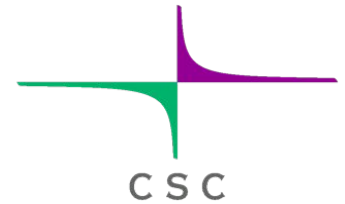




Laboratoire de Glaciologie et Géophysique de l'Environnement



First Elmer/Ice course

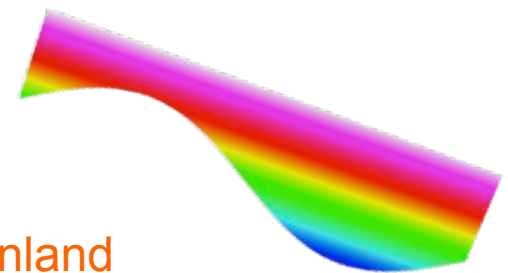
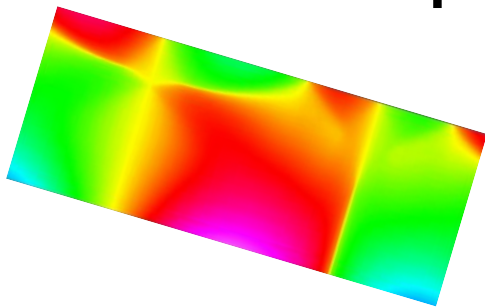
14-15 February 2008 – Updated 2012

Thomas ZWINGER ⁽¹⁾ and Olivier GAGLIARDINI ⁽²⁾

Application to ISMIP HOM⁽³⁾

tests B and D

Step by step !



(1) CSC-Scientific Computing Ltd., Espoo - Finland

(2) LGGE - Grenoble - France

(3) <http://homepages.ulb.ac.be/~fpattyn/ismip/>

Outline

- Step 0** Start from a very simple test case. What are we solving? (Glen' s law, ...)
- Step 1** Move to test ISMIP-HOM B020 (mesh, periodic BC)
- Step 2** Add SaveData solver to get output on the BC (SaveLine)
- Step 3** Add SaveData solver to get cpu and volume of the domain (SaveScalars)
- Step 4** Add ComputeDevStress solver to get the stress field
- Step 5** Move to test ISMIP-HOM D020 (sliding law from user function or MATC)
- Step 6** Restart from Step 4: Move to Prognostic ISMIP B020.
 - Move from a steady to a transient simulation
 - Free surface solver
 - Mesh Update solver
- Step 7** Move to Prognostic ISMIP D020.

Step 0

Create a `My_ISMIP_Appli` directory

Copy the directory *Step0* in `My_ISMIP_Appli`

- Make the mesh :> `ElmerGrid 1 2 square.grd`
- Run the test :> `ElmerSolver ismip_step0.sif`
- Watch the results :> `ElmerPost` and open `square\ismip_step0.ep`
- What are we solving?

Stokes:

$$\text{div } \sigma + \rho g = 0$$

$$u_{i,i} = 0$$



Navier-Stokes with convection and acceleration terms neglected :

Flow Model = String Stokes

in the Stokes solver section

Step 0 – Glen’s law and Elmer

In glaciology, you can find (at least) two definitions for Glen’s law:

$$D_{ij} = \frac{B}{2} \tau_e^{n-1} S_{ij} \quad ; \quad S_{ij} = 2B^{-1/n} \dot{\gamma}^{(1-n)/n} D_{ij}$$

$$D_{ij} = A \tau_e^{n-1} S_{ij} \quad ; \quad S_{ij} = A^{-1/n} I_{D_2}^{(1-n)/n} D_{ij} \quad \text{ISMIP notation}$$

$$\text{where } I_{D_2}^2 = D_{ij} D_{ij} / 2 \quad \text{and} \quad \dot{\gamma}^2 = 2 D_{ij} D_{ij}$$

The power-law implemented in Elmer writes: $S_{ij} = 2\eta_0 \dot{\gamma}^{m-1} D_{ij}$

$$\eta_0 = B^{-1/n} = (2A)^{-1/n}$$

$$m = 1/n$$

$$\dot{\gamma}^2 \geq \dot{\gamma}_c^2$$

In Material Section:

Viscosity Model = String “power law”

Viscosity = Real η_0

Viscosity Exponent = Real m

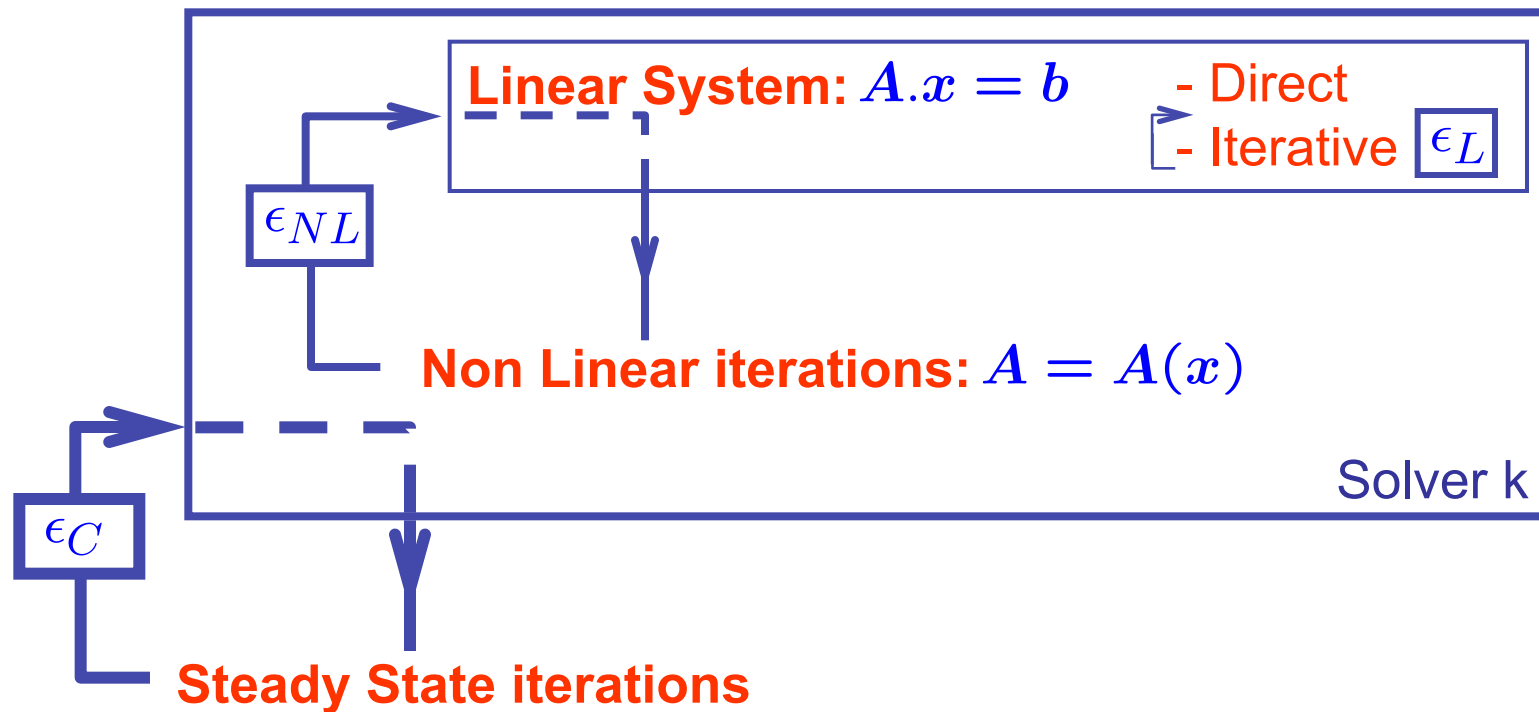
Critical Shear Rate = Real $\dot{\gamma}_c$

