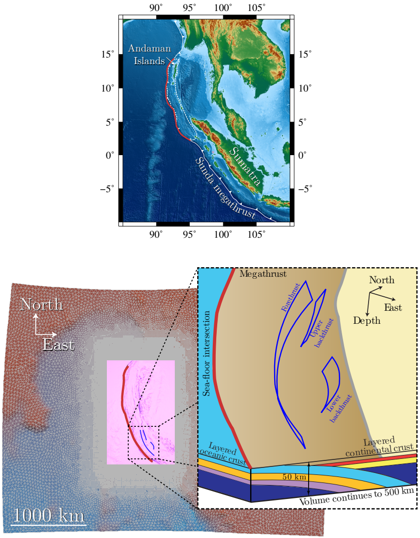
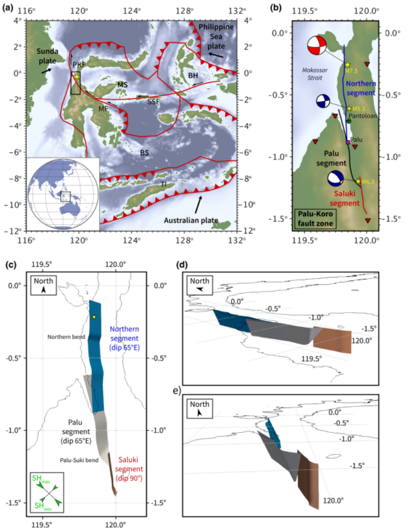
The meshing challenge

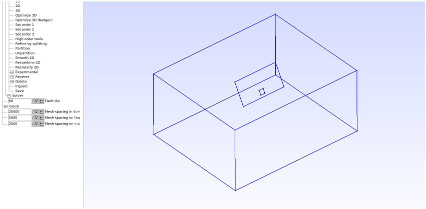


Uphoff et al., SC ’17, Article 21, 2017.



Ulrich et al., Pure Appl. Geophys. 176, 4096–4109, 2019.

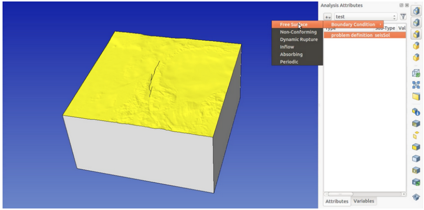
Meshing workflow



Gmsh

- Open source
- CAD modeling
- **Serial** mesh generation
- www.gmsh.info

☞ Hands-on later



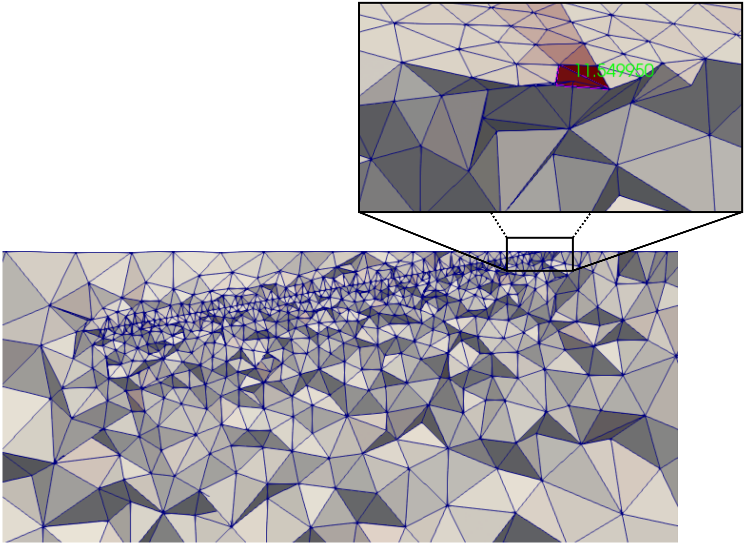
Simulation Modeling Suite / SimModeler

- Commercial (free for academic use)
- Discrete toolbox (e.g. mesh intersection)
- **Parallel** mesh generation
- www.simmetrix.com

Mesh generation problems

Automatic mesh generation often generates "slivers".

Right-hand side:
Cross-section of mesh with shallow dipping fault (10°)
Sliver at free-surface intersection with circumsphere to insphere ratio > 11.



Impact of slivers on computational cost

CFL condition of explicit time-stepping algorithms:

$$\Delta t \leq C_{\text{CFL}} \min_{K_i \in \mathcal{T}_h} \left(\frac{h_i}{c_i} \right)$$

$\mathcal{T}_h$	Mesh
h_i	Element insphere radius
c_i	Maximum wave-speed in element

Global time-stepping (GTS): Run-time $\propto \frac{1}{\Delta t_{\text{min}}}$

$\Rightarrow$ One "bad" sliver makes scenario super expensive or even computational infeasible

Remedy: Local time-stepping (LTS)

Main idea: Let each element update with its own time-step.[1]

Assume we have a mesh with E elements, where a fraction α has time-step Δt_1 and the other elements have time-step $\Delta t_2 = \tau \Delta t_1, \tau > 1$.
Number of space-time updates with end-time T:

$$N_{\text{GTS}} = \alpha E \frac{T}{\Delta t_1} + (1-\alpha) E \frac{T}{\Delta t_1}$$
$$N_{\text{LTS}} = \alpha E \frac{T}{\Delta t_1} + (1-\alpha) E \frac{T}{\Delta t_2}$$

Speed-up:

$$\frac{N_{\text{GTS}}}{N_{\text{LTS}}} = \frac{\tau}{1-\alpha+\tau\alpha}$$

If $\alpha \to 0$ (only a handful of slivers), then speed-up is τ.

