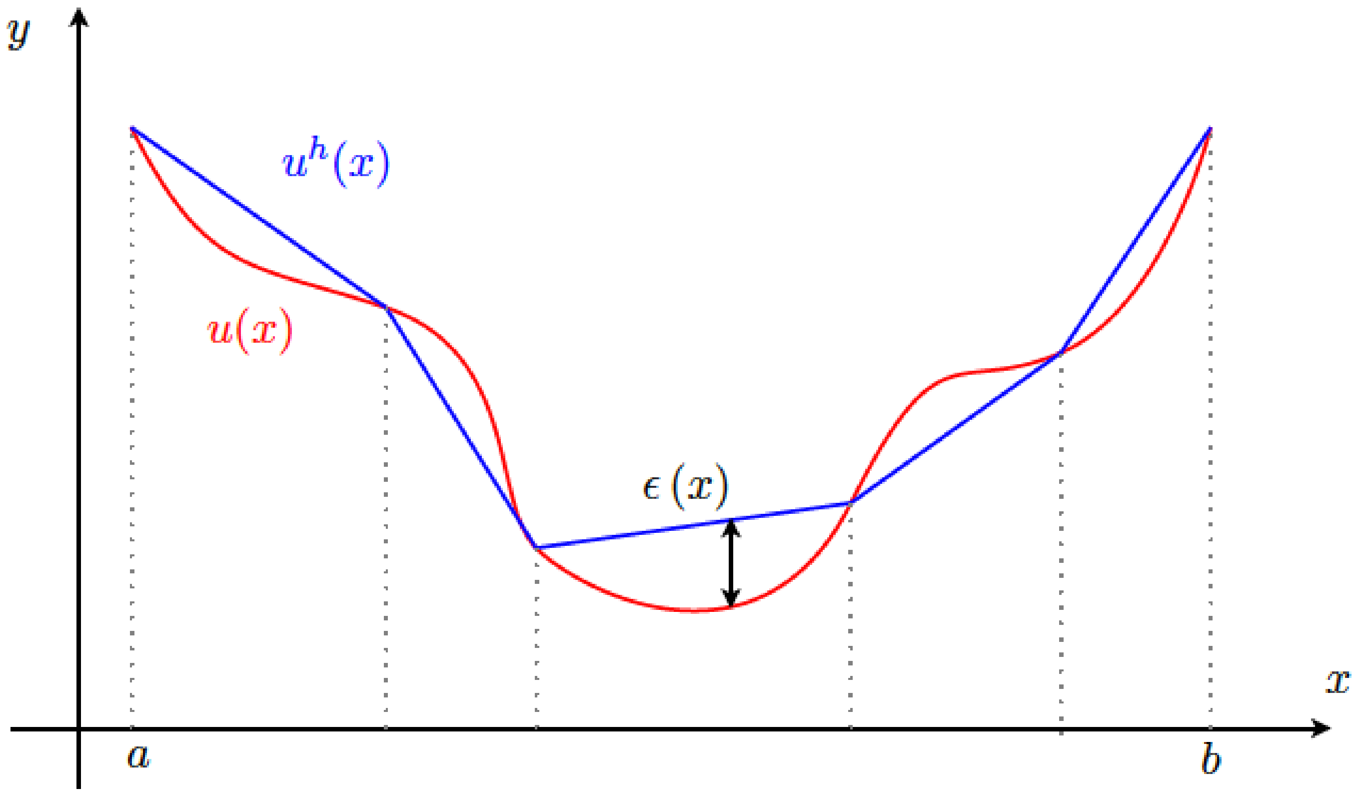


Figure 3.7: 1d interpolation error $\epsilon(x) = \left|u(x) - u^h(x)\right|$

Source: Morlighem,PhD Thesis, 2011

YAMS: a free-software for anisotropic mesh adaptation

- The **mesh size** adapted according to the given **anisotropic metric field** $\boldsymbol{M}$

$$\mathscr{M} = \mathscr{R}\widetilde{\Lambda}\mathscr{R}^{-1}, \text{ with } \widetilde{\Lambda} = \begin{pmatrix} \widetilde{\lambda}_1 & 0 & 0 \\ 0 & \widetilde{\lambda}_2 & 0 \\ 0 & 0 & \widetilde{\lambda}_3 \end{pmatrix},$$

$$\widetilde{\lambda}_i = \min\left(\max\left(\frac{c|\lambda_i|}{\varepsilon}, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right),$$