Step 0 – Sketch of a Steady simulation

Geometry + Mesh



Degrees of freedom



$$\epsilon_L < \epsilon_{NL} < \epsilon_C$$

Step 0 – Numerical methods

In the NS Solver Section: (see Chapters 3 and 4 of Elmer Solver Manual)

- Solution for the Linear System:

```
Linear System Solver = Direct  
Linear System Direct Method = umfpack
```

- Non-Linear System :

```
Nonlinear System Max Iterations = 100  
Picard Nonlinear System Convergence Tolerance = 1.0e-5 =  $\epsilon_{NL}$   
Newton Nonlinear System Newton After Iterations = 5  
Nonlinear System Newton After Tolerance = 1.0e-02  
Nonlinear System Relaxation Factor = 1.00
```

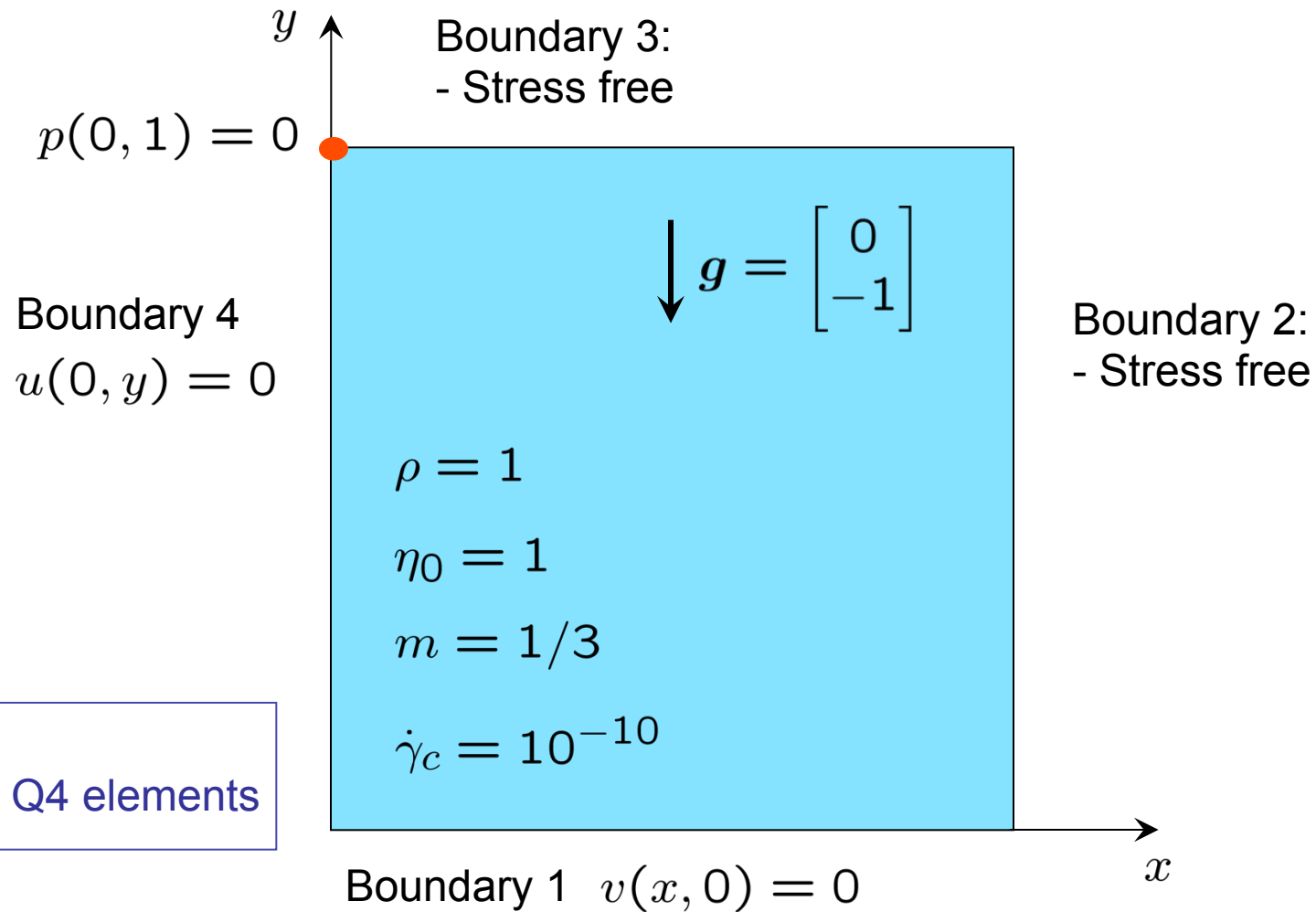
- Coupled problem (not needed here in fact...):

```
Steady State Convergence Tolerance = Real 1.0e-3 =  $\epsilon_C$ 
```

- Stabilization of the Stokes equations:

```
Stabilization Method = String Bubbles (other options: Stabilized, P2P1)
```

