Solving Hyperbolic PDE Systems with ExaHyPE

M. Bader, J.-M. Gallard, L. Rannabauer, A. Reinarz (Technical University of Munich)
M. Dumbser (Univ. Trento), A.-A. Gabriel (LMU Munich), T. Weinzierl (Durham Univ.)

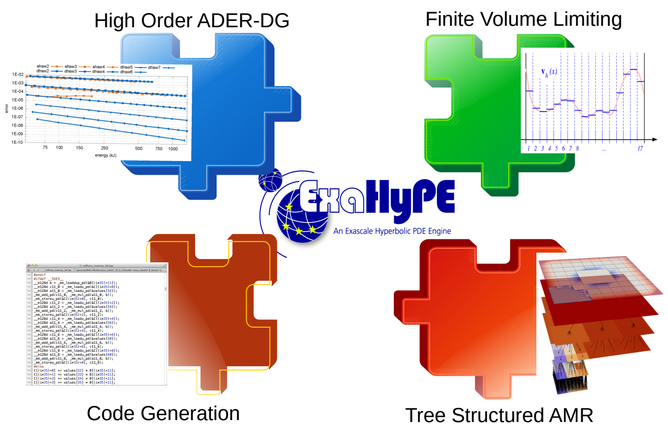
Towards an Exascale *PDE Engine*

ExaHyPE is designed to enable medium-sized interdisciplinary research teams to quickly realise extreme-scale simulations of grand challenges.

The **ExaHyPE Engine** solves systems of first-order hyperbolic PDEs of the form:

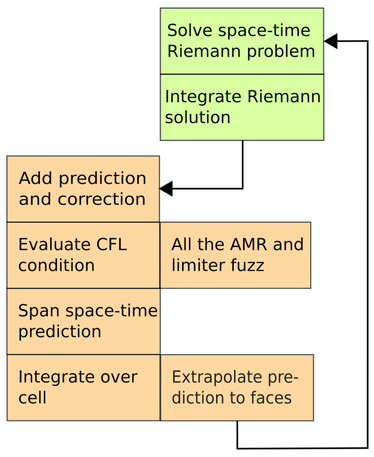
$$\mathbf{P}\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{Q}, \nabla \mathbf{Q}) + \sum_{i=1}^{d} \mathbf{B}_i(\mathbf{Q}) \frac{\partial \mathbf{Q}}{\partial x_i} = \mathbf{S}(\mathbf{Q}) + \sum \delta$$

ExaHyPE employs higher-order ADER-DG on tree-structured adaptive Cartesian grids using a-posteriori subcell Finite-Volume limiting [4]:



Parallel AMR and Tasking:

- based on the Peano framework: www.peano-framework.org
- 2D and 3D dynamic mesh refinement on tree-structured Cartesian grids
- task-based implementation of space-time predictor, corrector, limiter steps
  → single amortized traversal per time step [2]
- shared memory parallelisation through Intel's Threading Building Blocks (TBB)
- distributed memory parallelisation with MPI (Peano framework)



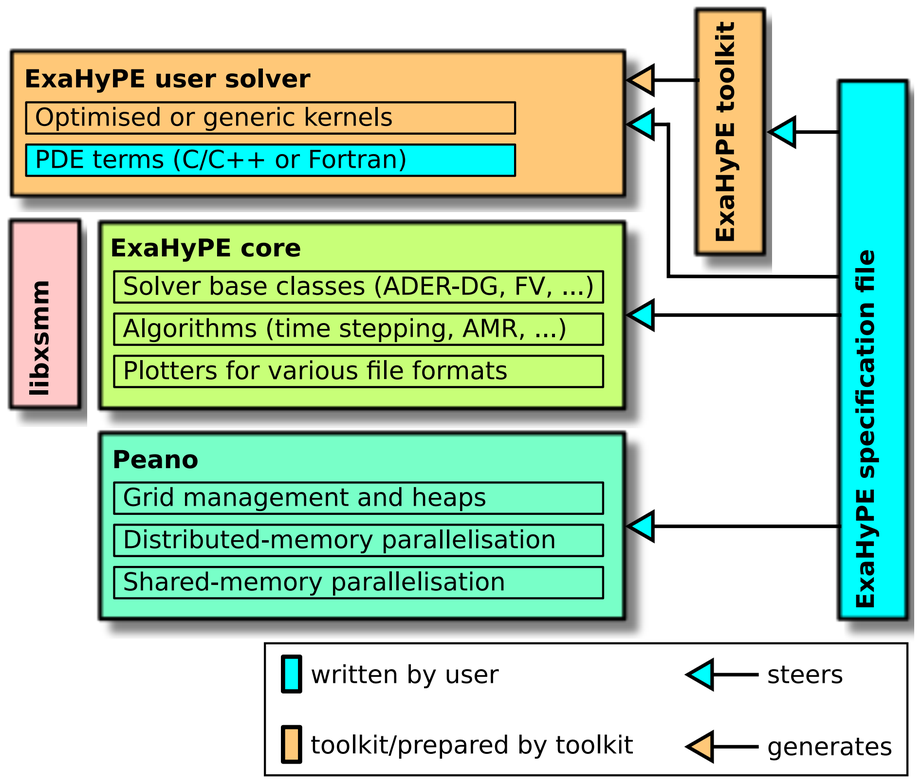
Role-Oriented Code Generation:

