CARE: Computation for Advanced Reactor Engineering

UNIVERSITY OF LEEDS · Newcastle University · Imperial College London · MANCHESTER 1824 · Computation for Advanced Reactor Engineering · EPSRC Engineering and Physical Sciences Research Council

Thread-Parallel Anisotropic Mesh Optimisation Using PRAgMaTIc

Georgios Rokos
Software Performance Optimisation Group,
Department of Computing,
Imperial College London
`georgios.rokos09@imperial.ac.uk`
`http://www.doc.ic.ac.uk/~gr409/`

Gerard Gorman
Applied Modelling and Computation Group,
Department of Earth Science and Engineering,
Imperial College London
`g.gorman@imperial.ac.uk`
`http://www3.imperial.ac.uk/people/g.gorman/`

Introduction

The numerical methods used to model complex geometries required by many scientific applications (e.g. climate prediction, simulation of cardiac electrophysiology) often favour the use of unstructured meshes and finite element discretisation methods over structured grid alternatives. This flexibility introduces complications, such as the management of mesh quality and additional computational overheads arising from indirect addressing. Mesh adaptivity methods ([2, 3, 4]) provide an important means to control solution error by focusing mesh resolution in regions of the computational domain where it is required.
PRAgMaTIc (Parallel Anisotropic Adaptive Mesh Toolkit) is an open-source 2D/3D mesh adaptivity framework, built with large-scale multiprocessing in mind. It is the first hybrid OpenMP/MPI implementation of anisotropic mesh adaptivity, based on earlier work by Freitag *et al.* [3]. Although mesh adaptivity has been successfully parallelised using MPI by a number of groups, thread-level parallelism is significantly more challenging because each thread modifies mesh data and therefore operations must be carefully marshalled to avoid data races while maintaining good load-balance and ensuring enough parallelism is exposed to achieve good parallel efficiency. Currently, the main focus of our work is on threaded (OpenMP) 2D mesh adaptivity [5]. PRAgMaTIc can be downloaded from Launchpad: `https://launchpad.net/pragmatic`.

Adaptive Algorithms

Coarsening

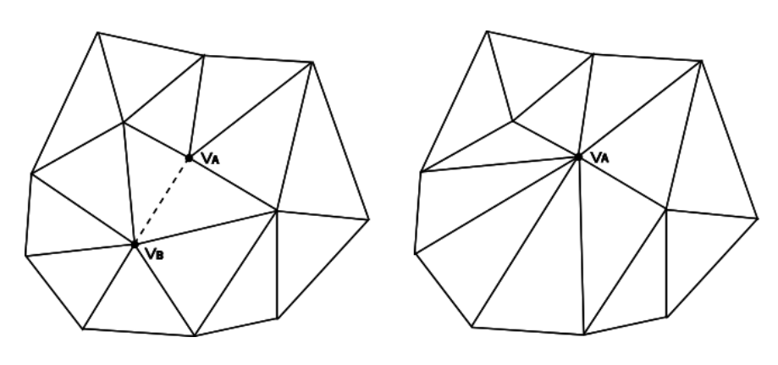

Fig. 2: Edge collapse: vertex V_B collapses onto V_A, removing the dashed edge and the corresponding elements from the mesh.

Swapping

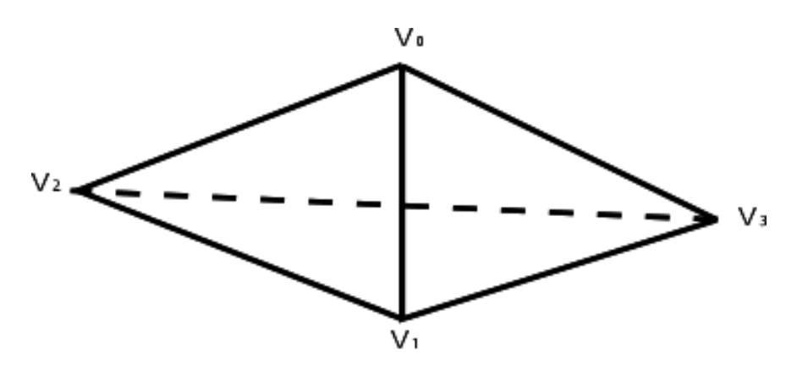

Fig. 3: Edge swapping: edges shared between two elements can be flipped if doing so produces elements of higher quality than the original ones.