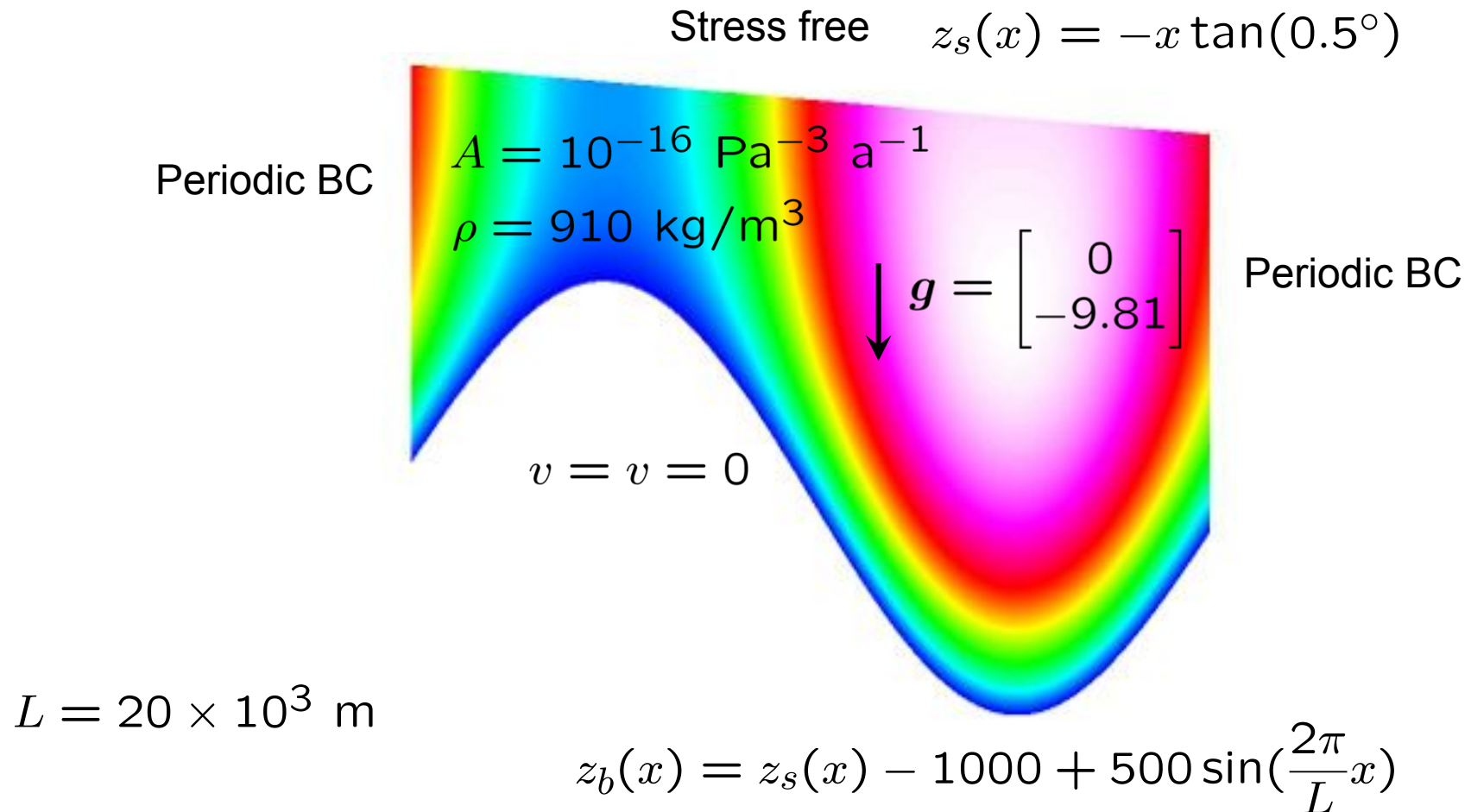
Step 0 – What are we solving?



Mesh:
20 x 20 Q4 elements

Step 1 – Move to ISMIP-HOM B020

What we have to solve :



Step 1 – Changes from Step0

- New directory, new names !

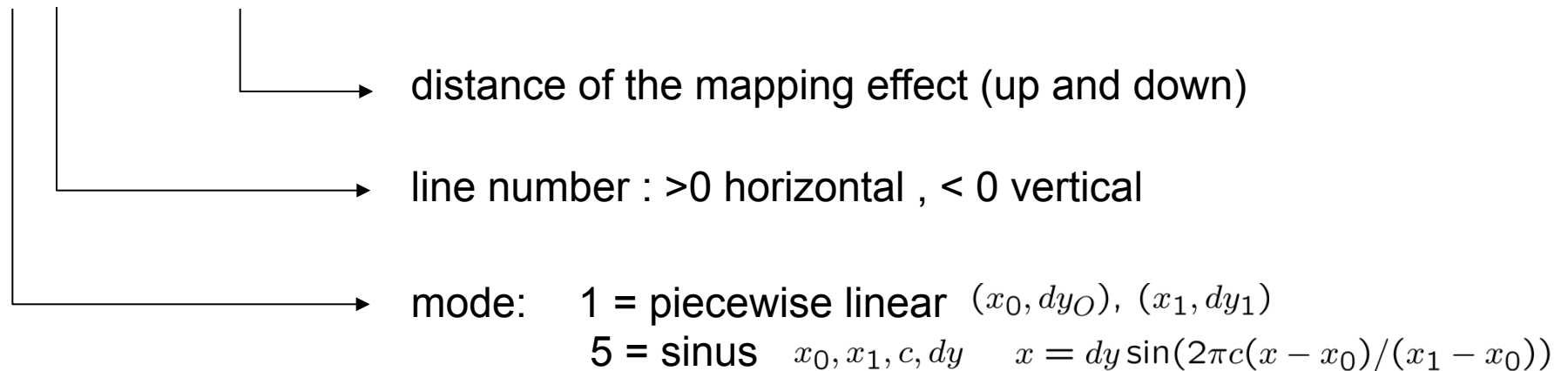
e.g.: `square.grd -> mesh_B.grd`

- Make the mesh : three (at least) possibilities
- Right values for the different constants (which system of Units ?)
- Add the periodic boundary conditions

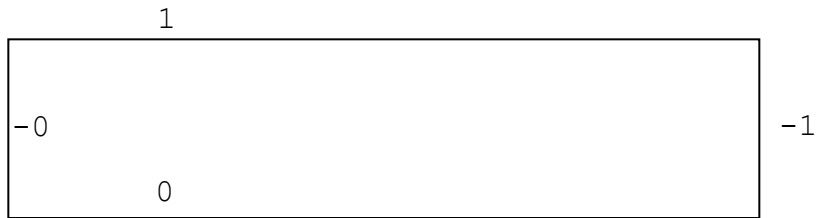
Step 1 – Mesh with ElmerGrid

Use the boundary mappings of ElmerGrid (in `mesh_B.grd`)

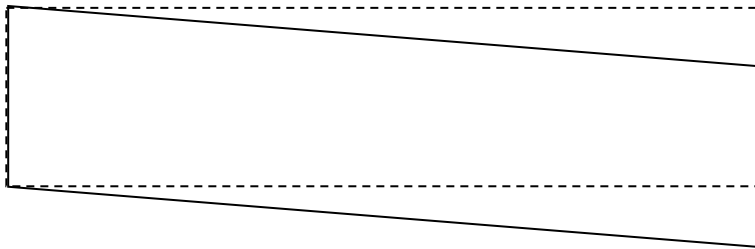
```
...
Subcell Limits 1 = 0.0 2.000000000e+04
Subcell Limits 2 = -1000.0 0.0
...
Geometry Mappings
! mode line limits(2) Np params(Np)
1 1 1000.0 1000.0 4 0.0 0.0 2.000000000e+04 -1.74537356e+02
1 0 1000.0 1000.0 4 0.0 0.0 2.000000000e+04 -1.74537356e+02
5 0 1000.0 1000.0 4 0.0 2.000000000e+04 1.0 500.0
End
```