Toolkit and *Code Generator* [3] provide views for:

- **application expert(s)**: straightforward user API that hides complexity of solver and optimization
- **algorithms expert(s)**: architecture-oblivious algorithm design via custom macros that isolate low-level optimizations
- **optimization expert(s)**: templating logic, e.g. for hardware-aware optimizations

Toolkit and Code Generator are stand-alone applications based on the Jinja2 templating engine.

How to Create Code that is Easy to Use & Extend, Flexible, Efficient, . . . ?



Using the ExaHyPE Toolkit:

1. create a specification file that defines the domain, PDE system, required architecture, parallelisation, etc.
2. ExaHyPE toolkit creates glue code, application-specific template classes and core routines (tailored to application and architecture)
3. implement the application classes with PDE- and scenario-specific methods:
   - `flux(...)`, `ncp(...)`, ... for PDE terms (conservative fluxes, non-conservative products, etc.)
   - `eigenvalues(...)` to compute eigenvalues (for Riemann solvers)
   - `boundaryValues(...)`, etc.

Creating an ExaHyPE Application

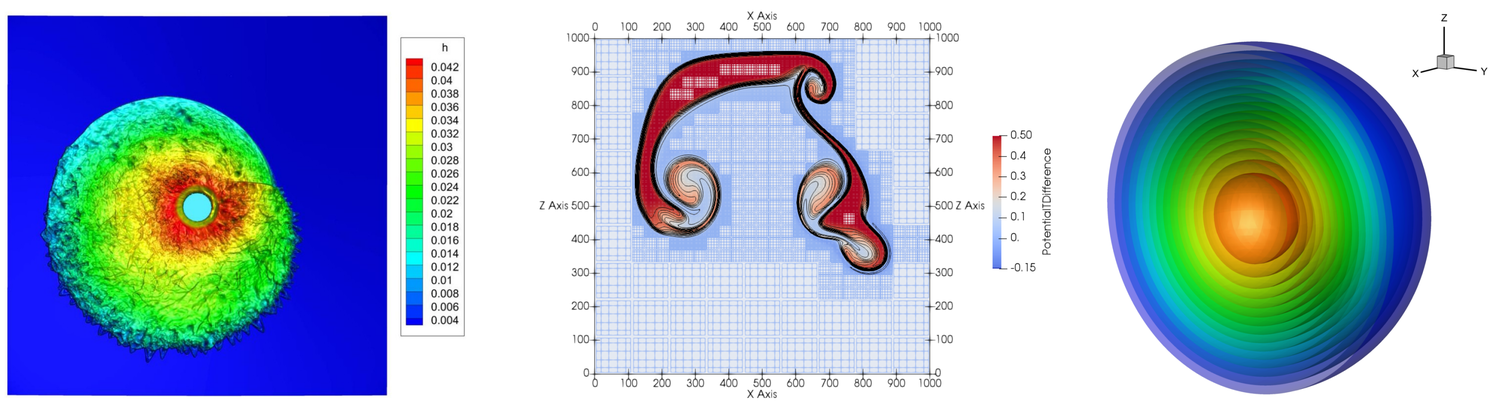
Specification file:

```
exahype-project Elastic
  peano-kernel-path const = ./Peano
  exahype-path const      = ./ExaHyPE
  output-directory const  = ./Elastic

  computational-domain
    dimension const     = 3
    offset              = 0.0, 0.0, 0.0
    width               = 1.0, 1.0, 1.0
    end-time            = 1.0
  end computational-domain

  solver ADER-DG ElasticWaveSolver
    variables const     = v:3,sigma:6
    parameters const    = rho:1,cp:1,cs:1
    order const         = 7
    maximum-mesh-size   = 2e-2
    maximum-mesh-depth  = 2
    terms const         = flux,ncp,
      material_parameters,point_sources
    optimisation const  = optimised
    language const      = C
    basis               = Lobatto
  end solver
end exahype-project
```