Refinement

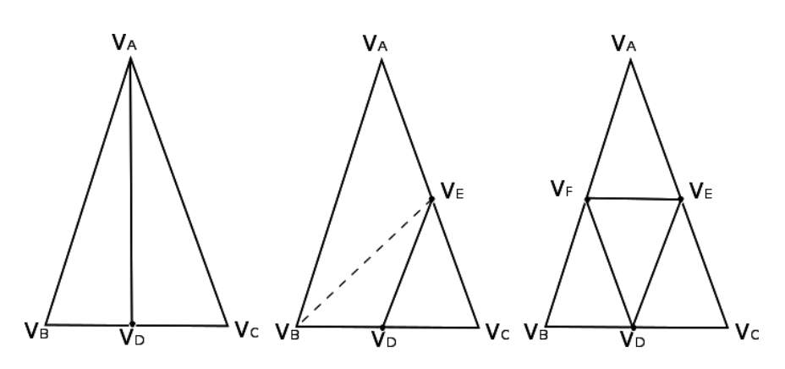

Fig. 4: Edge refinement: edges are split, leading to 1:2 (bisection), 1:3 or 1:4 (regular refinement) division of elements, which increases local mesh resolution.

Smoothing

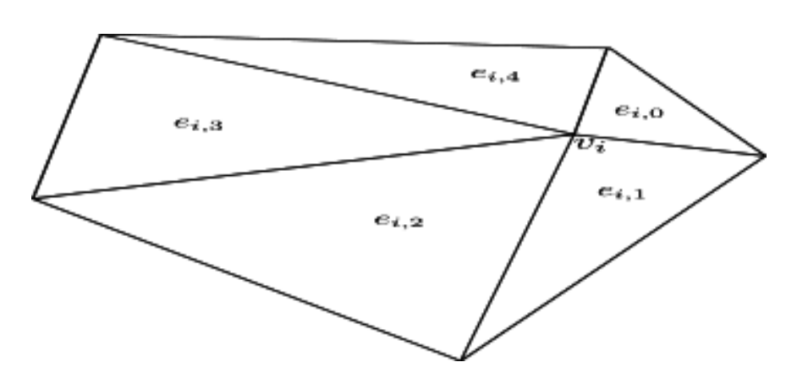

Fig. 5: Optimisation-based vertex smoothing: a vertex u_i is relocated to a new position so that the quality of the worst element among $\{e_{i,0}..e_{i,5}\}$ is maximised.

A heuristic method of improving mesh quality is the iterative application of the four adaptive algorithms until the desired minimum quality is achieved. Mesh adaptation is driven by a posteriori error estimations, encoded in the form of a metric tensor field which is discretised node-wise. In essence, the metric tensor at each node shows how the euclidean space has been distorted (amount and direction of distortion) at that point. The ideal mesh is the one in which all elements are equilateral with edges of unit length in metric space.

Under the Hood

Mesh Data: The mesh is represented using a 1D Element-Node adjacency list, a 2D Node-Node adjacency list and a 2D Node-Element adjacency list, alonside 1D arrays in which we store node coordinates and metric tensor values.
Structural Hazards: When running adaptivity in parallel, unsafe operations which can result in invalid mesh elements are avoided by using a fast graph colouring technique (an improved version of [1]) and applying the algorithms to vertices in the maximal independent set.
Deferred Updates: Thread-safe updates to adjacency lists are guaranteed by deferring the updates until the maximal independent set has been processed. All pending operations are distributed to threads, with all operations pertaining to a specific vertex being applied by one and only one thread.
Worklists: Colouring only the subset of the mesh which needs to be adapted has been shown to improve performance significantly. Additionally, adaptive operations on a local mesh patch need to be propageted to the wider neighbourhood. These two requirements suggest the use of worklists. Efficient parallel manipulation of worklists is achieved using `atomic captures`, an operation introduced in OpenMP 3.1 to implement atomic fetch-and-add.

