For transient simulations we have implemented the algorithm proposed by Alauzet et al. (2007).

1 IMPLEMENTATION

This algorithm is implemented in a *bash* script that can be found under *[ELMER_TRUNK]/elmerice/Solvers/MeshAdaptation_2D/Script_Transient.sh*.

The algorithm description is given below:

Algorithm 1 Script_Transient.sh

N := Integer ▷ Number of Timestep Intervals for Physical Simulation
dt := Real ▷ Timestep Sizes for Physical Simulation
$OuputIntervals$:= Integer ▷ Output Intervals for Physical Simulation
Execute RUN_INIT.sif ▷ Initialisation
for $i = 0, i_{max}$ **do** ▷ t=$[0, (i_{max} + 1) * N * dt]$
    for $j = 0, j_{max}$ **do** ▷ t=$[i * N * dt, (i + 1) * N * dt]$
        Execute RUN_I_J.sif ▷ Physical simulation
        Convergence ?
        if $((< CONVERGED >)$ .OR. $(j = j_{max}))$ **then**
            $k_{max} = 1$
            $< ReadTransientResult >:= False$
            $i+ = i + 1$
            $j+ = 0$
        else
            $k_{max} =< N > / < OuputIntervals > +1$
            $< ReadTransientResult >:= True$
            $i+ = i$
            $j+ = j + 1$
        end if
        Execute MESH_OPTIM_I_J.sif ▷ Mesh adaptation
    end for
end for

The general steps are as follow:

- A physical simulation for $t = [0, t_f]$ is divided in $i_{max} + 1$ subsets;
- The configuration file *RUN_IINIT.sif* initialises the transient ice flow problem; *e.g.* initialises the ice-sheet geometry from observations.
- The configuration file *RUN_I_J.sif* solves a transient ice flow problem (*e.g.* solves a balance equation to compute the velocity field and a mass conservation equation for the geometry evolution) for the subset $t = [i * N * dt, (i + 1) * N * dt]$, saving the results every $< N >$ time-steps.
- The configuration file *MESH_OPTIM_I_J.sif* performs the mesh adaptation. The metric is constructed using the informations saved by the

physical simulation, *i.e.* the mesh is adapted using the simulation history, allowing to refine the mesh where needed during the transient simulation. When moving to the next subset, the mesh is adapted using only the informations from the mast time-step.

- If j_{max} is set to 0, this is equivalent to adapt the mesh every N time-steps using only the information from this time-step. Each subset $t = [i*N*dt, (i+1)*N*dt]$ is solved only once, but the mesh has been adapted using only the informations at $t = i*N*dt$. If the simulation involves large changes, the mesh must be adapted often to keep high resolution where needed; this can result in a loss of accuracy due to the interpolation between meshes.

- If $j_{max} > 0$, the subset $t = [i*N*dt, (i+1)*N*dt]$ is solved iteratively several times. At each iteration the mesh has been adapted using the informations saved every $< N >$ intervals for the subsets $t = [i*N*dt, (i+1)*N*dt]$. Requires more computing time as the subset is solved several times and meshes are usually larger as they are refined where needed to keep high accuracy during the transient simulation. However, sensitivity of the results to the mesh is part of the outputs.

The algorithms for the configuration files are given below:

- **RUN_INIT.sif**: Initialisation file. Typically, Initialise the ice sheet geometry and create the first restart file *M_I0_I0.result*.

Algorithm 2 RUN_INIT.sif

Mesh := MESH_I0_J0
$t := 0$
$h0$:= Get Initial Ice Sheet Geometry
Save M_I0_I0.result

- **RUN_I_J.sif**: Physical simulation for the subset $t = [i*N*dt, (i+1)*N*dt]$; Typically compute the velocity and geometry evolution.

- **MESH_OPTIM_I_J.sif**: Mesh adaptation. Compute the metrics and metric intersection.

