Adaptive Mesh Refinement

Asger Sørensen, Matias Fjeldmark and Nicolai Riis
02623 - The Finite Element Method for Partial Differential Equations

Problem

The goal is to create a partial differential equation solver that uses an Adaptive Mesh Refinement (AMR) algorithm. The adaptive algorithm attempts to refine the mesh only in necessary areas reducing the needed computation time and degrees of freedom. This is done by using a special error estimator and then refining only elements in which the estimated error is larger than a set tolerance.
Additionally, we show another use of FEM other than solving PDEs, namely compressing images.

Error Estimate

The AMR algorithm needs an error estimator to mark elements for refinement.
One can approximate volume below an element as

$$\mathrm{vol}(\hat{u}(e_i)) \approx \frac{\Delta}{3}(\hat{u}_1 + \hat{u}_2 + \hat{u}_3)$$

where Δ is the area of e_i.

The element e_i is refined into, say, $e_i^{(1)}, \ldots, e_i^{(N)}$.
The change in volume is used as an error estimate

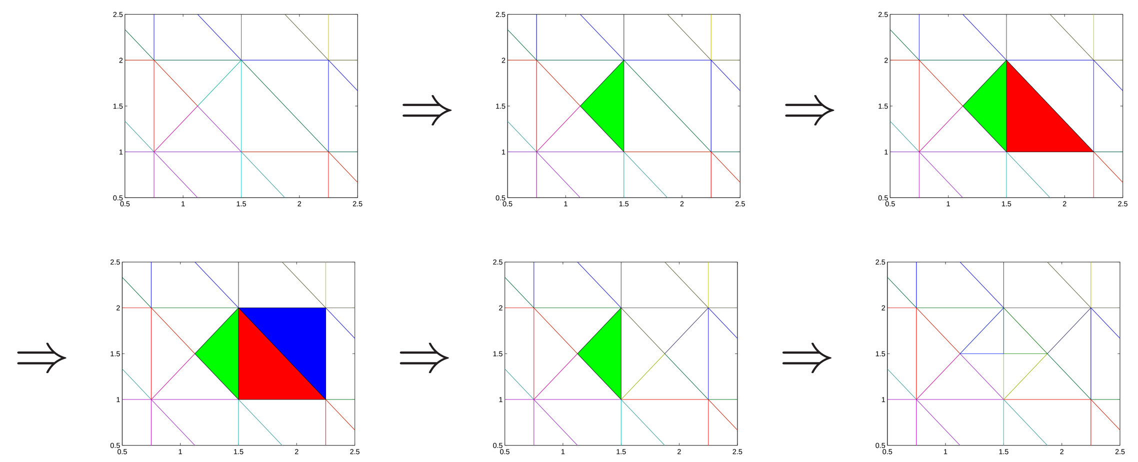
$$\Delta\mathrm{err}_i = \left| \mathrm{vol}(\hat{u}(e_i)) - \sum_{k=1}^{N} \mathrm{vol}(\hat{u}(e_i^{(k)})) \right|$$

Elements where $\Delta\mathrm{err}_i > \texttt{tol}$ are then marked.

Method

Short overview of the AMR algorithm:
1) Create initial mesh
2) Solve BVP with FEM for initial mesh
3) Refine mesh uniformly and mark all elements
4) While there are marked elements
   - Solve BVP w. FEM for refined mesh
   - Estimate change in error
   - Mark elements where error > tolerance
   - Locally refine marked elements

