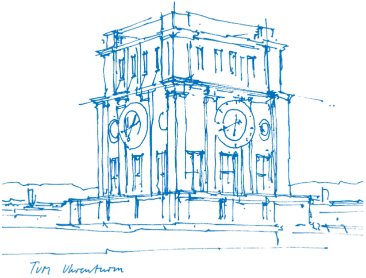
POP and ChEESE

SeisSol for Computational Earthquake Simulations with GPU-Aware MPI Communication for Local Time Stepping

Alice-Agnes Gabriel [1], Michael Bader [2], Ravil Dorozhinskii [2]
Ludwig Maximilian University of Munich [1]
Technical University of Munich [2]

July 7th 2021

PASC 2021, Virtual Conference



Outline of talk

ChEESE
Center of Excellence for Exascale in Solid Earth

European Commission

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823844

What is SeisSol?

SeisSol - software for simulating seismic waves and earthquake dynamic

based on:

- Discontinious Galerking method
- ADER time-integration scheme
- tetrahedral meshing

supports:

- elastic and visco-elastic wave propagation models
- plasticity model
- Local and Global Time Stepping schemes
- point sources and rupture surfaces to model source terms
- fused-simulations

originally came with:

- MPI+OpenMP parallelization
- code generator - YATeTo DSL, [4]

ADER-DG in a Nutshell

Update Scheme

$$
\begin{aligned}
Q_k^{n+1} = {} & Q_k^n + M^{-1}(K^\xi \mathcal{D}_k A_k^* + K^\eta \mathcal{D}_k B_k^* + K^\zeta \mathcal{D}_k C_k^*) \\
& - \frac{1}{|J|} M^{-1}\Big(\sum_{i=1}^{4} |S_i| F^{-,i} \mathcal{D}_k \widehat{A}_k^+\Big) \\
& - \frac{1}{|J|} M^{-1}\Big(\sum_{i=1}^{4} |S_i| F^{+,i,j_k,h_k} \mathcal{D}_{k(i)} \widehat{A}_{k(i)}^-\Big)
\end{aligned} \tag{1}
$$

Cauchy-Kowalewski

$$
\mathcal{D}_k = \sum_{j=0}^{\mathcal{O}-1} \frac{(t^{n+1} - t^n)^{j+1}}{(j+1)!} \frac{\partial^j}{\partial t^j} Q_k^n \tag{2}
$$

$$
\frac{\partial^{j+1}}{\partial t^{j+1}} Q_k^n = M^{-1} \left[(K^\xi)^T \Big(\frac{\partial^j}{\partial t^j} Q_k^n\Big) A_k^* + (K^\eta)^T \Big(\frac{\partial^j}{\partial t^j} Q_k^n\Big) B_k^* + (K^\zeta)^T \Big(\frac{\partial^j}{\partial t^j} Q_k^n\Big) C_k^* \right] \tag{3}
$$

Source Code Structure and Code Generation with YATeTo

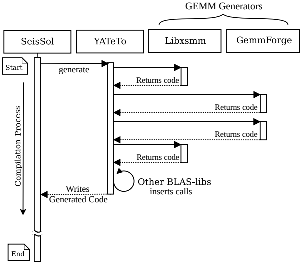


Figure: Compilation process (from [2])

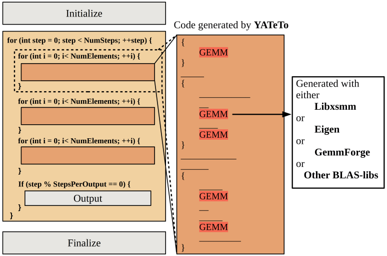


Figure: Simplified source code structure (from [2])

Listing: Example of YATeTo DSL

```
volumeSum = self.Q['kp']
for i in range(3):
volumeSum += self.db.kDivM[i][self.t('kl')] * self.I['lq'] * self.starMatrix(i)['qp']

volume = (self.Q['kp'] <= volumeSum)
generator.add('volume', volume)
```