[1] Dumbser et al., Geophys. J. Int. 171, 695–717, 2007.

LTS in computational seismology

1. LTS-Newmark[2]
   - Second-order accurate
   - Multi-level scheme, i.e. $\Delta t_l = r_l \Delta t_{l-1}$ with $r_l \in \mathbb{N}$.
   - Implemented in SpecFEM3D Cartesian
2. ADER
   - Arbitrary high-order accurate
   - In theory, every element may have its own time-step[3]
   - In practice, complicated control-flow $\Rightarrow$ group elements in time clusters[4]
   - Implemented in SeisSol, EDGE

[2] Rietmann et al., JCP 334, 308–326, 2017.
[3] Dumbser et al., Geophys. J. Int. 171, 695–717, 2007.
[4] Breuer et al., IPDPS '16, 854–863, 2016.

ADER-LTS: Cauchy-Kowalevski procedure

Elastic wave equation written as general system of linear PDEs (Einstein convention):

$$\frac{\partial q_p}{\partial t} + A_{pq1}\frac{\partial q_q}{\partial x_1} + A_{pq2}\frac{\partial q_q}{\partial x_2} + A_{pq3}\frac{\partial q_q}{\partial x_3} = E_{pq}\, q_q$$

Use PDE to express time derivatives as spatial derivatives:

$$\frac{\partial q_p}{\partial t} = \left(-A_{pqd}\frac{\partial}{\partial x_d}\right) q_q + E_{pq} q_q$$

Cauchy-Kowalevski procedure:

$$\frac{\partial^i q_p}{\partial t^i} = \left(-A_{pqd}\frac{\partial}{\partial x_d}\right)\frac{\partial^{i-1} q_q}{\partial t^{i-1}} + E_{pq}\frac{\partial^{i-1} q_q}{\partial t^{i-1}}$$

ADER-LTS: Discrete Cauchy-Kowalevski procedure

Plug basis expansion $q_p(\boldsymbol{x}, t) = Q_{lp}(t)\phi_l(\boldsymbol{x})$ in Cauchy-Kowalevski procedure and recover coefficients via L^2-projection on element K:

$$\frac{\partial^i Q_{lp}}{\partial t^i} \int_K \phi_k \phi_l \,\mathrm{d}\boldsymbol{x} = -\frac{\partial^{i-1} Q_{lq}}{\partial t^{i-1}} A_{pqd} \int_K \phi_k \frac{\partial \phi}{\partial x_d} \,\mathrm{d}\boldsymbol{x} + E_{pq} \frac{\partial^{i-1} Q_{lq}}{\partial t^{i-1}} \int_K \phi_k \phi_l \,\mathrm{d}\boldsymbol{x}$$

Can define a time predictor using the truncated Taylor expansion:

$$Q_{lp}(t) = \sum_{i=0}^{N} \frac{(t - t_0)^i}{i!} \frac{\partial^i Q_{lp}}{\partial t^i}$$

$\Rightarrow$ Integration over arbitrary time intervals trivial

ADER-LTS: Alternative predictors

Define $\chi_i(t) = \frac{(t-t_0)^i}{i!}$ and $D_{lpi} = \frac{\partial^i Q_{lp}}{\partial t^i}$, then we can write the time predictor as

$$q_p(\boldsymbol{x}, t) = D_{lpi}\chi_i(t)\phi_l(\boldsymbol{x})$$

$\Rightarrow$ The discrete Cauchy-Kowalevski yields a space-time polynomial basis expansion (with a monomial basis in time).

Other predictors have been developed, which are more suitable for non-linear or stiff equations.[5] The so-called continuous Galerkin predictor and discontinuous Galerkin predictor also return a space-time polynomial, i.e. the Cauchy-Kowalevski procedure can be replaced by another predictor.

$\Rightarrow$ Sebastian Wolf, MS4, "Advanced Material Models for Seismic Simulations using ADER-DG".

[5] Gassner et al., JCP 230, 4243–4247, 2011.