Implementation of flux function:

```
void Elastic::ElasticWaveSolver
  ::flux(const double* const Q,
         double** const F) {

  // PDE-specific implementation:
  VariableShortcuts s;
  double sigma_xx=Q[s.sigma + 0];
  double sigma_yy=Q[s.sigma + 1];
  double sigma_zz=Q[s.sigma + 2];
  double sigma_xy=Q[s.sigma + 3];
  double sigma_xz=Q[s.sigma + 4];
  double sigma_yz=Q[s.sigma + 5];

  F[0][ s.v + 0] = -sigma_xx;
  F[0][ s.v + 1] = -sigma_xy;
  F[0][ s.v + 2] = -sigma_xz;

  F[1][ s.v + 0] = -sigma_xy;
  F[1][ s.v + 1] = -sigma_yy;
  F[1][ s.v + 2] = -sigma_yz;

  F[2][ s.v + 0] = -sigma_xz;
  F[2][ s.v + 1] = -sigma_yz;
  F[2][ s.v + 2] = -sigma_zz;
}
```

Application Examples: SWASI Experiment, Cloud Formation, TOV Star



References

[1] A. Reinarz et al.: *ExaHyPE: An engine for parallel dynamically adaptive simulations of wave problems*, under review (Comp. Phys. Comm.), https://arxiv.org/abs/1905.07987.

[2] D. E. Charrier, T. Weinzierl: *Stop talking to me – a communication avoiding ADER-DG realisation*, under review (SIAM J. of Scient. Comput.), https://arxiv.org/abs/1801.08682.

[3] J.-M. Gallard et al.: *Role-oriented code generation in an engine for solving hyperbolic PDE systems.* 2019 Int. Workshop on Softw. Eng. for HPC-Enabled Research (SE-HER).

[4] O. Zanotti, F. Fambri, M. Dumbser, A. Hidalgo: *Space-time adaptive ADER discontinuous Galerkin finite element schemes with a posteriori sub-cell finite volume limiting.* Computers & Fluids 118, 2015, p. 204–224.

Download the ExaHyPE engine from:
www.ExaHyPE.org

Acknowledgments

ExaHyPE was developed as a joint project of:

in particular by:

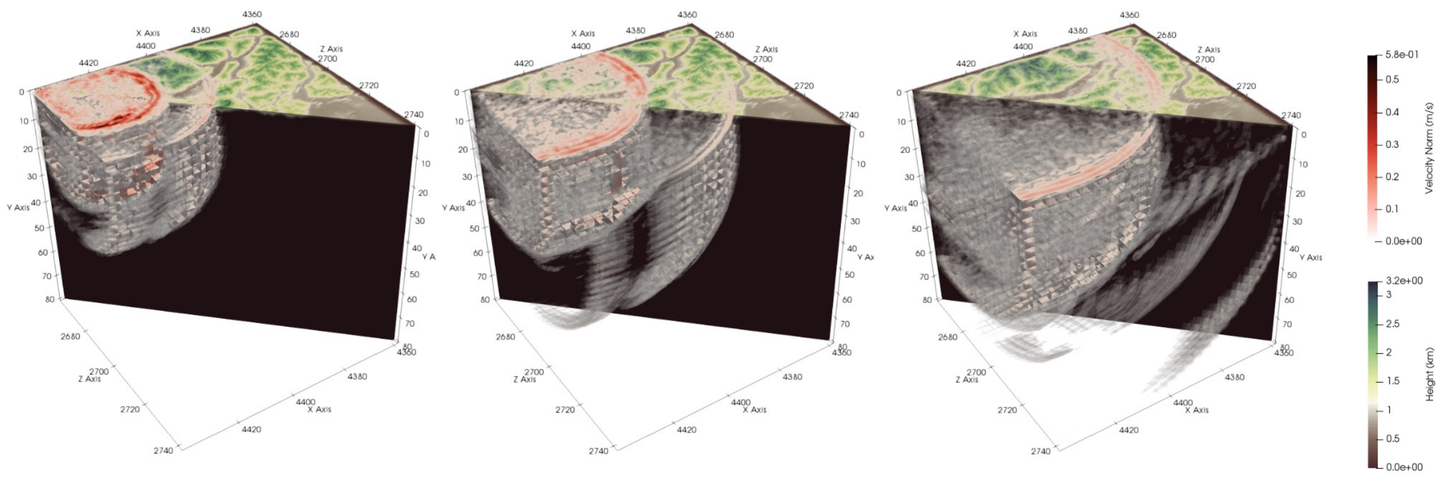
Dominic Charrier, Benjamin Hazelwood, Tobias Weinzierl (University of Durham), Michael Dumbser, Francesco Fambri, Maurizio Tavelli, Olindo Zannotti (University of Trento), Alice Gabriel, Kenneth Duru (Ludwig-Maximilians-University Munich), Luke Bovard, Luciano Rezzolla, Sven Köppel (Frankfurt Institute for Advanced Studies), Jean-Mathieu Gallard, Leonhard Rannabauer, Anne Reinarz, Philipp Samfaß, Angelika Schwarz and Vasco Varduhn (Technical University of Munich).
We thank the Leibniz Supercomputing Centre and the Russian Academy of Sciences for their support.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 823844 (project ChEESE, https://cheese-coe.eu/) and No 671698 (www.exahype.eu).

