Discontinuous Galerkin & Julia Project Report
Adaptive Mesh Refinement

M.Sc. Praktikum: Modern Wave Propagation - Discontinuous Galerkin & Julia (WS24/25)

Riccardo Capellupo, Elham Rahmani, Matteo Troppina
School of Computation, Information and Technology
Techinische Universität München, Boltzmannstraße 3, Garching, 85748, Munich, Germany

Adaptive Mesh Refinement (AMR) is a computational technique that dynamically adjusts mesh resolution to efficiently capture localized features requiring high accuracy, while coarsening less critical regions to optimize the total number of cells in the simulation. In our current work, we developed an AMR module in Julia by extending an already implemented matrix-based grid with a *quadtree*-based adaptive structure. A custom library was developed to handle the quadtree dynamics and support the specific operations needed in our application; however, as this is not the primary focus of the project, we refer the reader to [3] for further details. The system allows the user to choose between the original uniform grid approach and the new adaptive method by changing the configuration file, depending on the problem requirements. This report presents the implementation, with a primary focus on the refinement logic and the challenges posed by inter-cell communication within the underlying Discontinuous Galerkin (DG) framework in the adaptive setting.

I. Dynamic Mesh

At the beginning of the execution, the computational mesh is initialized as a uniform grid. A refinement step is then performed based on the indicator described in Section III, applied to the initial state of the system. The number of elements along each spatial dimension must be a power of two, as the initial quadtree is constructed recursively from the root node by successive refinement until the desired initial resolution of n^2 is reached.

To simplify the data structure and ensure consistent communication between cells, we impose a restriction that each cell can have at most two neighbors in any given direction. This enforces a 2:1 balancing constraint on the tree, meaning that neighboring cells can differ in refinement level by at most one. As a result, any local refinement or coarsening requires checking and potentially updating neighboring cells to maintain this constraint. Finally, to ensure correct inter-cell communication, each cell maintains an up-to-date list of neighboring cells in all spatial directions.

II. inter-cell communication

To enable communication between neighboring cells of different sizes, the degrees of freedom (DOFs) and fluxes on their shared faces must be evaluated consistently. This is achieved by evaluating the neighboring cell's DOFs and fluxes at points corresponding to the quadrature nodes on the reference face of the current cell. Since each cell operates within its own reference space, appropriate coordinate mappings are required to establish correspondence between face points.

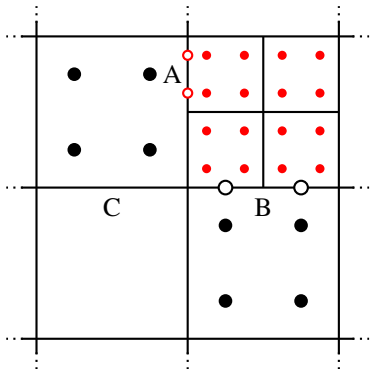

Fig. 1. Example of communication between cells of different size

In Fig. 1(A), a cell at refinement level ℓ (fine) evaluates the flux on its west face, which it shares with a larger neighboring cell at level $\ell - 1$ (coarse). Let $\boldsymbol{v} = (\boldsymbol{v}_x, \boldsymbol{v}_y)^T$ denote the quadrature nodes on the reference face of the level-ℓ cell. Let $\boldsymbol{c}^\ell$ and $\boldsymbol{c}^{\ell-1}$ denote the centers of the fine and coarse cells, respectively. Without loss of generality, we define the mapping:

$$\boldsymbol{v} \mapsto \begin{pmatrix} 1 - \delta_{\boldsymbol{v}_x,1} \\ \dfrac{\boldsymbol{v}_y}{2} + \dfrac{1}{2} \cdot \delta_{\boldsymbol{c}_y^\ell \geq \boldsymbol{c}_y^{\ell-1}} \end{pmatrix} = \boldsymbol{v}^*$$

This mapping is valid since $\boldsymbol{v}_x \in \{0, 1\}$ on the reference face. The coarse cell's DOFs and fluxes are then evaluated at the mapped points $\boldsymbol{v}^*$.

In Fig. 1(B), cell at refinement level $\ell - 1$ evaluates the flux on its north face, which is shared with two finer neighbors at level ℓ. The quadrature points on the reference face of the coarse cell are split into two subsets, $\boldsymbol{v}^{(1)}$ and $\boldsymbol{v}^{(2)}$, each corresponding to one of the neighboring fine cells. Without