YAMS and ELMER

- YAMS is a stand alone program

- **Pre-processing step** (i.e. not included in Elmer)

  => **build a framework around YAMS for glaciological applications**
    - Mesh 2D Footprints of Ice sheet, drainage basins, or glaciers
    - Observed velocities (Hessian matrix) used to construct the metric

- **Home-made routines and programs** in fortran to:
  - Construct the initial mesh (GMSH)
  - Construct the metric
  - Run YAMS
  - Convert to Elmer format

    => [Under course material]/MeshAdaptation/src

- **Pre-requisites :**
  - Have GMSH
  - Have ElmerGrid
  - Have fortran/C compilers
  - **Install and compile YAMS and medit (http://www.ann.jussieu.fr/frey/ftp/archives/)** (have **yams** and **medit** executables in your **PATH**)

YAMS and ELMER

- **Algorithm:**

1) Create a first mesh from the **contour of the domain** using **GMSH**

=> Create the **.geo** file directly => cf Case0_Gaussian
=> Create the **.geo** file from a contour of your domain => cf Case1 and Case2 (MakeGeo_Greenland_IceSheet.f90 / MakeGeo_Greenland_Basin.f90)

=> Initial .geo file : Contour.geo

YAMS and ELMER

- **Algorithm:**

1) Create a first mesh from the contour of the domain using GMSH

2) **Convert to 2D mesh format**

=> GeoToMesh.f90 (call to gmsh to create the .mesh from .geo and do some format transformation)

=> Initial mesh file: mesh2D_1.mesh

YAMS and ELMER

- **Algorithm:**

Make2DMesh.f90:

3) Read Input File "**input.txt**" (Fortran NAMELIST)

4) Read **observed surface velocities** DEM → READ_DEM...f90

↓ Initialisation.f90

5) Compute the **Hessian matrix** of the observed velocities at each mesh node → biquad.f90
   - This is done by computing the coefficients of a quadratic function $v=ax^2+by^2+cxy+dx+ey+f$ that best fit (least-square) the observations within a circle of radius R

6) Create the metric according to user input parameters: err, hmin and hmax (2 sets of values possible with a criteria on the velocity magnitude (treshold); Max mesh size is used for No data areas) → Metric.f90

↓ CreateSol.f90

$$\mathcal{M} = \mathcal{R}\widetilde{\Lambda}\mathcal{R}^{-1}, \text{ with } \widetilde{\Lambda} = \begin{pmatrix} \widetilde{\lambda}_1 & 0 & 0 \\ 0 & \widetilde{\lambda}_2 & 0 \\ 0 & 0 & \widetilde{\lambda}_3 \end{pmatrix}, \quad \widetilde{\lambda}_i = \min\left(\max\left(\frac{c|\lambda_i|}{\varepsilon}, \frac{1}{h^2_{\max}}\right), \frac{1}{h^2_{\min}}\right),$$

YAMS and ELMER

- **Algorithm:**

Make2DMesh.f90:

↓ Initialisation.f90

3) Read Input File "**input.txt**" (Fortran NAMELIST)

4) Read **observed surface velocities** DEM → READ_DEM...f90

↓ CreateSol.f90

5) Compute the **Hessian matrix** of the observed velocities at each mesh node
   - This is done by computing the coefficients of a quadratic function $v=ax^2+by^2+cxy+dx+ey+f$ that best fit (least-square) the observations within a circle of radius R → biquad.f90

6) Create the metric according to user input parameters: err, hmin and hmax (2 sets of values possible with a criteria on the velocity magnitude (treshold); Max mesh size is used for No data areas) → Metric.f90

=> Initial mesh metric mesh2D_1.sol (mesh2D_1.vel contains the velocity variable)

YAMS and ELMER

- **Algorithm:**

Make2DMesh.f90:

=> Initial mesh **mesh2D_1.mesh + mesh2D_1.sol**

7)Run **YAMS** to produce a new mesh

8)Visualise the new mesh (**medit** or convert to **vtk (paraview)**) Bash script Yams2VTK.sh