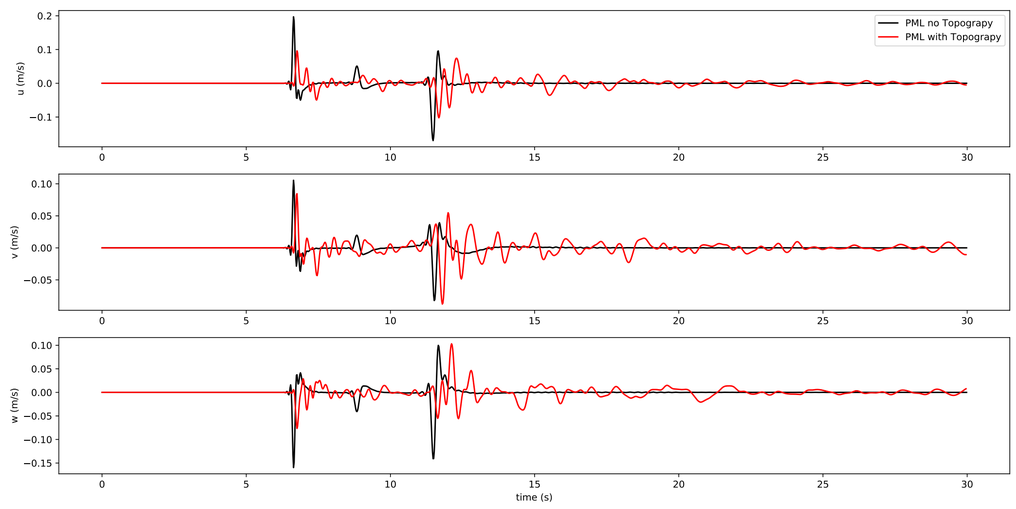
ExaSeis: Seismic Simulations with ExaHyPE

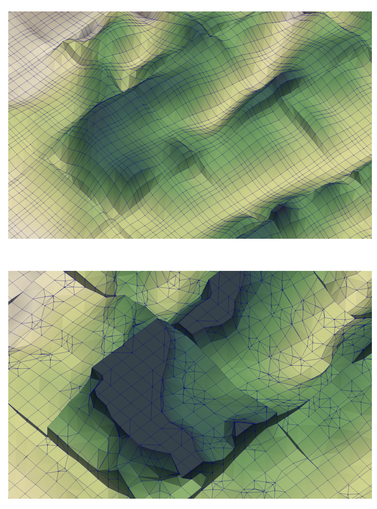
L. Rannabauer, J.-M. Gallard, A. Reinarz, M. Bader (Technical University of Munich)
K. Duru, D. Li, A.-A. Gabriel (LMU Munich), M. Tavelli, M. Dumbser (University of Trento)

Wave Propagation with Complex Topography



We analyze wave scattering effects on complex, free-surface topographies. For that purpose we designed a scenario in the Alpine region including the Zugspitze and run simulations with the same initial conditions with and without topography. The choice of method is crucial as DIM and Curvilinear Mesh show high differences in the representation of the topography.