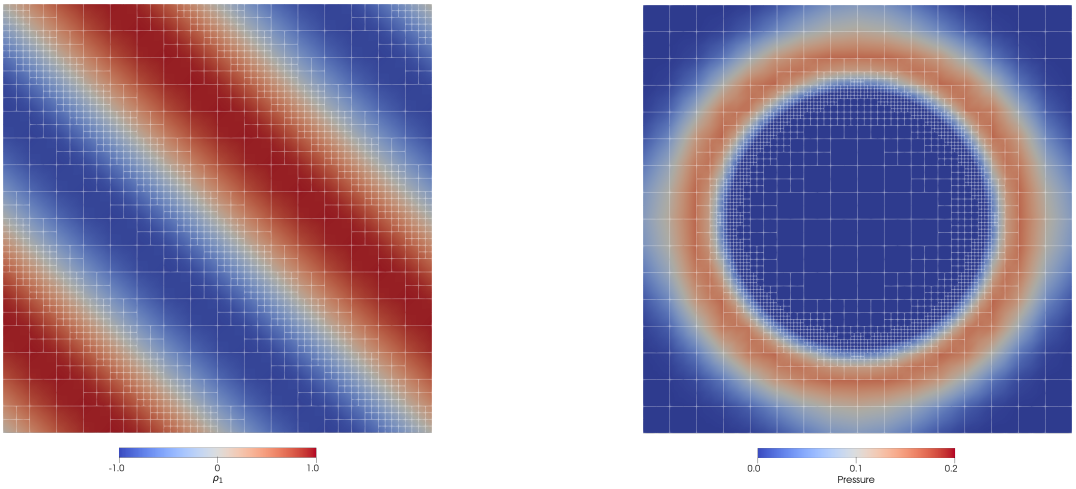

Fig. 2. Simulation results with AMR: (left) advection equation showing scalar field evolution; (right) acoustic equation illustrating pressure distribution.

loss of generality, the mapping for $\boldsymbol{v}^{(1)}$ is defined as:

$$\boldsymbol{v}^{(1)} \mapsto \begin{pmatrix} 2 \cdot \boldsymbol{v}_x^{(1)} - \frac{1}{2}\delta_{\boldsymbol{c}_{1,x}^{\ell} \geq \boldsymbol{c}_x^{\ell-1}} \\ 1 - \delta_{\boldsymbol{v}_x,1} \end{pmatrix} = (\boldsymbol{v}^{(1)})^*$$

If two neighboring cells are of the same size, no mapping is necessary and the face projection can be applied directly [1].

III. Mesh Update

The update is carried out cell by cell, following the node's Morton index [4] and sorting the leaves of the tree accordingly.

The decision to refine or coarsen each cell k is guided by a total variation indicator TV_k [2] and a user-specified DOF $\boldsymbol{u}$ that requires tracking. In our implementation, we employ a simple indicator computed as $\mathrm{TV}_k = \|[dx, dy]_k\|_2$.

$$dx = \frac{1}{n \cdot \Delta x} \sum_{i=1}^{n} (u_E^{(i)} - u_W^{(i)})$$

$$dy = \frac{1}{n \cdot \Delta y} \sum_{i=1}^{n} (u_N^{(i)} - u_S^{(i)})$$

The cell is then updated based on:

$$\begin{cases} \text{refine}, & \text{if } \mathrm{TV}_j \geq \tau_{\text{refine}}, \\ \text{coarsen}, & \text{if } \mathrm{TV}_j < \tau_{\text{coarsen}} \\ \text{nothing} & \text{otherwise} \end{cases}$$

where $\tau_{\text{coarsen}} = \mu + \sigma \cdot T_{\text{coarsen}}$ and $\tau_{\text{refine}} = \mu + \sigma \cdot T_{\text{refine}}$, where

$$\mu = \frac{1}{N} \sum_{k=1}^{N} \mathrm{TV}_k, \quad \sigma = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (\mathrm{TV}_k - \mu)^2}$$

while T is a fixed custom threshold and N is the number of cells. As the mesh evolves, solution data must be consistently transferred across different levels of resolution. When a cell is refined, the solution from the coarse parent is interpolated onto the quadrature points of the newly created finer children. In the case of coarsening, the fine-level data from the child cells are interpolated back onto the quadrature points of the parent cell. This approach ensures continuity and accuracy of the numerical solution throughout the update process.

The time-step update rule [1] is also adjusted to reflect the current maximum refinement level ℓ, since the smallest cell size becomes $h(\ell) = s \cdot 2^{-\ell}$, where s denotes the size of the domain.

IV. Results and Conclusions

In Fig. 2, two snapshots from simulations of the advection and acoustic equations [1] using the AMR framework are presented. The initial computational mesh consisted of 16×16 cells, and a second-order interpolation scheme was employed. The AMR solver demonstrates reliable performance in both solution accuracy and spatial adaptivity.

It is important to emphasize that the primary goal of this work was not to develop a more efficient solver in terms of computational speed or accuracy, but rather to implement an AMR algorithm within a discontinuous Galerkin (DG) framework. Future research should focus on rigorously evaluating the solver's performance. Potential directions include leveraging adaptivity to refine mesh regions around embedded objects or walls, increasing flexibility in mesh refinement by allowing a variable number of neighbors per face for each cell, and exploring parallelization strategies to improve computational efficiency.

References

[1] L. Krenz, A. Reinarz, S. Wolf and M. Marot-Lassauzaie, 'Discontinuous Galerkin Methods for Hyperbolic Problems'.

[2] L. Krenz, L. Rannabauer, and M. Bader, "A High-Order Discontinuous Galerkin Solver with Dynamic Adaptive Mesh Refinement to Simulate Cloud Formation Processes," vol. 12043, 2020, pp. 311–323. doi: 10.1007/978-3-030-43229-4_27.

[3] F. Jaillet and C. Lobos, "Fast Quadtree/Octree adaptive meshing and re-meshing with linear mixed elements," Engineering with Computers, 2021, doi: 10.1007/s00366-021-01330-w.

[4] M. Kirilin and C. Burstedde, "Alternative quadrant representations with Morton index and AVX2 vectorization for AMR algorithms within the p4est software library," Aug. 24, 2023, arXiv: arXiv:2308.13615. doi: 10.48550/arXiv.2308.13615.