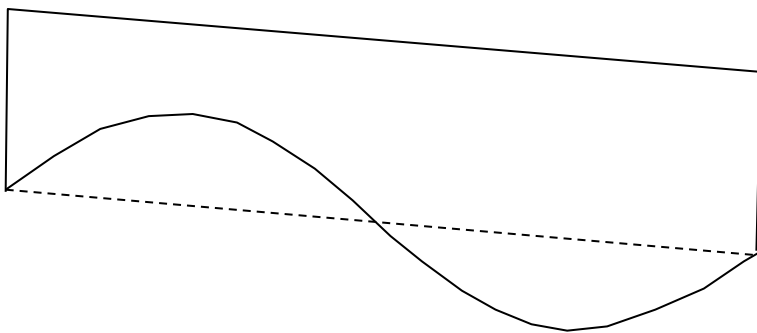
Step 1 – Mapping with ElmerGrid



1 1 1000.0 1000.0 4 0.0 0.0 2.00000000e+04 -1.74537356e+02



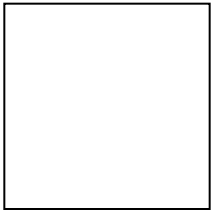
1 0 1000.0 1000.0 4 0.0 0.0 2.00000000e+04 -1.74537356e+02



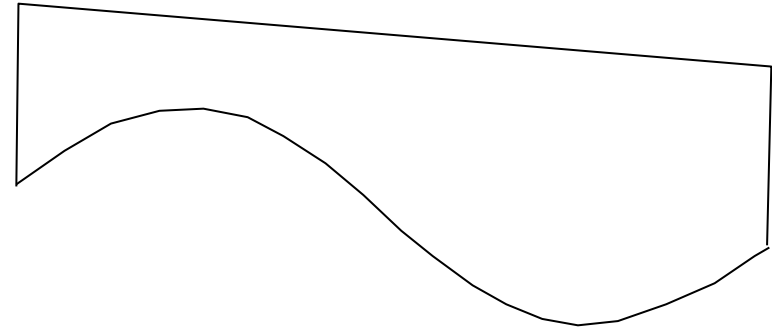
5 0 1000.0 1000.0 4 0.0 2.00000000e+04 1.0 500.0

Step 1 – Mesh with your own script

- Start from the square 1 x 1 of Step 0 and deform it to obtain the mesh of ISMIP B



The script of your choice !
(awk, f90, matlab, ...)



- Open the file `mesh.nodes`

- Read (x, y)

- Make the scaling :
$$\begin{cases} x \rightarrow L \times x \\ y \rightarrow z_b + (z_s - z_b) \times y \end{cases} \quad \begin{cases} z_s(x) = -x \tan(0.5^\circ) \\ z_b(x) = z_s(x) - 1000 + 500 \sin\left(\frac{2\pi}{L}x\right) \\ L = 20 \times 10^3 \text{ m} \end{cases}$$

- Re-write (x, y) in `mesh.nodes`

Step 1 – Mesh with your own script

Format of the file `mesh.nodes` :

Node number , -1, x, y, z (z=0 in 2D)

Examples:

- f90: see file `MSH_ismipB.f90`

```
PROGRAM MSH_ismipB
IMPLICIT NONE
REAL(KIND=8)  ::  x, y, z, xnew, ynew, zb, zs, L, Pi
REAL(KIND=8), ALLOCATABLE  ::  xnode(:), ynode(:)
CHARACTER  ::  NameMsh*20
INTEGER  ::  NtN, i, j, N
    Pi = ACOS(-1.0)
    WRITE(*,*)'Name of the Elmer mesh directory ?'
    READ(*,*)NameMsh
    WRITE(*,*)'Lenght of the mesh L [m] :'
    READ(*,*)L
    OPEN(10,file=TRIM(NameMsh)//"/mesh.header")
        READ(10,1000)NtN
    CLOSE(10)
    ALLOCATE(xnode(NtN), ynode(NtN))
    OPEN(12,file=TRIM(NameMsh)//"/mesh.nodes")
    READ(12,*) (N, j, xnode(i), ynode(i), z, i=1,NtN)
    REWIND(12)
```

```
DO N=1, NtN
    x = xnode(N)
    y = ynode(N)
    xnew = x * L
    zs = -xnew*TAN(0.5*Pi/180.0)
    zb = zs - 1000.0 + 500.0*SIN(2.0*Pi*xnew/L)
    ynew = zb + y * (zs - zb)
    WRITE(12,1200)N,j,xnew,ynew,z
END DO
DEALLOCATE(xnode, ynode)
1000 FORMAT(I6)
1200 FORMAT(i5,2x,i5,3(2x,e22.15))
END PROGRAM MSH_ismipB
```

- See other examples :

awk: see file `script_awk`

matlab: see file `dilat_mesh.m`

Step 1 – Mesh with your own mesher

Possible input format for ElmerGrid:

- 4) `.ansys` : Ansys input format
- 5) `.inp` : Abaqus input format by Ideas
- 6) `.fil` : Abaqus output format
- 7) `.FDNEUT` : Gambit (Fidap) neutral file
- 8) `.unv` : Universal mesh file format
- 9) `.mphtxt` : Comsol Multiphysics mesh format
- 10) `.dat` : Fieldview format
- 11) `.node,.ele`: Triangle 2D mesh format
- 12) `.mesh` : Medit mesh format
- 13) `.msh` : GID mesh format
- 14) `.msh` : Gmsh mesh format

Examples with gmsh: see file `mesh_Bgmsh.geo`

```
> gmsh mesh_Bgmsh.geo -2 -o mesh_Bgmsh.msh
> ElmerGrid 14 2 mesh_Bgmsh.msh -autoclean
  → mesh_Bgmsh/mesh.*


> ElmerGrid 14 3 mesh_Bgmsh.msh -autoclean
  → mesh_Bgmsh.ep (ElmerPost)
```


Step 1 – Elmer and Units


The choice of Units have to be coherent.
But you are free because the Stiff matrix is normalized.

For the Stokes problem, one should give values for:

- the density: ρ ($= 910 \text{ kg/m}^3$)
- the gravity: g ($= 9.81 \text{ m s}^{-2}$)
- the viscosity: η_0 ($\text{Pa s}^{1/n}$) ($1 \text{ Pa} = 1 \text{ kg s}^{-2} \text{ m}^{-1}$)

kg – m – s [USI] : velocity in m/s and timestep in secondes 

kg – m – a : velocity in m/a and timesteps in years  1 a = 31 557 600 s

Pa – m – a : velocity in m/a and Stress in Pa 
(What I will use in the following)