Towards Urgent Seismic Simulation – A ChEESE Pilot Demonstrator (www.cheese-coe.eu):

This pilot shall explore the possibilities of employing urgent supercomputing to obtain fast (hours) shaking maps for regions affected by recent earthquakes. We created an interface in ExaHyPE to the MUQ C++ toolbox for uncertainty quantification (muq.mit.edu) and plan to exploit the strengths of ExaSeis (fast initialization of single forward simulations as no meshing is required) to compute UQ-based shaking maps.

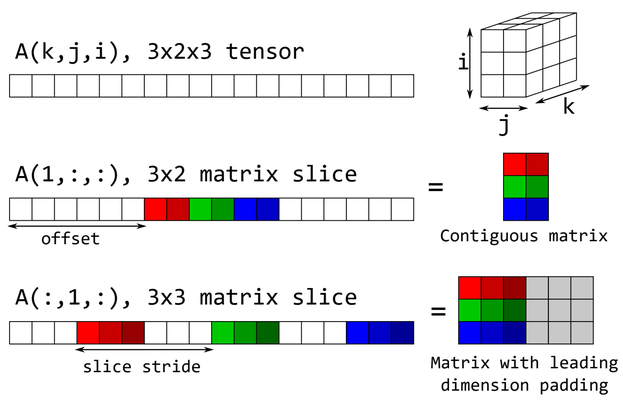
Optimized Kernels: Vectorization and Minimization of Memory Footprint

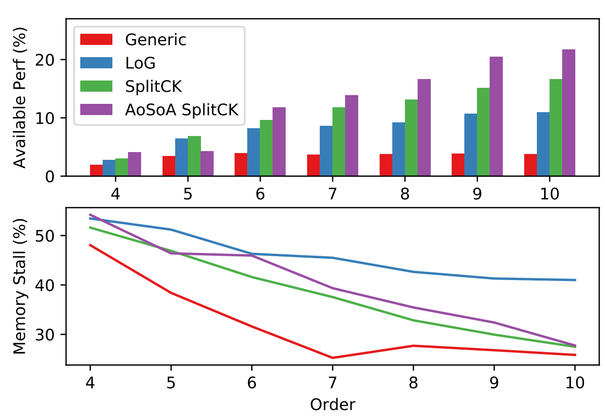
ExaSeis faces conflicting demands for data layout:

- DG tensor operations are turned into sequences of matrix multiplications (“loop over gemm”) → suggests quantities as leading dimension (AoS)
- evaluation of fluxes loops over integration points calling user-functions (`flux()`, e.g.) → suggests integration points as leading dimension (SoA)
- choose AoSoA as data layout: → single out one dimension
- In addition: provide dimensional `flux()` function to reduce the memory footprint

```
// scalar version of flux_x
void flux_x(double* Q, double* F) {
  F[0] = -Q[3];
  F[1] = -Q[6];
  F[2] = -Q[7];
}

// vectorized formulation of flux_x
void flux_x_vect(double* Q, double* F) {
  #pragma omp simd aligned(Q,F:ALIGNMENT)
  for(int i=0; i<VECTLENGTH; i++) {
    F[0*VECSTRIDE+i] = -Q[3*VECSTRIDE+i];
    F[1*VECSTRIDE+i] = -Q[6*VECSTRIDE+i];
    F[2*VECSTRIDE+i] = -Q[7*VECSTRIDE+i];
  }
}
```

Extracting matrix slices from a tensor *A*:



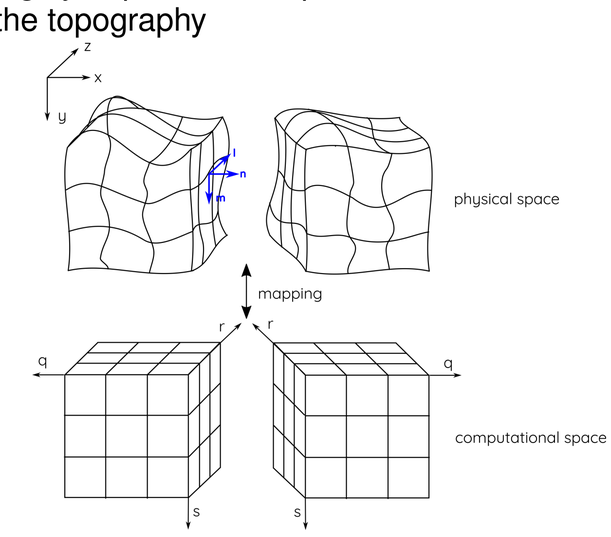


- significantly reduces the L2-cache footprint
- 5.7× speedup for order 10 compared to generic implementation.