GPU computing in SeisSol

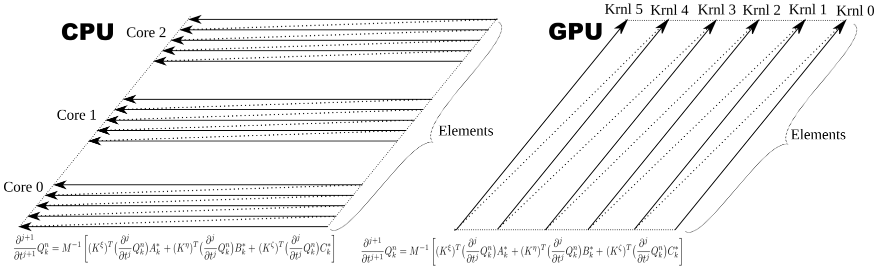


Figure: CPU/GPU task parallelism

Binary Batched Operations:

- trivial grid/block distribution
- easy to estimate run-time resources i.e., shared memory, registers

But:

- finer granularity w.r.t CPU-like parallelism
- lower arithmetic intensity

GemmForge

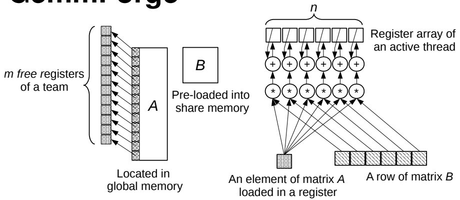


Figure: Sum of parallel outer products (from [2])

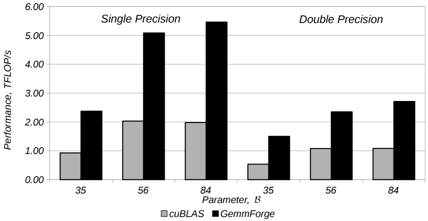


Figure: GemmForge vs. cuBLAS (from [2])

Benchmark:

$$L_e = D \cdot A_e \cdot B_e + L_e \qquad (4)$$

where $L, A \in \mathbb{R}^{\mathcal{B}\times 9}$ and $B \in \mathbb{R}^{9\times 9}$. $D \in \mathbb{R}^{\mathcal{B}\times\mathcal{B}}$ represents either a mass or stiffness matrix.

Implementation:

$$T_e = A_e \cdot B_e$$
$$L_e = D \cdot T_e + L_e \qquad (5)$$

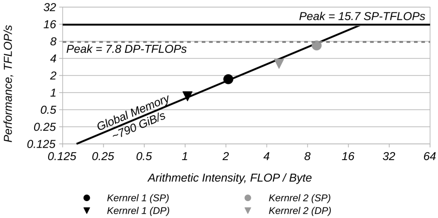


Figure: Roofline model analysis (from [2])

POP audit I

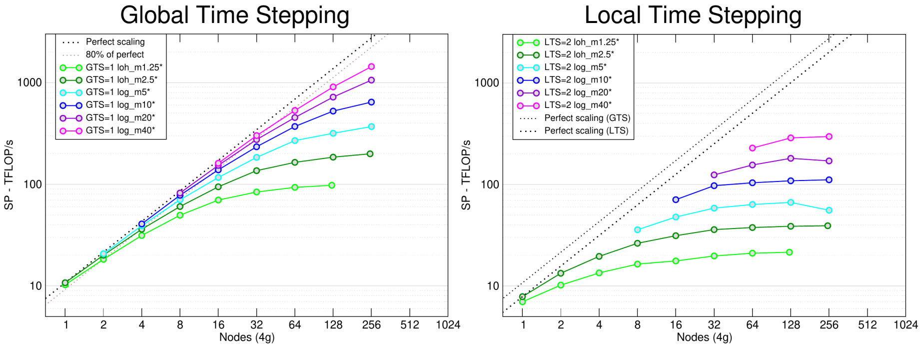


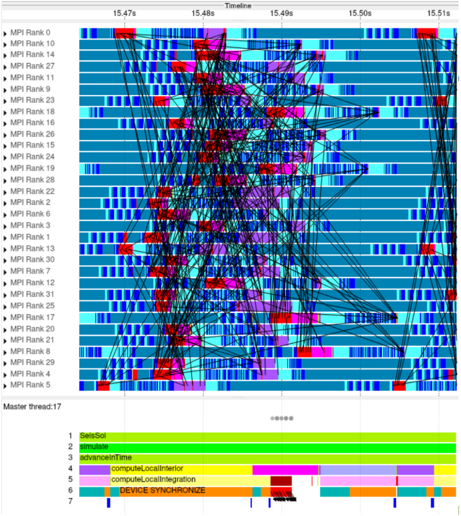
Figure: Strong/Weak scaling of SeisSol using LOH.1 benchmark obtained on Marconi 100