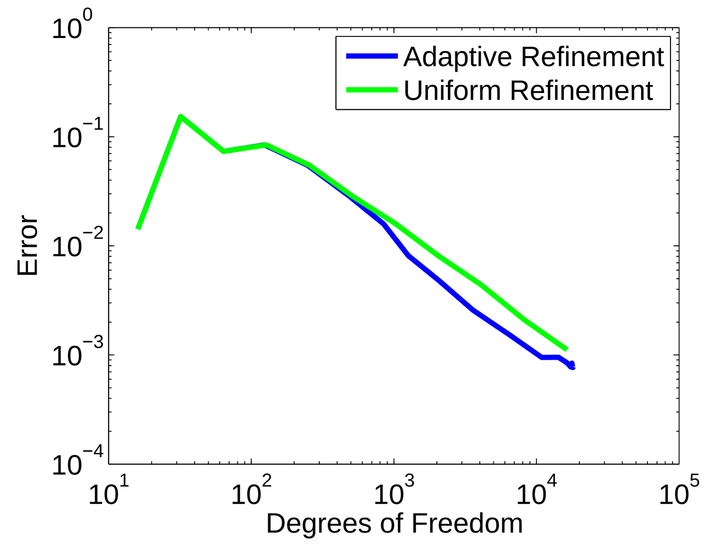
Refining Elements



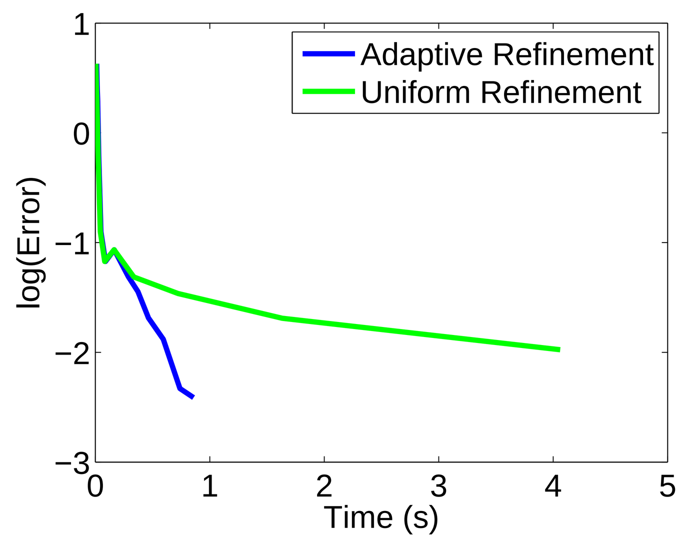
To preserve order in the meshgrid, we have constructed the algorithm so that an edge in an element never has two neighbouring elements. In the figures we see how we do a local refinement of the green element. To preserve the order this requires the red element to be refined, which in turn requires the blue element to be refined. This is done recursively so that the original green element is refined last and finally a local refinement is completed, and no edge neighbours two elements.

Performance

DOF vs. true error

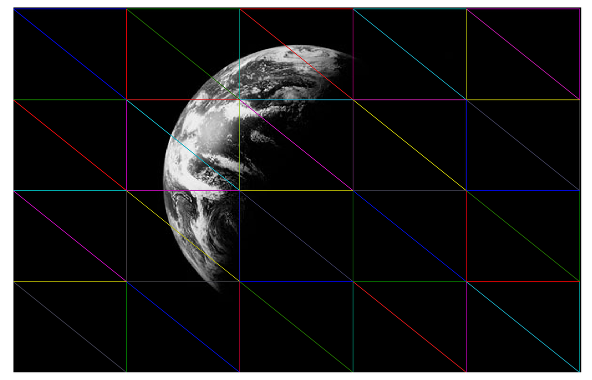


Time vs. true error

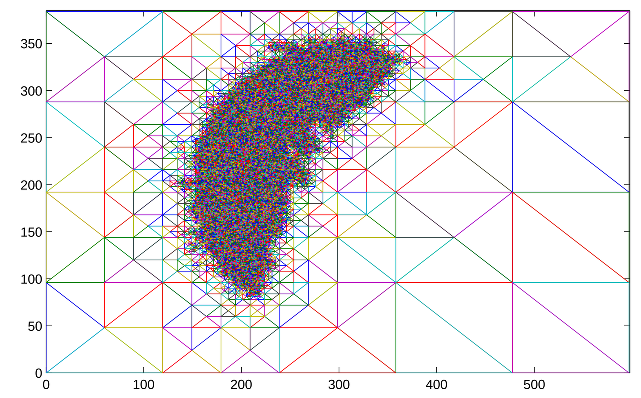


In the figures we see how the adaptive algorithm compares to a uniform refinement algorithm. We see that the two algorithms do nearly the same until a certain point at which the adaptive algorithm starts outperforming the uniform refinement, in respect to both time spent and degrees of freedom used. This is due to our very coarse starting mesh from which the AMR actually has to refine every single element. The difference between the two algorithms happens when the AMR starts only refining specific elements.