9) Go back to 4) or convert to Elmer format (using **GMSH** format and **ElmerGrid**) → MeshToElmer.f90

YAMS and ELMER

- **Algorithm:**

1) Create a first mesh from the contour of the domain using **GMSH**

2) Convert to 2D mesh format

3) Read observed surface velocities DEM

4) Compute the Hessian matrix of the observed velocities at each mesh node
   - This is done by computing the coefficients of a quadratic function $v=ax^2+by^2+cxy+dx+ey+f$ that best fit (least-square) the observations within a circle of variable size R

5) Create the metric according to user input parameters: error, min and max mesh size (2 sets of values possible with a criteria on the velocity magnitude; Max mesh size is used for No data areas)

6) Run **YAMS** to produce a new mesh

7) Visualise the new mesh (**medit** or convert to **vtk (paraview)**)

8) Go back to 4) or convert to Elmer format (using **GMSH** format and **ElmerGrid**)

=> Require user intervention/coding

YAMS and ELMER

3 application test cases under

[Under course material]/MeshAdaptation/Case[1-3]

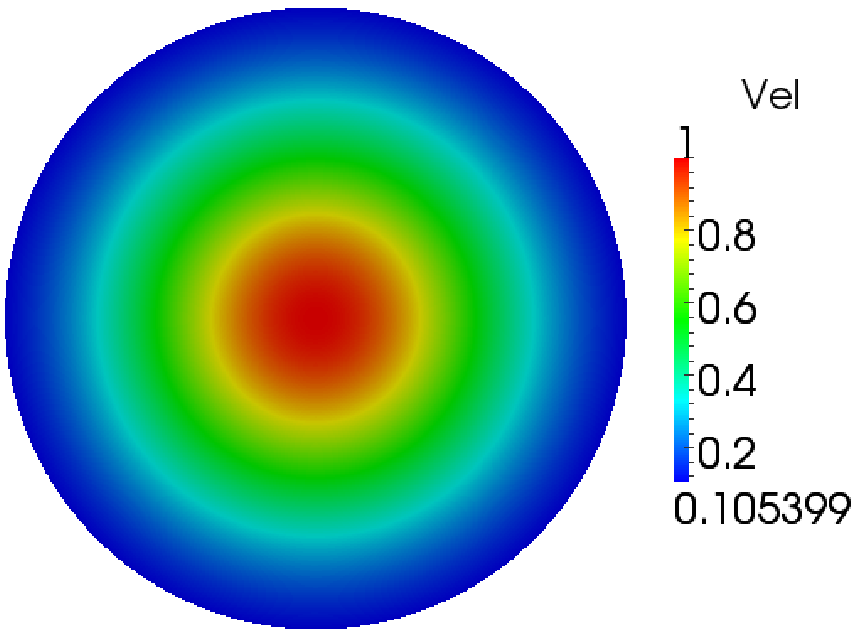
Case 0: a simple test case for illustration