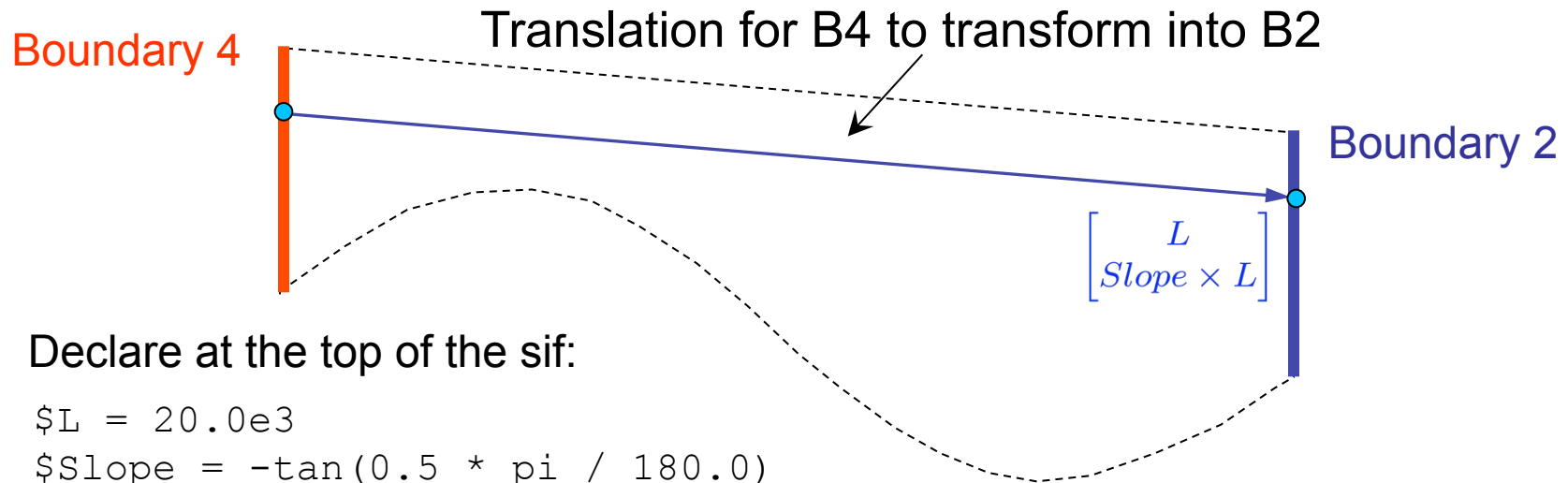
Step 1 – Value of the ISMIP constants

For ISMIP tests A-D, the value for the constants are

- the density: $\rho = 910 \text{ kg/m}^3$
- the gravity: $g = 9.81 \text{ m s}^{-2}$
- the fluidity: $A = 10^{-16} \text{ Pa}^{-3} \text{ a}^{-1}$

	USI kg - m - s		kg - m - a		MPa - m - a	
$g =$	9.81	m / s^2	9.7692E+15	m / a^2	9.7692E+15	m / a^2
$\rho =$	910	kg / m^3	910	kg / m^3	9.1380E-19	$\text{MPa m}^{-2} \text{ a}^2$
$A =$	3.1689E-24	$\text{kg}^{-3} \text{ m}^3 \text{ s}^5$	1.0126E-61	$\text{kg}^{-3} \text{ m}^3 \text{ a}^5$	100	$\text{MPa}^{-3} \text{ a}^{-1}$
$\eta =$	5.4037E+07	$\text{kg m}^{-1} \text{ s}^{-5/3}$	1.7029E+20	$\text{kg m}^{-1} \text{ a}^{-5/3}$	0.1710	$\text{MPa a}^{1/3}$

Step 1 – Periodic boundary conditions



Declare at the top of the sif:

```
$L = 20.0e3
```

```
$Slope = -tan(0.5 * pi / 180.0)
```

...

```
Boundary Condition 2
```

```
Target Boundaries = 2
```

```
Periodic BC = 4
```

```
Periodic BC Translate(2) = Real $ (L) (Slope*L)
```

```
Periodic BC Velocity 1 = Logical True
```

```
Periodic BC Velocity 2 = Logical True
```

```
Periodic BC Pressure = Logical True
```

```
End
```

```
Boundary Condition 4
```

```
Target Boundaries = 4
```

Nothing to declare for BC4 !

```
End
```

Step 2 – Add SaveData Solver (SaveLine)

Objective: save the variables on the top surface (ASCII matrix file)

- Add a new solver

```
Solver 2
  Exec Solver = After All
  Procedure = File "SaveData" "SaveLine"
  Filename = "ismip_surface.dat"
  File Append = Logical False
End
```

- Add this solver in the Equation Section

```
Active Solvers(2) = 1 2
```

- Tell in which BC you want to save the data

```
Boundary Condition 3
  Target Boundaries = 3
  Save Line = Logical True
End
```

- Ordering of the variables: see file `ismip_surface.dat.names`

Step 2 – Add SaveData Solver (SaveLine)

SaveLine can also be used to save data at a ‘drilling site’ (a line which is not a boundary). Here, the data are saved at $x = 10\text{km}$.

- Add a new solver

Solver 2

```
Exec Solver = After All
Procedure = File "SaveData" "SaveLine"
Filename = "ismip_drilling.dat"
Polyline Coordinates(2,2) = Real $ (0.5*L) -1000. (0.5*L) 0.0
File Append = Logical False
```

End

- Add this solver in the Equation Section

```
Active Solvers(2) = 1 2
```

- And don't forget to comment the `Save line = Logical True` in BC3

Step 3 – Add SaveScalars

SaveScalars allows to save scalars and derived quantities. Here, we will save:

- 1/ the volume of the domain (surface),
- 2/ the maximum value of the absolute horizontal velocity,
- 3/ the flux on the 3 boundaries 2, 3 and 4.
- 4/ the CPU time,
- 5/ the CPU memory, (? not working)

Step 3 – Add SaveScalars

-Add a new solver

```
Solver 3
  Exec Solver = After TimeStep    !! For transient simulation
  Procedure = "SaveData" "SaveScalars"
  Filename = "ismip_scalars.dat"
  File Append = Logical True      !! For transient simulation
  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"
  Operator 4 = String "cpu time"
  Operator 5 = String "cpu memory"
End
```

- Add this solver in the Equation Section

```
Active Solvers(3) = 1 2 3
```

- Tell at which boundaries you want to save the flux

```
Flux Integrate = Logical True
```

Step 4 – Add ComputeDevStress

Objective: compute the stress field as

$$\int_V S_{ij} \Phi \, dV = 2 \int_V \eta D_{ij} \Phi \, dV$$

where D_{ij} and η are calculated from the nodal velocities using the derivative of the basis functions

- Add a Solver

```
Solver 2
  Equation = Sij
  Variable = -nooutput "Sij"
  Variable DOFs = 1
  Exported Variable 1 = Stress[Sxx:1 Syy:1 Szz:1 Sxy:1]
  Exported Variable 1 DOFs = 4
  Procedure = "./ComputeDevStress" "ComputeDevStress"

  Flow Solver Name = String "Flow Solution"

  Linear System Solver = Direct
  Linear System Direct Method = umfpack
End
```