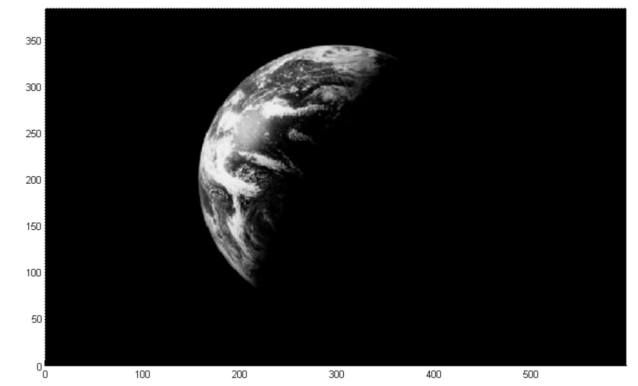
Image compression using AMR



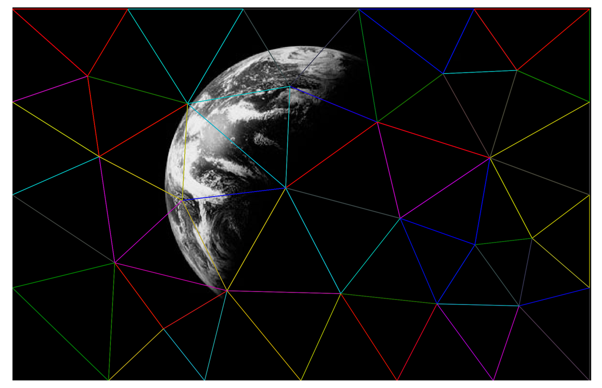
Initial structured mesh



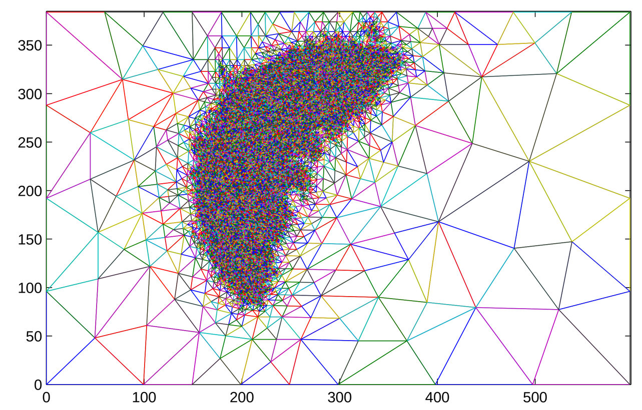
AMR-Mesh - 19278 nodes



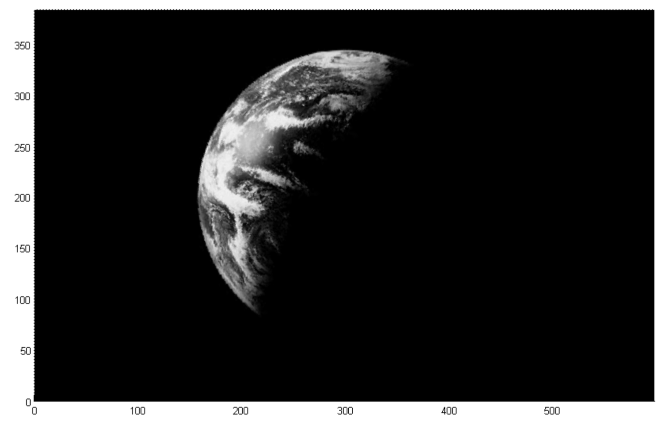
Approximation



Initial non-structured mesh



AMR-mesh - 22258 nodes



Approximation

Applications of FEM are not restricted to solving partial differential equations. The AMR-algorithm has here been used to compress a grayscale image. The above image with $385 \cdot 598 = 230230$ pixels has been compressed using the adaptive FEM. For the structured mesh this gives a compression of 74.88% and for the non-structured mesh 71%. The algorithm creates a fine mesh in areas with large changes in pixel values and a coarse mesh in areas with small changes as seen above.