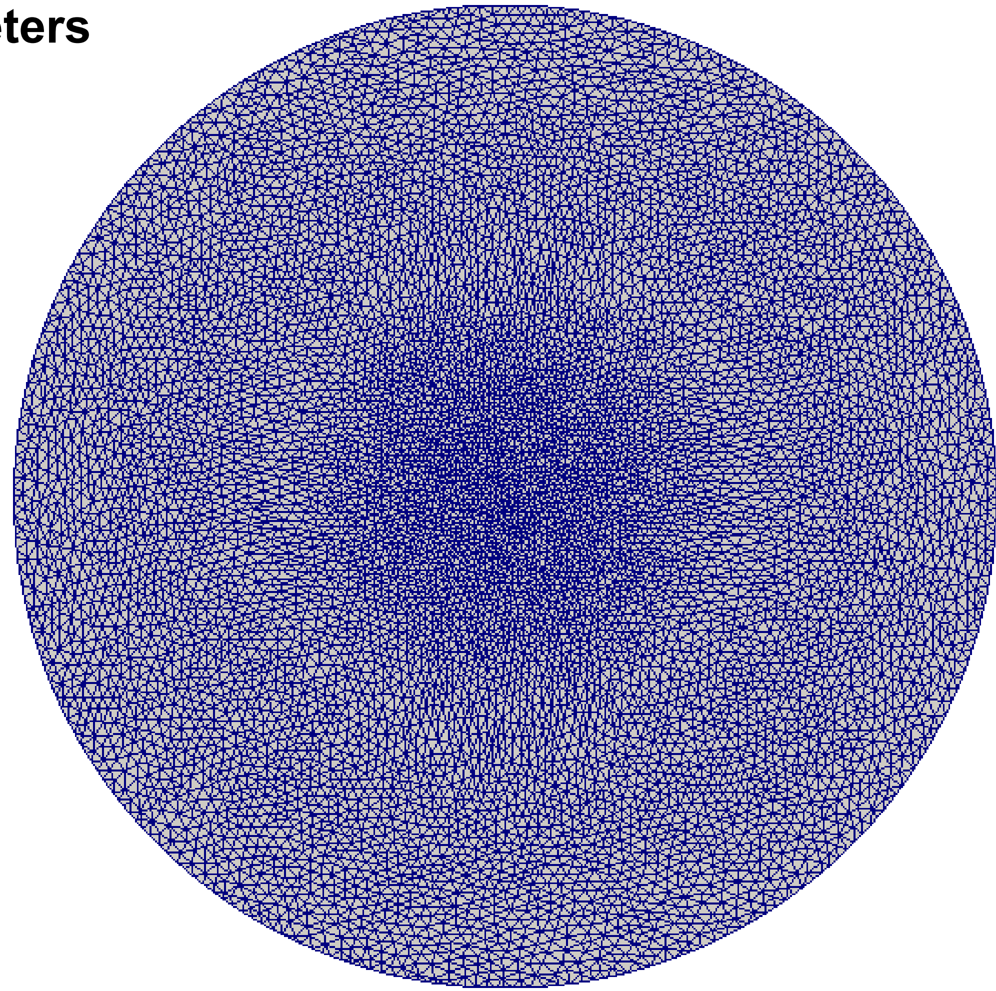
- Circular domain [Center (0;0) , radius 5.0e04]
- Ideal "velocities" given by a gaussian function ($\exp(-r^2/(1.0e05/3)^2)$)

=> To run the test:

> make

Look at the results, **change the input parameters**





Greenland meshes

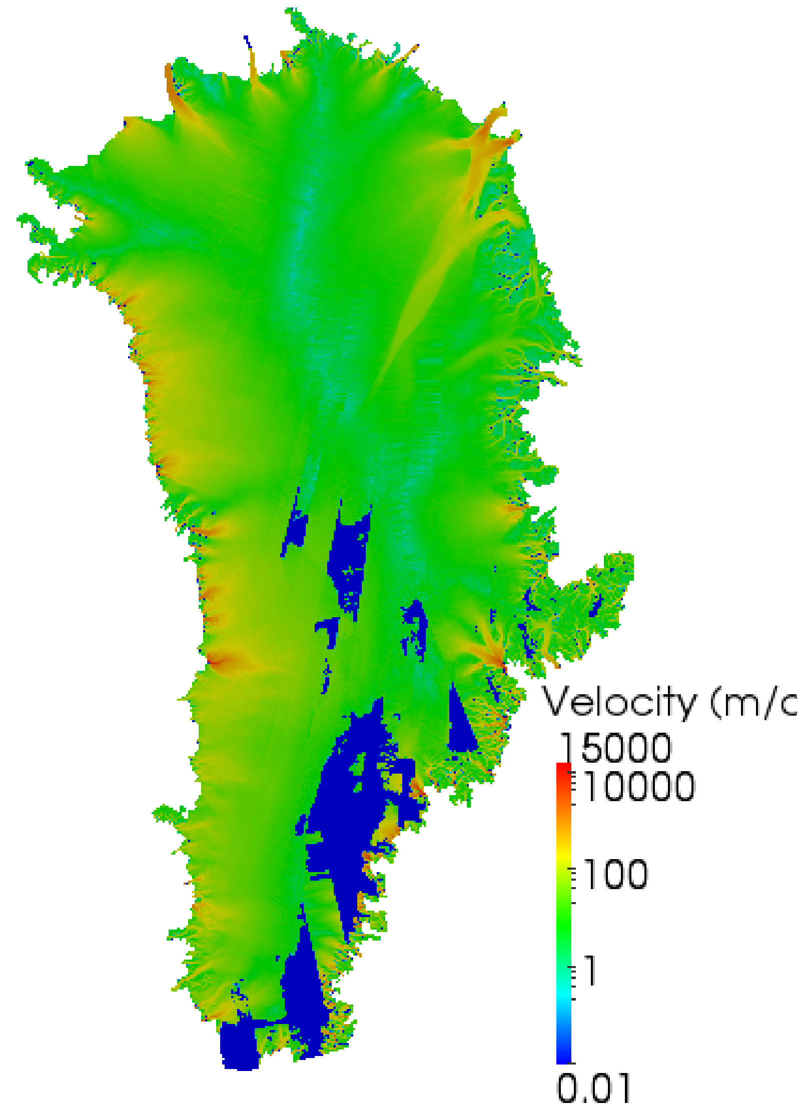
- Joughin, I., B. Smith, I. Howat, and T. Scambos. 2010. *MEaSUREs Greenland Ice Velocity Map from InSAR Data*. Boulder, Colorado, USA: National Snow and Ice Data Center. Digital media.