Step 4 – Add ComputeDevStress

- Add this solver in the Equation Section

```
Active Solvers(4) = 1 2 3 4
```

- Add the 4 stress components in the periodic BC

```
Boundary Condition 2
```

```
...
```

```
Periodic BC Sxx = Logical True
```

```
Periodic BC Syy = Logical True
```

```
Periodic BC Szz = Logical True
```

```
Periodic BC Sxy = Logical True
```

```
End
```

- Add in the material section:

```
Cauchy = Logical False
```

- compile this new Solver:

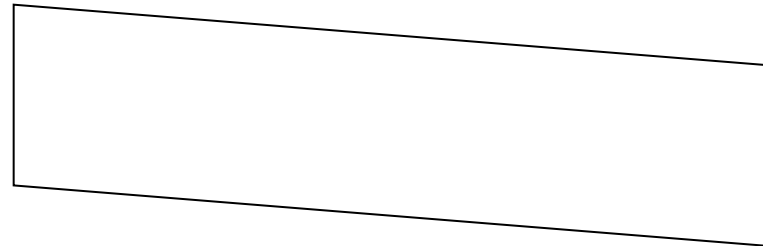
```
> elmerf90 ComputeDevStressNS.f90 -o ComputeDevStress
```

Step 5 – Move to ISMIP-HOM D020

Changes from B020:

- geometry of domain

$$\begin{cases} z_s(x, y) = -x \tan(0.1^\circ) \\ z_b(x, y) = z_s(x, y) - 1000 \end{cases}$$



➔ modify the mesh and the Boundary Condition 2
(Easy !)

- boundary condition at the bedrock interface

$$\begin{cases} \tau_{nt} = \beta^2 u_t \\ u_n = \mathbf{u} \cdot \mathbf{n} = 0 \end{cases} \quad \text{with } \beta^2(x) = 1000 + 1000 \sin\left(\frac{2\pi}{L}x\right)$$

in [Pa a m⁻¹] !

➔ modify the Boundary Condition 1

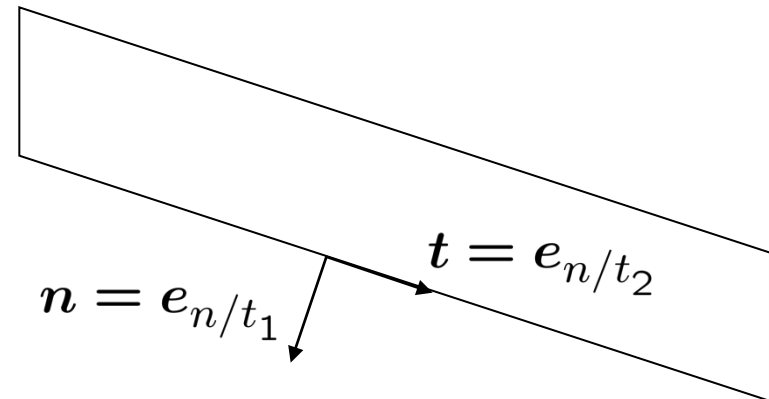
Step 5 – Move to ISMIP-HOM D020

Friction law in Elmer:

$$C_i u_i = \sigma_{ij} n_j \quad (i = 1, 2)$$

→ $C_t u_t = \sigma_{nt} ; C_n u_n = \sigma_{nn}$

where n is the surface normal vector



Modification of the Boundary Condition 1:

- First Solution: MATC definition of C_t

Boundary Condition 1

Target Boundaries = 1

Flow Force BC = Logical True

Normal-Tangential Velocity = Logical True

Velocity 1 = Real 0.0e0

Slip Coefficient 2 = Variable coordinate 1

Real MATC "1.0e-3*(1.0 + sin(2.0*pi* tx / L))

End

} Stress condition defined in a normal-tangential coordinate system

} $u_n = 0$

} $C_t = \dots$

in [MPa a m⁻¹] !

Step 5 – Move to ISMIP-HOM D020

- Second Solution: User Function to define Ct

```
Boundary Condition 1
```

```
...
```

```
Slip Coefficient 2 = Variable coordinate 1
```

```
Real Procedure “./ISMIP_D” “Sliding”
```

```
End
```

where Sliding is a User Function defined in the file `ISMIP_D.f90`
(see next slide)

Compilation:

```
> elmerf90 ISMIP_D.f90 -o ISMIP_D
```

Step 5 – Move to ISMIP-HOM D020

```
FUNCTION Sliding ( Model, nodenumber, x) RESULT(C)
  USE Types

  IMPLICIT NONE
  TYPE(Model_t) :: Model
  INTEGER :: nodenumber, i
  REAL(KIND=dp) :: x, C, L
  LOGICAL :: FirstTime=.True.

  SAVE FirstTime, L

  IF (FirstTime) THEN
    FirstTime=.False.
    L = MAXVAL(Model % Nodes % x)
  END IF

  x = Model % Nodes % x(nodenumber)
  C = 1000.0e-6_dp*(1.0_dp + SIN(2.0_dp * Pi * x/ L))
      ! in MPa a /m

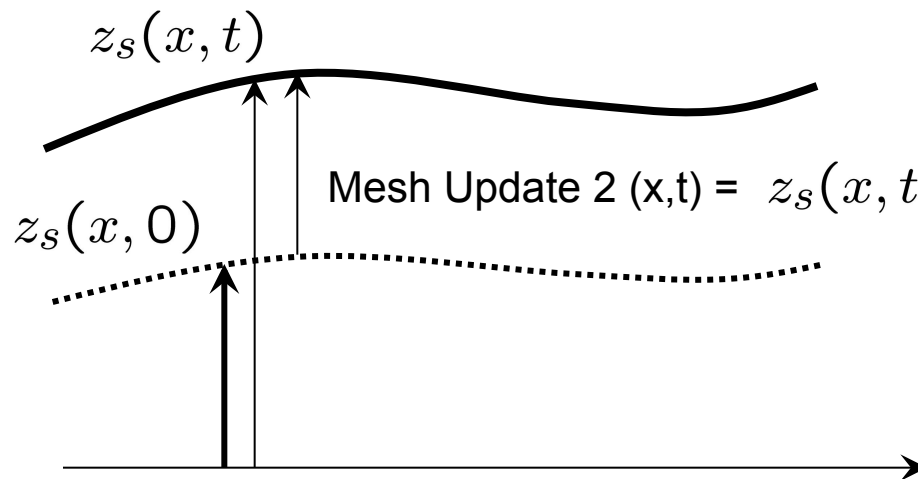
END FUNCTION Sliding
```

Step 6 – Move to prognostic B020

Move from a Diagnostic to a prognostic simulations:

- Steady to transient
- Add two solvers: the **free surface** and the **Mesh Update** solvers

$$\frac{\partial z_s}{\partial t} + u_x \frac{\partial z_s}{\partial x} - u_z = a$$



Mesh Update solver deforms the mesh to follow the moving boundary (Elastic deformation)

Step 6 – Steady to transient

The simulation Section has to be modified:

Simulation Type = Transient

Timestepping Method = "bdf" → Backward Differences Formulae

BDF Order = 1

Output Intervals = 1 → Save in .ep file

Timestep Intervals = 200

Timestep Sizes = 1.0

Steady State Min Iterations = 1

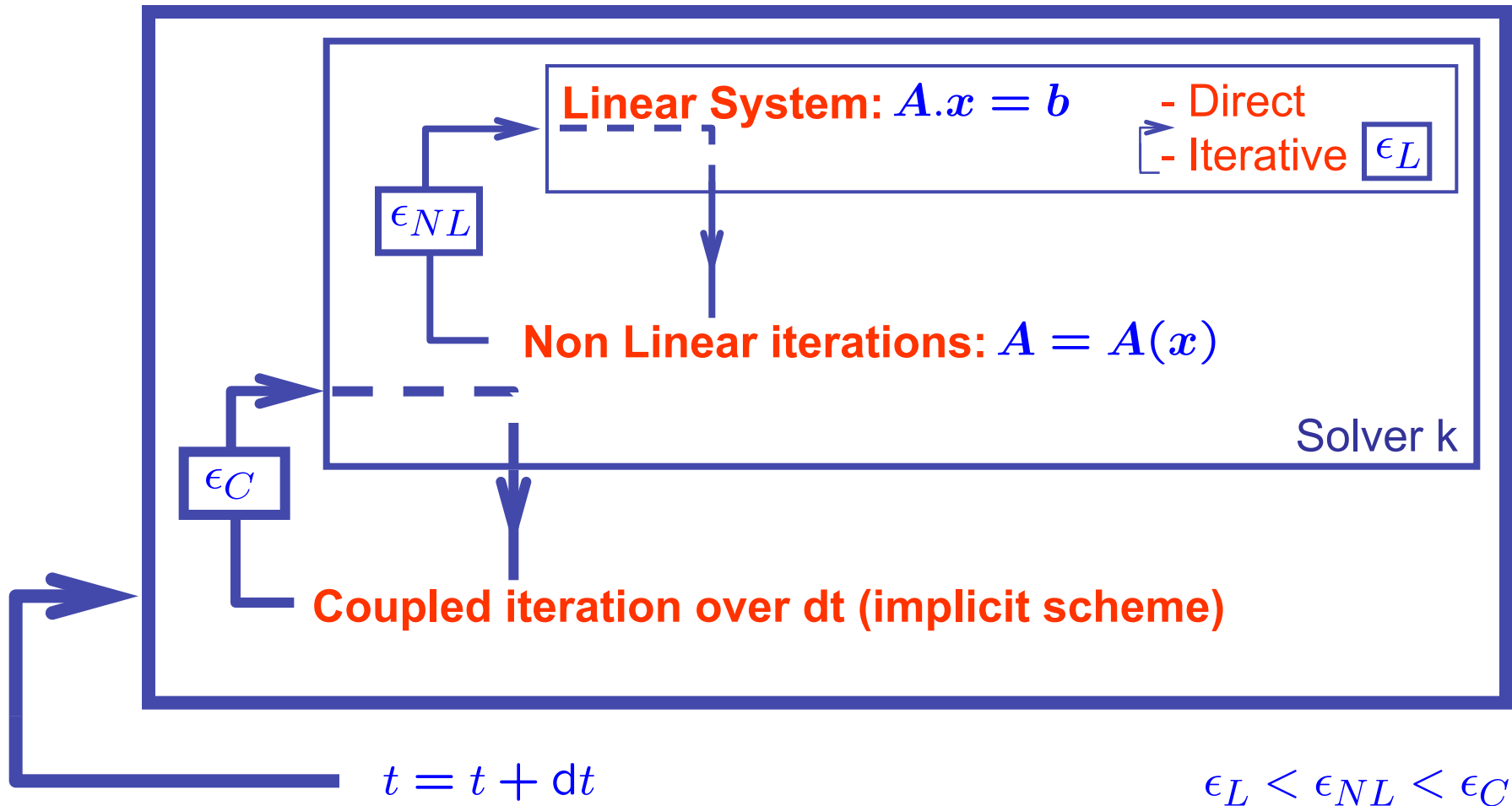
Steady State Max Iterations = 10 → To control the "implicitness" of the solution over one time step (see example below).

Step 6 – Sketch of a transient simulation

Geometry + Mesh



Degrees of freedom



Step 6 – Free surface Solver

The free surface solver only apply to the boundary 3 (top surface)

➔ Define a 2nd body which is the boundary 3.

```
Body 2
  Equation = 2
  Body Force = 2
  Material = 1
  Initial Condition = 2
End
```

where Equation 2, Body Force 2 and Initial Condition 2 are defined for the free surface equation.

Tell in BC3 that this is the body 2:

```
Boundary Condition 3
  Target Boundaries = 3
  ...
  !!! this BC is equal to body no. 2 !!!
  Body Id = 2
  ...
End
```

Step 6 – Free surface Solver

Add the Free Surface Solver:

The minimum is presented here,
you can add limits not to be
penetrated by the free surface

```
Solver 2
Equation = "Free Surface"
Variable = String Zs
Variable DOFs = 1
Exported Variable 1 = String "Zs Residual"
Exported Variable 1 DOFs = 1

Procedure = "FreeSurfaceSolver" "FreeSurfaceSolver"
Before Linsolve = "EliminateDirichlet" "EliminateDirichlet"

Linear System Solver = Iterative
Linear System Max Iterations = 1500
Linear System Iterative Method = BiCGStab
Linear System Preconditioning = ILU0
Linear System Convergence Tolerance = Real 1.0e-5
Linear System Abort Not Converged = False
Linear System Residual Output = 1

Steady State Convergence Tolerance = 1.0e-03

Relaxation factor = Real 1.0
Stabilization Method = Bubbles
End
```

...

Step 6 – Free surface Solver

Body Force 2:

```
Body Force 2
  Zs Accumulation Flux 1 = Real 0.0e0
  Zs Accumulation Flux 2 = Real 0.0e0
End
```

Equation 2:

```
Equation 2
  Active Solvers(1) = 2
  Flow Solution Name = String "Flow Solution"
  Convection = String Computed
End
```

Initial Condition 2: (tell that $z_s(x, 0) =$ ordinate of the initial top surface)

```
Initial Condition 2
  Zs = Variable Coordinate 1
  Real MATC "tx*Slope"
End
```

OR

```
Initial Condition 2
  Zs = Variable Coordinate 1
  real Procedure "USF_Zs" "ZsIni"
End
```

Step 6 – Mesh Update Solver

Add the Mesh Update Solver:

```
Solver 3
  Equation = "Mesh Update"
  Linear System Solver = "Direct"
  Linear System Direct Method = umfpack
  Steady State Convergence Tolerance = 1.0e-04
End
```

Material parameter for this solver:

```
Mesh Youngs Modulus = Real 1.0
Mesh Poisson Ratio = real 0.3
```