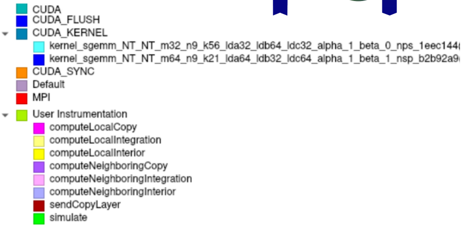
Conclusion

Computation scaling and communication efficiency rapidly deteriorate for LTS

MPI communication cost grows progressively with scale

POP audit II





- GPUs idle during message exchange
- Rank 17 starts and finishes later than the other ranks

  CPU predominantly in CUDA synchronization while kernels execute on GPU
- In general, traces and analysis are much more complicated for LTS scheme

Figure: Execution timeline (single step) for GTS

LTS in a Nutshell

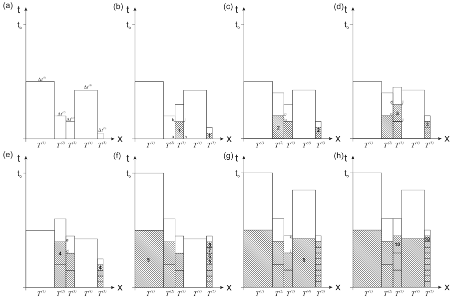


Figure: Local time stepping in motion (from [3])

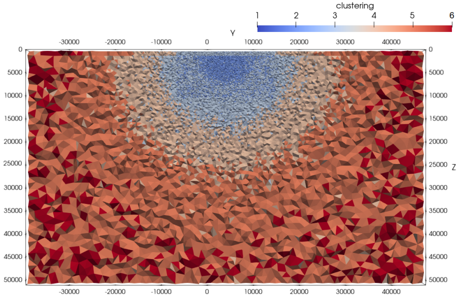


Figure: Example of elements distribution over 6 LTS clusters (from [2])

Courant-Friedrichs-Lewy condition:

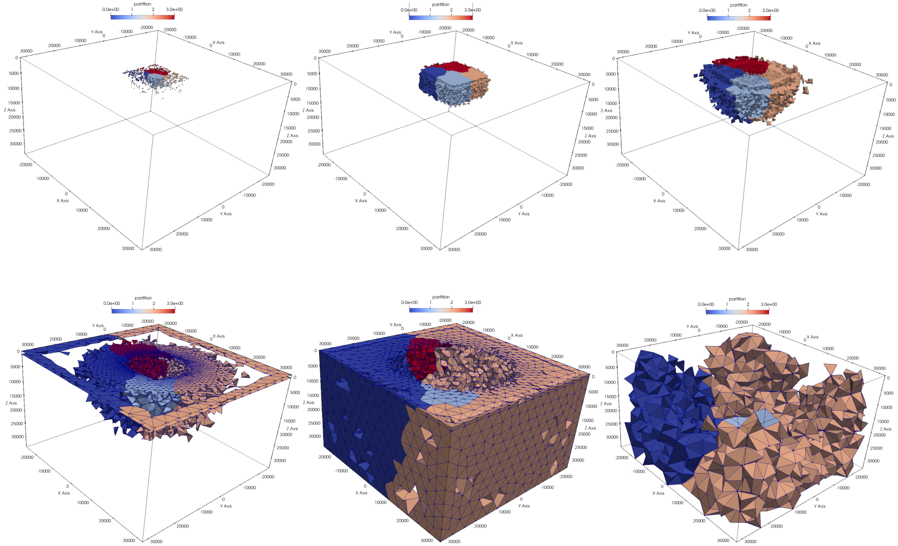
- necessary condition for convergence
- determined by local wave speed and element size

Workload per element, proposed by Breuer, Heinecke, and Bader in [1]:

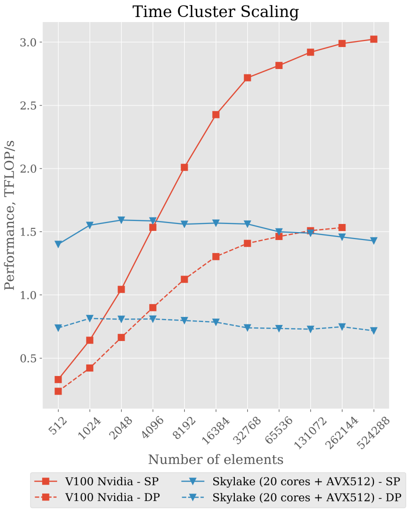
$$w_k = R^{L-l_k} \tag{6}$$

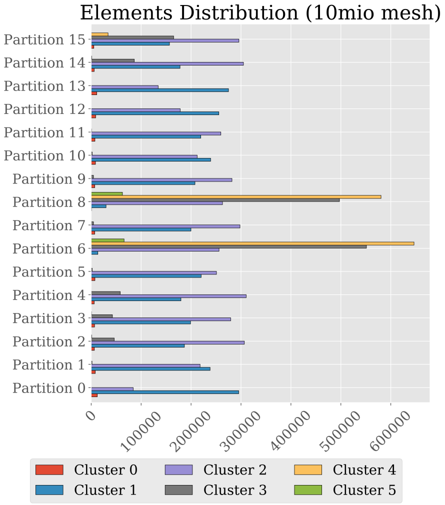
where R is update cluster ratio, L is the total number of clusters and l_k is a linear index of the time cluster to which element k belongs.

Time Clustering & Mesh partitioning



2 Inherited Problems





Balancing Strategies I

1. Original without any memory balancing:

$$w_k = R^{L-l_k} \tag{7}$$

   denoted as "exponential"

2. Exponential LTS weights with memory balancing:

$$w_k \in \mathbb{R}^2 \mid w_k = \begin{bmatrix} R^{L-l_k} \\ 1 \end{bmatrix} \tag{8}$$

   denoted as "exponential balanced"

3. Equal time clusters partitioning:

$$w_k \in \mathbb{R}^L \mid w_k^i = \begin{cases} 1, \text{if } i = l_k \\ 0, \text{otherwise} \end{cases} \tag{9}$$

   denoted as "encoded"

Balancing Strategies II

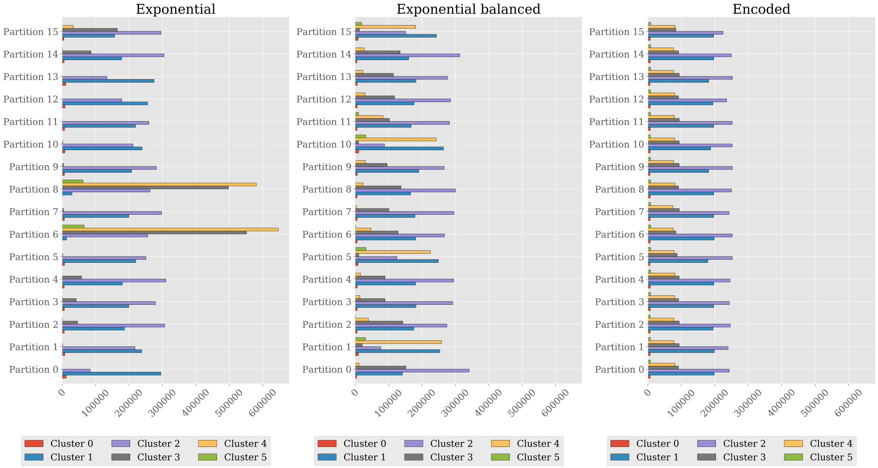
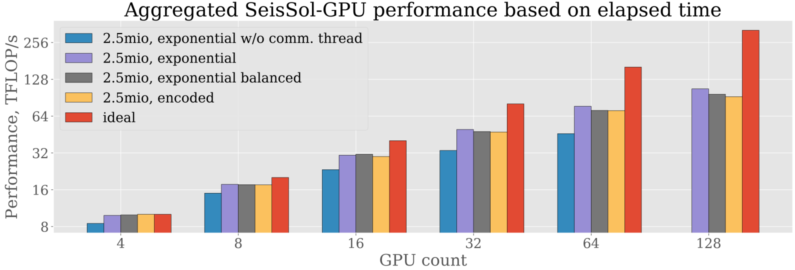
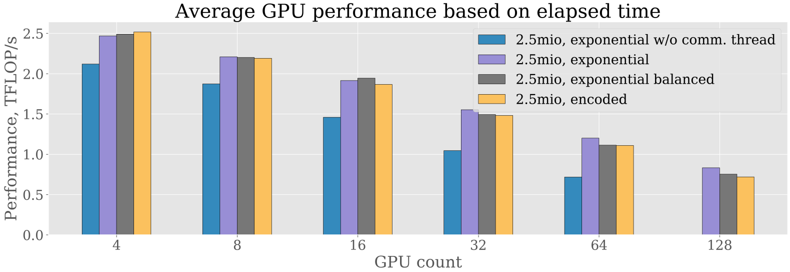