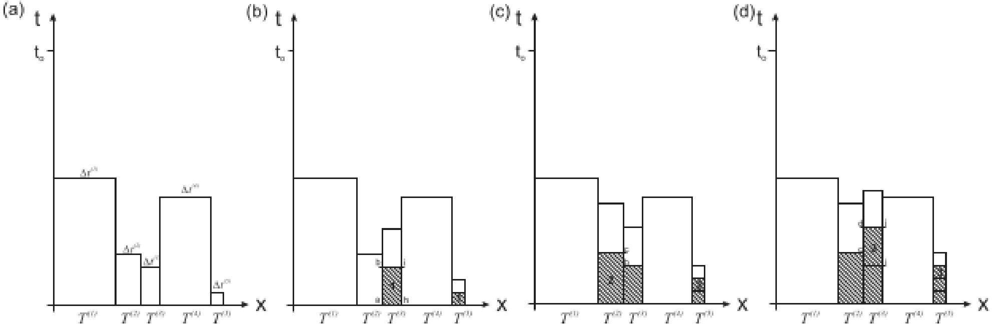
ADER-LTS in parallel

Update criterion:

$$t^m + \Delta t^m \leq \min_j \left(t^{m_j} + \Delta t^{m_j} \right)$$

That is, an element may only complete a time-step if the new time is smaller than the predicted time of its neighboring elements.

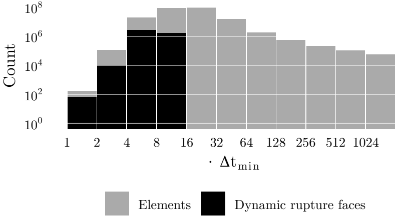
1D example:



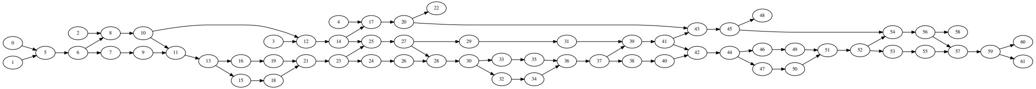
Dumbser et al., Geophys. J. Int. 171, 695–717, 2007.

Clustered LTS

For an efficient implementation, the control logic is simplified by clustering elements,[6] e.g.:

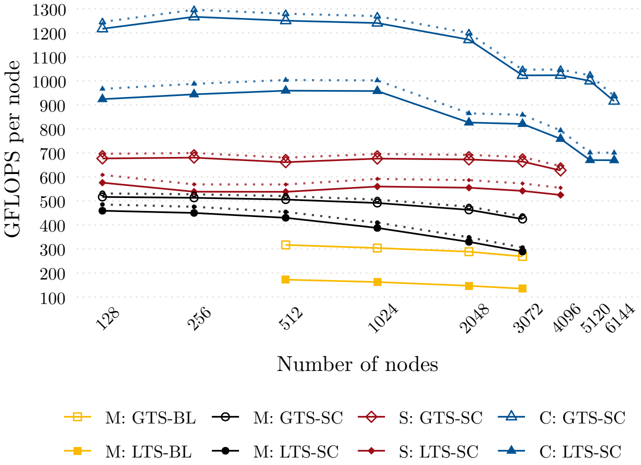


Gives a regular task graph with repeating structure (here 5 levels):



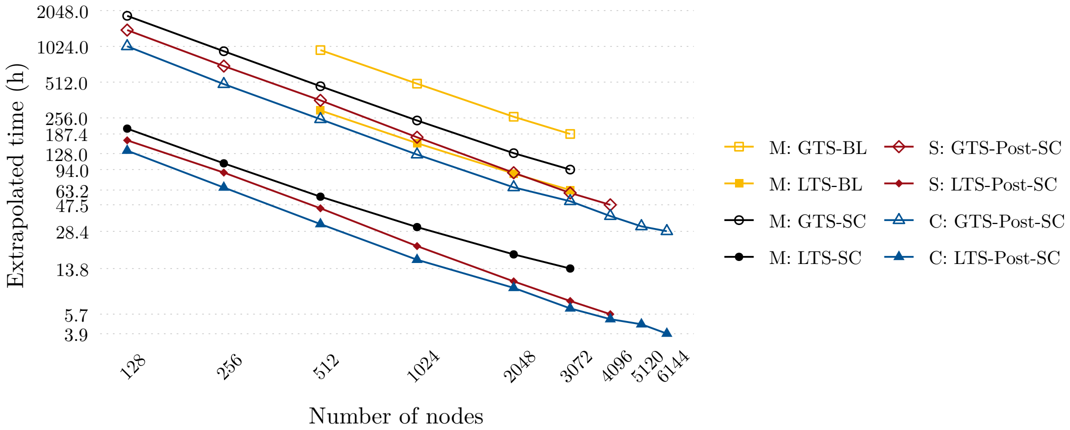
[6] Breuer et al., IPDPS '16, 854–863, 2016.

Clustered LTS for high-performance computing



For mesh with 221 million elements (111 billion DOFs)

Clustered LTS for high-performance computing



Theoretical speed-up of clustered LTS is 9.9, in practice 6.8-8.

Summary

Mesh generation for complex scenarios is time consuming, but...

- CAD and meshing tools improved a lot
- parallel automatic mesh generation works in practice for tetrahedral meshes (tested up to $\approx$ 1 billion elements)
- slivers can be handled with local time-stepping