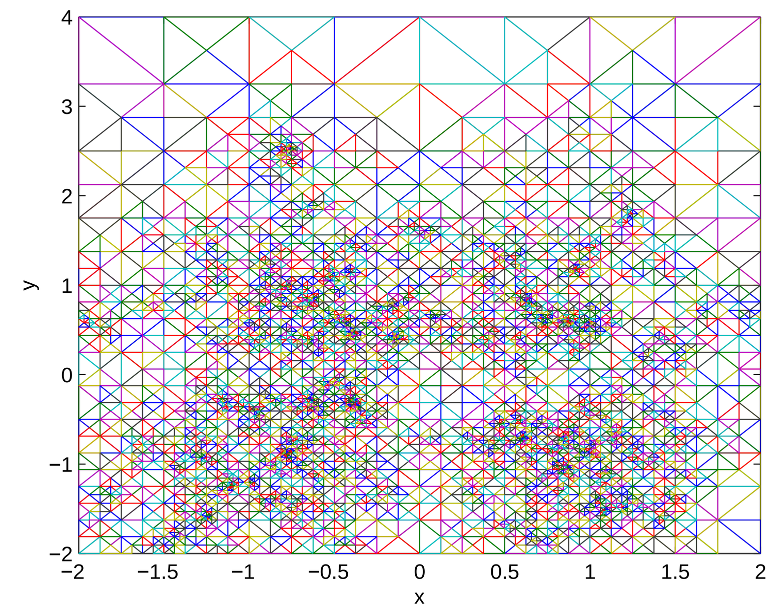
Solving PDE's using AMR and FEM

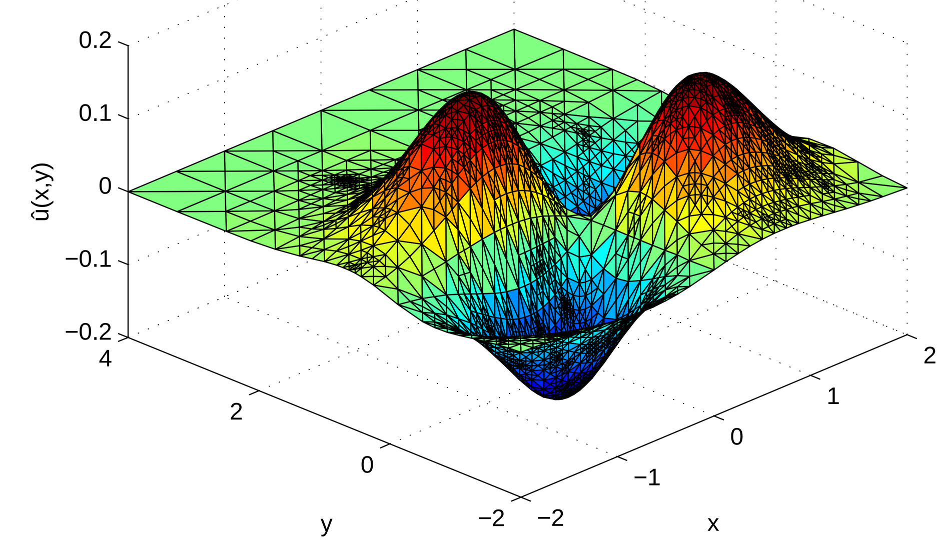
PDE problem with Dirichlet Boundary condition

$$\frac{\partial^2}{\partial x^2}u(x,y) + \frac{\partial^2}{\partial y^2}u(x,y) = 4xye^{-x^2-y^2}(y^2+x^2-3) \qquad (x,y) \in \Omega$$

$$u(x,y) = -xye^{-x^2-y^2} \qquad (x,y) \in \Gamma$$

Approximation of $u(x,y)$ using an adaptive mesh grid for the Finite Element Method.





Areas with high curvature gets more refined than areas with lower curvature.
This saves storage space and computation time compared to uniform refinement methods.