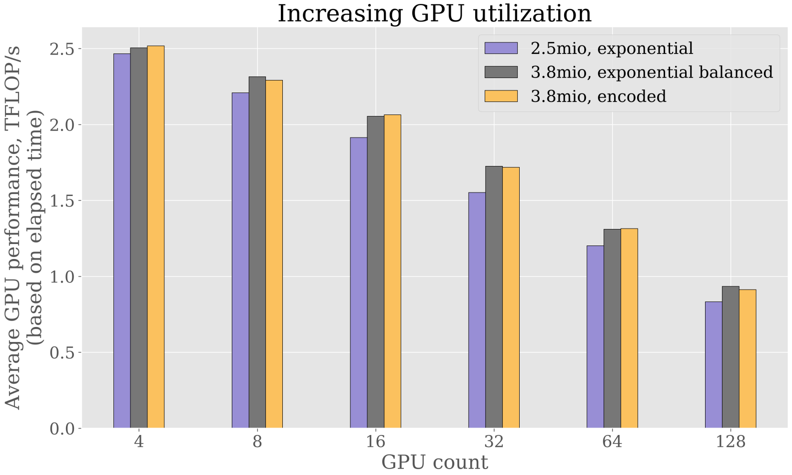
Figure: Distribution of 10mio elements over 16 partitions