Algorithm 3 RUN_I_J.sif

Mesh := MESH_I<i>_J<j>
Restart last step in M_I<i>_I<j>.result
$t := < i > \times < N > \times < dt >$
$h := h0$
for k = 1,<N> **do**
    $t := t+ < dt >$
    Compute h(t), *etc*...
    if $(k/ < OuputIntervals > -1 = 0)$ **then**
        Save in R_I<i>_J<j>.result
    end if
end for
Save in R_I<i>_J<j>.result

Algorithm 4 MESH_OPTIM_I_J.sif

Mesh := MESH_I<i>_J<j>
Restart last step in R_I<i>_I<j>.result
$h0 := h$ ▷ $h0 = h((i + 1) * N * dt)$
$M_t := M_0$
for $k = 1, k_{max}$ **do**
    if $(< ReadTransientResult >)$ **then**
        Read step k in R_I<i>_I<j>.result ▷ $overwrite h0 => h(i * N * dt)$
    end if
    Compute Metric $M = f(h, ...)$
    Compute Metric $M_t = M_t \cap M$
end for
Create Mesh := MESH_I$< i+ >$_J$< j+ >$
Save in R_I$< i+ >$_J$< j+ >$.result

2 EXAMPLE

An example can be found under *[ELMER_TRUNK]/elmerice/elmerice/Tests/MMG2D_Transient*. This test case is a classical problem of solid bodies in rotation used to test the performance of numerical methods to solve convection dominated transport equations.

Here we use the ThicknessSolver to solve

$$\frac{\partial H}{\partial t} + \nabla.(H\boldsymbol{u}) = 0 \tag{1}$$

where the divergence free velocity field is given by

$$\boldsymbol{u} = 2\pi(y - 0.5, 0.5 - x) \tag{2}$$

The initial bodies (a cone and a gaussian bump) experience a clockwise rotation. They come back at their initial position at $t = 1$.
In this example,

- **RUN_INIT.sif** initialise the two bodies.
- **RUN_I_J.sif** solve Eq. (1) where the velocity field Eq. (2) is prescribed.
- **MESH_OPTIM_I_J.sif** adapt the mesh using solutions for H.

Some results are shown in Fig.1. Parameters used for this example are:

1. SIMULATIONS PARAMETERS
   - dt=0.001 # time step size
   - time_intervals=500 # number of time steps
   - output_interval=10 # output intervals
2. ALGORITHM PARAMETERS
   - $i_{max} = 0$
   - $j_{max} = 10$

With $j = 0$ the bodies leave the initially refined areas and the shape is not preserved; After 3 iterations of the mesh adaptation loop ($j = 3$), the mesh has been adapted taking into account the displacement of the bodies computed during the previous iteration ($j = 2$). The shape of the bodies is better preserved.

N.B: The thickness solver uses by default a SUPG stabilisation scheme for the convection term; there is still some numerical diffusion and the initial shape is not totally preserved; This is a known issue with transport equations.

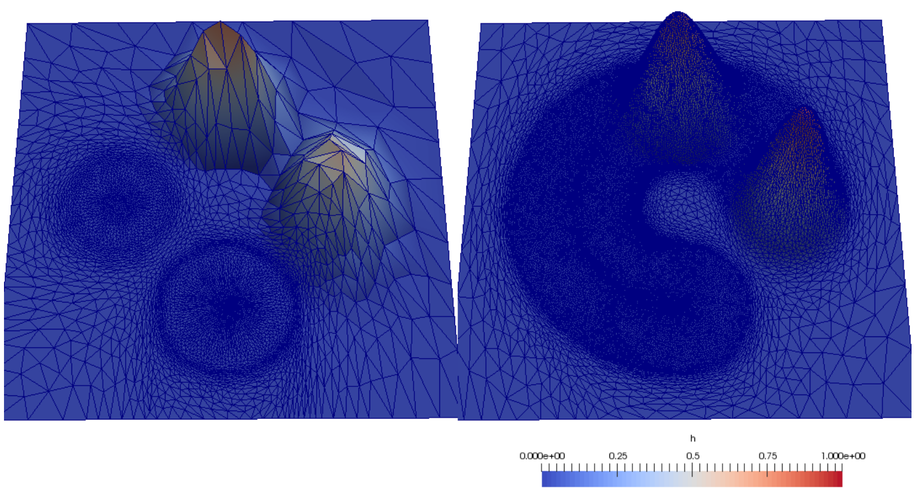

Figure 1: The Bump and the Cone at $t = 0.5$ (*i.e.* rotation of π) for (left) $j = 0$, (right) $j = 3$.

References

Alauzet, F., Frey, P.J., George, P.L., Mohammadi, B., 2007.
3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations.
Journal of Computational Physics, 222, https://doi.org/10.1016/j.jcp.2006.08.012