- Data sets for years 2000, 2005, 2006, 2007, 2008 available from NSIDC website http://nsidc.org/data/docs/measures/nsidc0478_joughin/

- DEM at 500mx500m resolution

- As each DEM has different areas with No Data => **an averaged DEM as been produced from the 5 data sets ([Under course material]/MeshAdaptation/Data)**

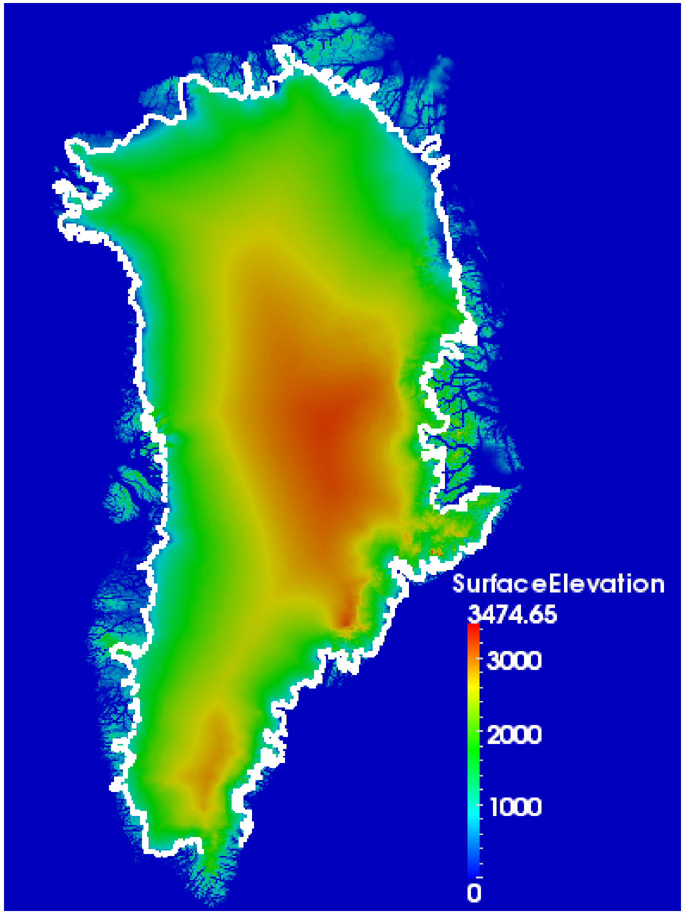
- !! Grid projection is polar stereographic true at 71N and meridian of origin 45W; but often topographic data (Bamber, SeaRise, Ice2sea datasets) use meridian 39W as the meridian of origin!!
  - **=> Contours points are re-projected in polar stereographic 71N/45W**
  - Mesh optimisation performed
  - **Final mesh re-projected in polar stereographic 71N/39W**



Case 1: Meshing the Greenland Ice Sheet

- **[Under course material]/MeshAdaptation/Case1_Greenland**
- Need an ascii file with ordered points forming a closed contour of the Greenland Ice Sheet (data, obtained from a 0m thickness contour (matlab, paraview, ….), extracted by hand, ….)