Strong Scaling I

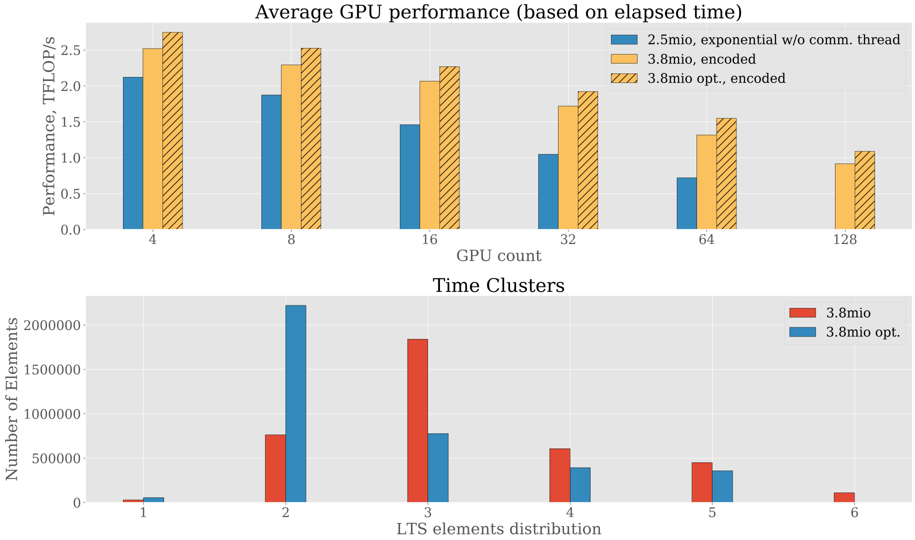




Strong Scaling II



Strong Scaling III: Improving Mesh Quality



Tracing SeisSol Proxy

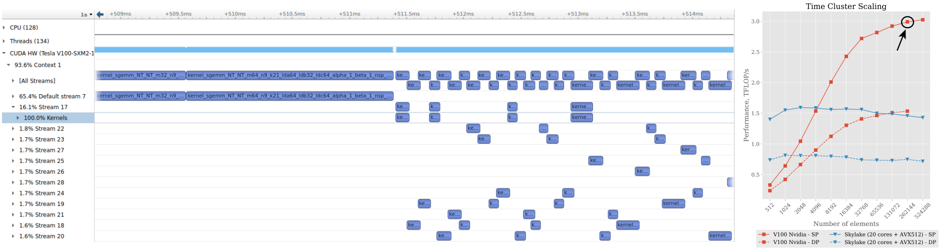


Figure: Time Cluster with 262144 (2^{18}) elements

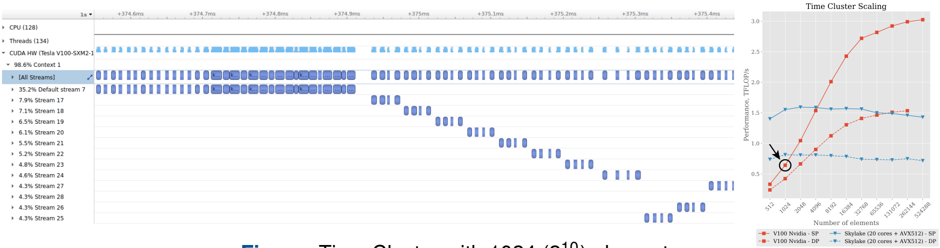
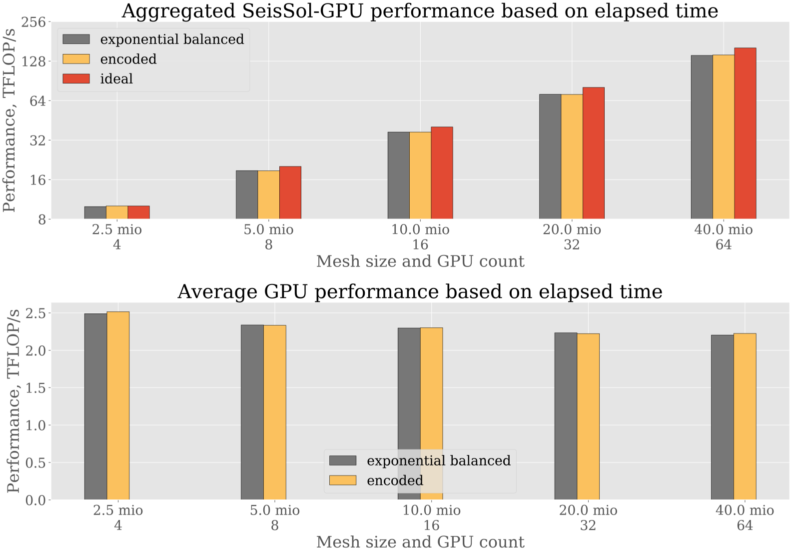


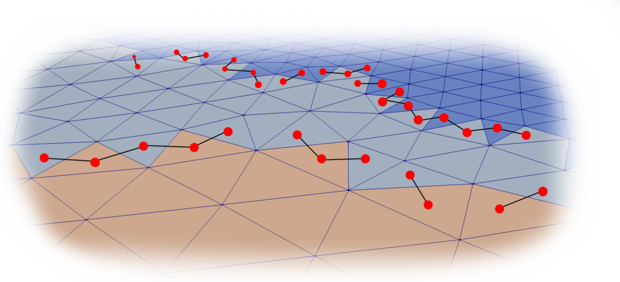
Figure: Time Cluster with 1024 (2^{10}) elements

Weak Scaling



Conclusion

- Algorithmic and hardware problems
  seems to be a general problem for GPU-LTS implementations
- Found two workload and memory balancing strategies
  a new research direction
- Weak scaling was achieved and looks reasonably good
- Communication may be further improved
  adding heavy edges along time cluster boarders



References I

[1] Alexander Breuer, Alexander Heinecke, and Michael Bader. "Petascale local time stepping for the ADER-DG finite element method". In: *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2016, pp. 854–863.

[2] Ravil Dorozhinskii and Michael Bader. "SeisSol on Distributed Multi-GPU Systems: CUDA Code Generation for the Modal Discontinuous Galerkin Method". In: *The International Conference on High Performance Computing in Asia-Pacific Region*. 2021, pp. 69–82.

[3] Michael Dumbser, Martin Käser, and Eleuterio F Toro. "An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes-V. Local time stepping and p-adaptivity". In: *Geophysical Journal International* 171.2 (2007), pp. 695–717.

[4] Carsten Uphoff and Michael Bader. "Yet Another Tensor Toolbox for discontinuous Galerkin methods and other applications". en. In: *ACM Transactions on Mathematical Software* 46.4 (2020). DOI: 10.1145/3406835.