ADER-DG on Curvilinear Meshes

ExaHyPE is based on Cartesian Meshes. To allow boundary fitting meshes, we developed a curvilinear method that maps Cartesian to curvilinear elements:

- retains the tensor structure of the DG basis
- allows automated initial mesh generation
- flux and source terms of the system are transformed with the element Jacobian
- but: eigenvalues (and thus the time-step size) highly depend on the perturbation introduced by the topography



Diffuse Interface Method

In this approach, we work with strictly Cartesian (adaptive) meshes, but augment the elastic wave equation with a color function α, which represents the location of solid medium and vacuum:

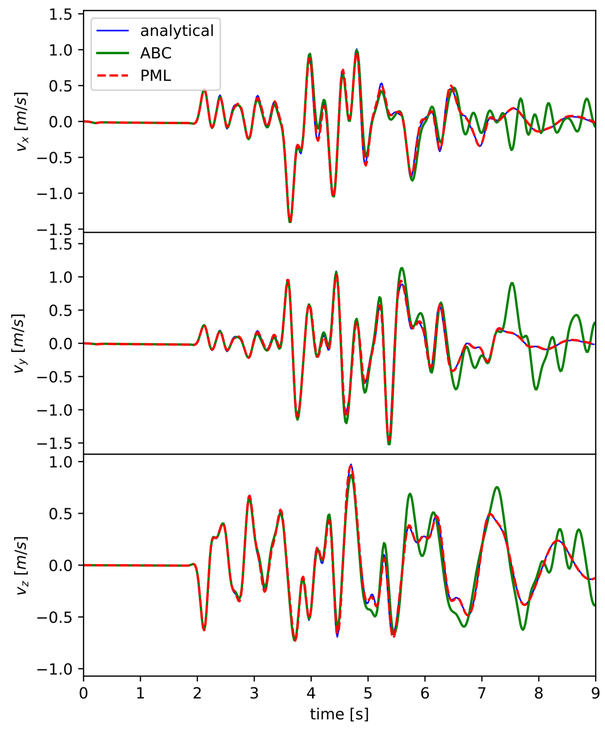
$$\frac{\partial \sigma}{\partial t} - \mathbf{E} \cdot \frac{1}{\alpha} \nabla(\alpha v) + \mathbf{E} \cdot v \otimes \nabla \alpha = 0,$$
$$\frac{\partial \alpha v}{\partial t} - \frac{\alpha}{\rho} \nabla \cdot \sigma - \frac{1}{\rho} \sigma \nabla \alpha = 0$$

- α represents the volume fraction of the solid medium ($\rightsquigarrow$ Baer-Nunziato approach)
- completely avoids problems with mesh generation
- time step sizes (and shape of grid cells) are not influenced by topography
- but: wherever $\alpha \neq 1$, fluxes are no longer linear (requires limiting!)

Perfectly Matched Layers

PML entirely avoids wave reflections at boundaries. We can place point sources and receivers close to boundaries and keep the domain small.

- based on complex coordinate stretching
- requires extension of the numerical DG fluxes, inter-element and boundary procedures to ensure numerical stability
- only applied in cells close to the boundary



Acknowledgments

ExaSeis is a joint development of:

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements No 823844 (project ChEESE, https://cheese-coe.eu/) and No 671698 (www.exahype.eu).

References

[1] K. Duru et al.: *A new discontinuous Galerkin method for elastic waves with physically motivated numerical fluxes* J. Comp. Phys. 386, 2019, submitted

[2] M. Tavelli et al.: *A simple diffuse interface approach on adaptive Cartesian grids for the linear elastic wave equations with complex topography.* J. Comp. Phys. 386, p. 158–189.

[3] K. Duru et al.: *A stable discontinuous Galerkin method for the perfectly matched layer for elastodynamics in first order form”* Numerische Mathematik, 2019, submitted