=> [Under course material]/MeshAdaptation/Data/Contour.dat



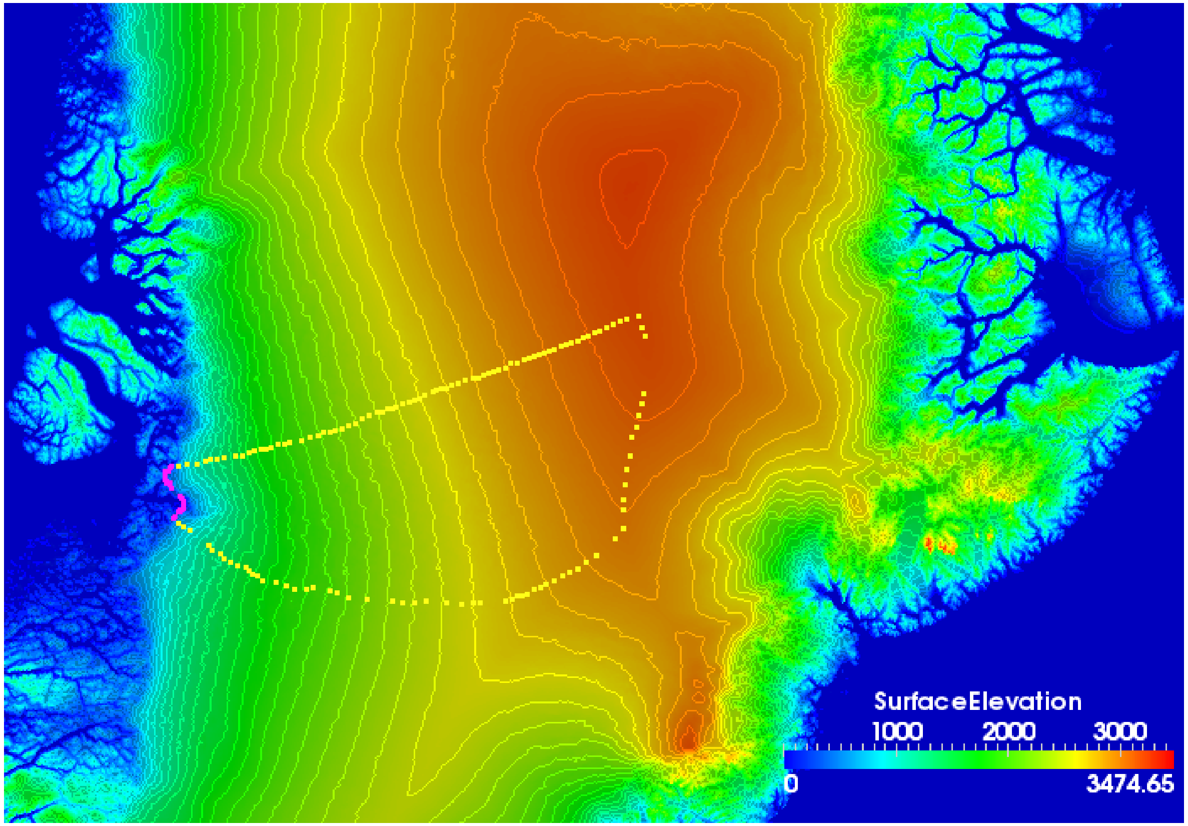
=> Produce a mesh with 30-40 10^3 nodes for the ParStokes application

Case 2: Meshing Jakobshavn drainage basin

- **[Under course material]/MeshAdaptation/Case1_Jakobsahvn**
- Need 2 ascii files with ordered points to define the contour of the drainage basin
  - 1 file for the margin (Neumann boundary conditions (water pressure at the calving front)

    => [Under course material]/MeshAdaptation/Data/ContourCote.dat
  - 1 file for the side of the drainage basin (artificial boundary condition => no flux)

    => [Under course material]/MeshAdaptation/Data/ContourSide.dat



=> Produce a mesh for the inverse methods application

Conclusion

- Should be relatively easy to use for your own applications
- Ask me for help if needed
- Please refer to Gillet-Chaulet et al., The Cryosphere, 2012 if you use these tools
- YAMS can also be used with a scalar metric to produce isotropic meshes with refined parts (e.g. near a grounding line; not demonstrated/programmed here)
- Perspectives: implement a similar mesh adaptation algorithm within an Elmer Solver?