Force that Mesh Update 1 = 0 everywhere:

```
Body Force 1
  Flow BodyForce 1 = Real 0.0
  Flow BodyForce 2 = Real -9.7696e15 !MPa - a - m
  Mesh Update 1 = real 0.0
End
```

Step 6 – Mesh Update Solver

Boundary condition:

BC1: Mesh Update 1 = real 0.0
Mesh Update 2 = real 0.0

BC2: Periodic BC Mesh Update 2 = Logical True
Mesh Update 1 = real 0.0

BC3: Mesh Update 1 = real 0.0

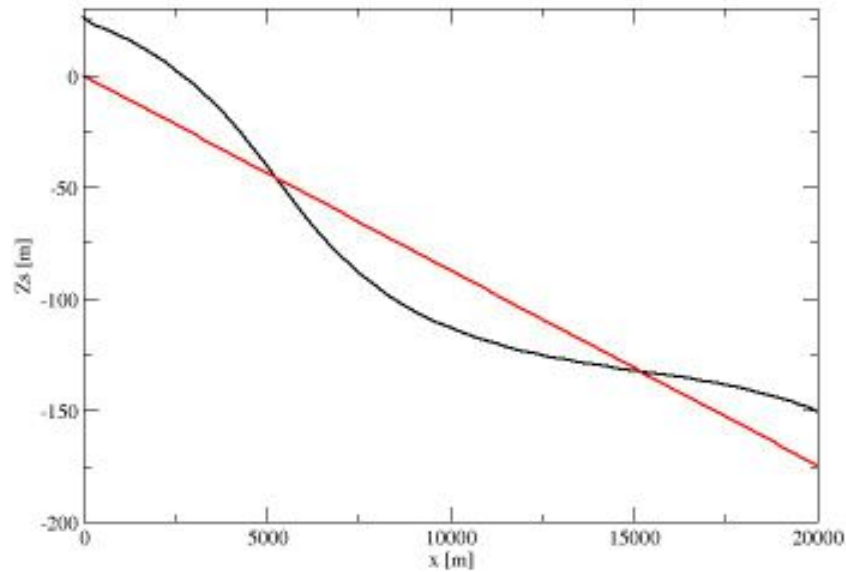
OR

Mesh Update 2 = Variable Zs, Coordinate 1
Real MATC "tx(0)-Slope*tx(1)"

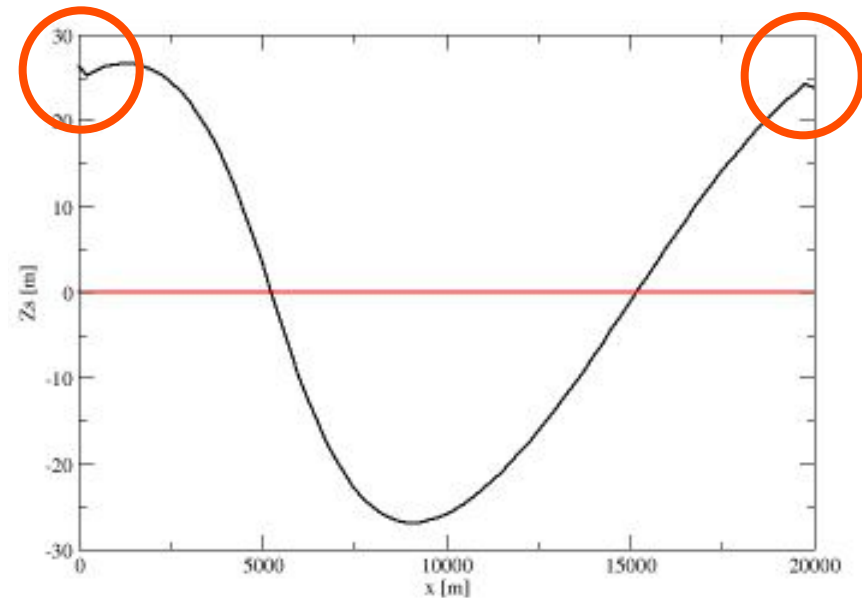
Mesh Update 2 = Variable Zs
Real Procedure "./USF_Zs" "ZsMZsIni"

Step 6 – Results !

Comparison of the initial and steady surface of the prognostic run



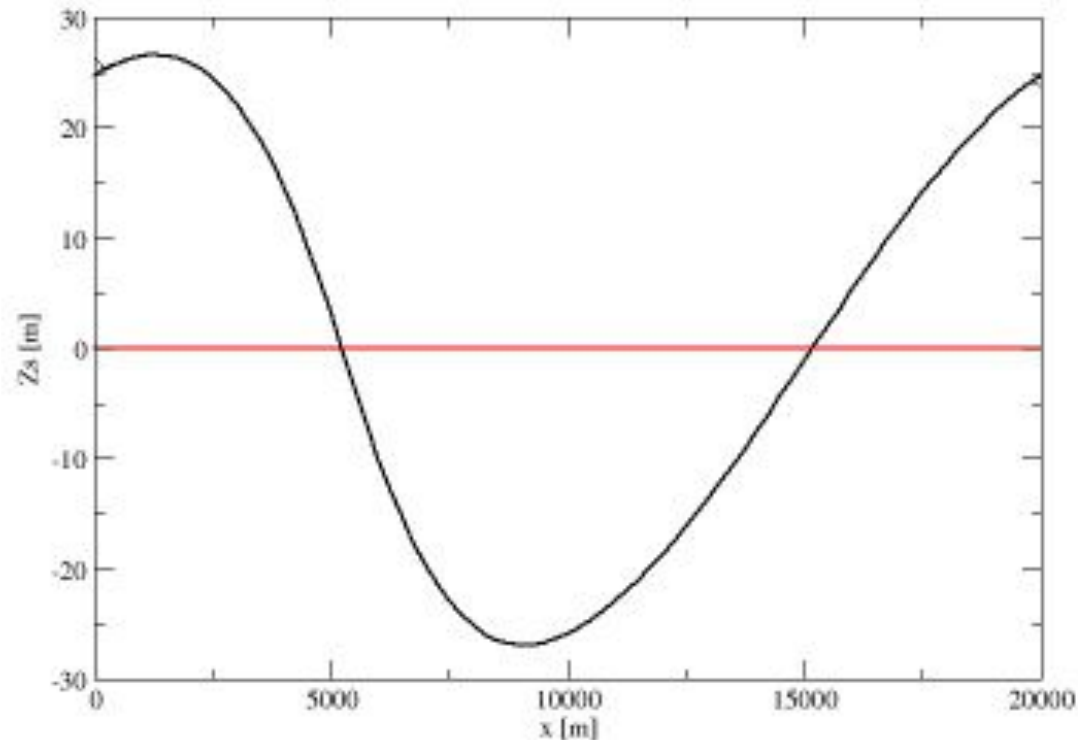
Not
periodic



Step 6 – better results !

Turn the mesh so that $z_s = 0$ (turn the gravity vector also !)
Force z_s to be periodic

See Step6_hori



Step 7 – Move to prognostic D020

Merge Step 5 and Step 6 and it should work !