Sample Benchmark

A synthetic solution ψ is defined to vary in time and space for some value of the period T:

$$\psi(x,y,t) = 0.1\sin(50x + 2\pi t/T) + \arctan(-0.1/(2x - \sin(5y + 2\pi t/T)))$$

Sample benchmark was run for $t \in [0, 51)$ in increments of unity. The code was compiled with Intel$^{®}$Compiler Suite 14.0.1 and with the `-Ofast` optimisation flag. The benchmark was executed on a dual-socket Xeon$^{®}$E5-2650 system using Intel's thread-core affinity support.

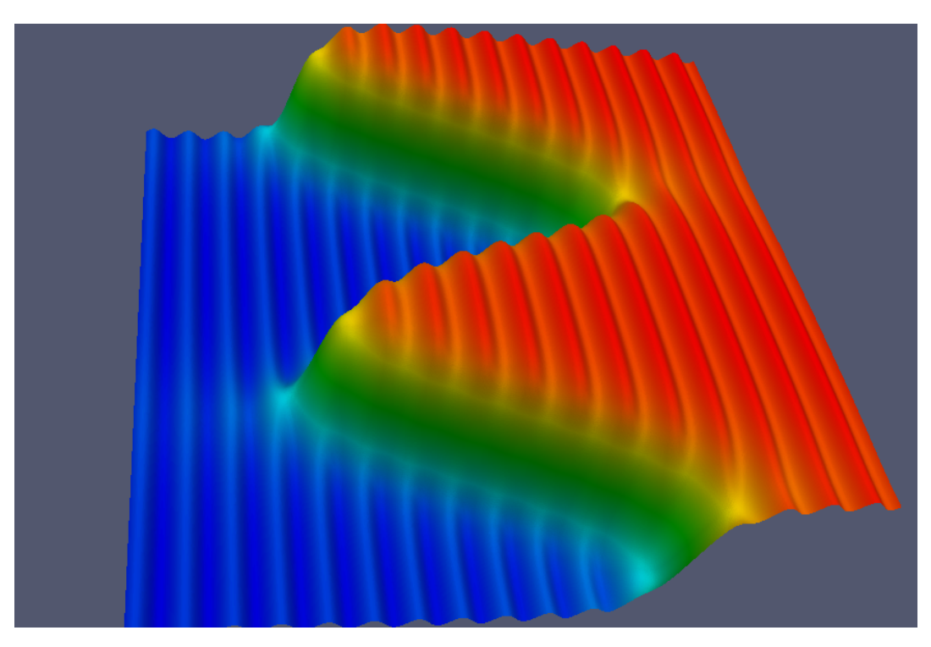

Fig. 6: Benchmark solution field

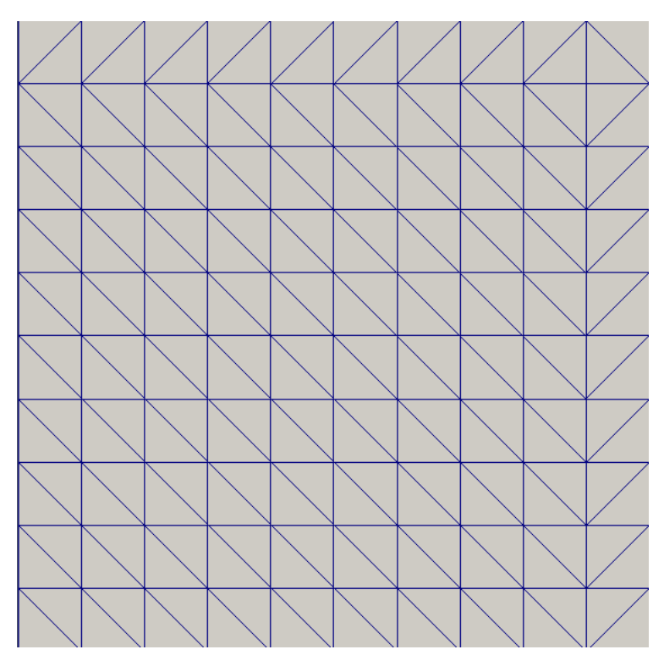

Fig. 7: Initial auto-generated mesh

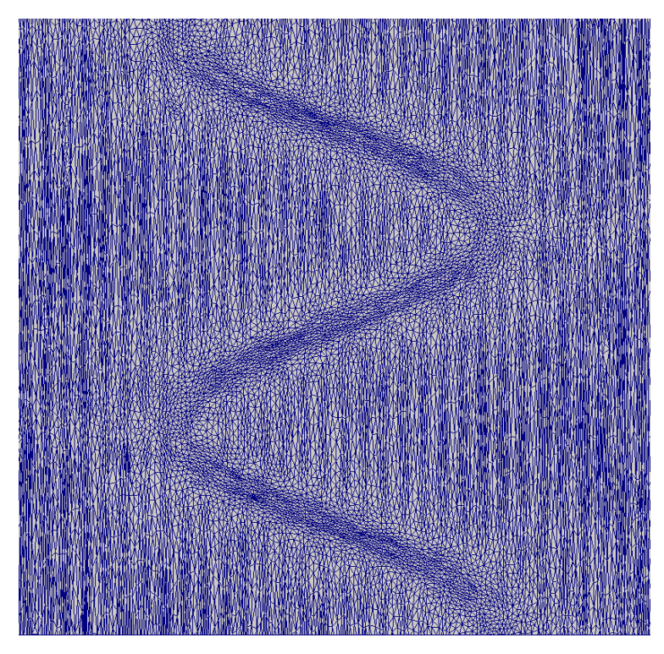

Fig. 8: Adapted mesh for some t

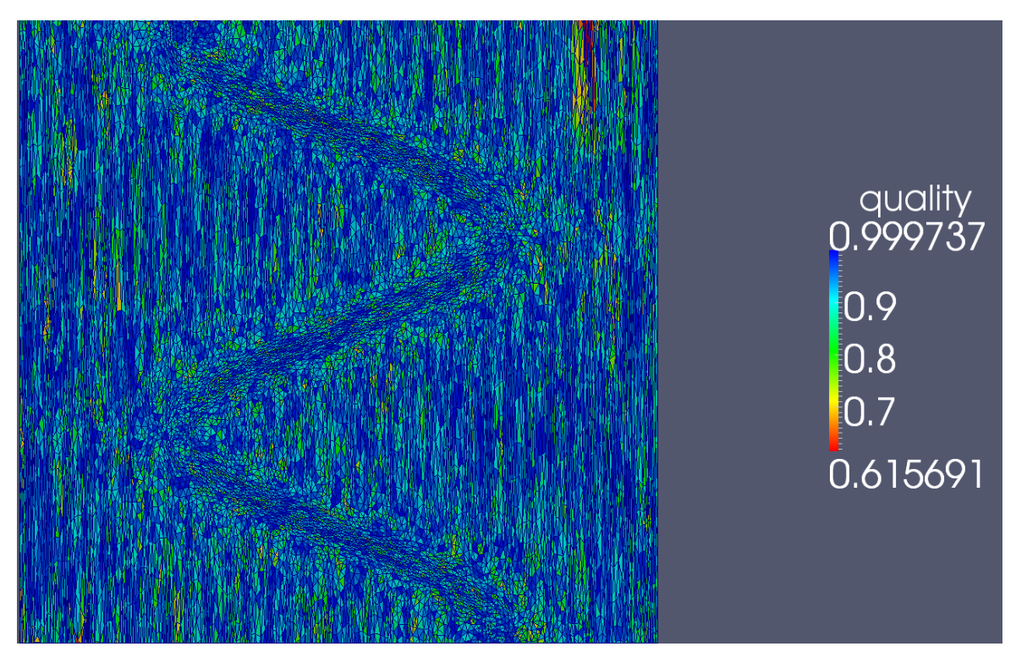

Fig. 9: Quality of adapted mesh

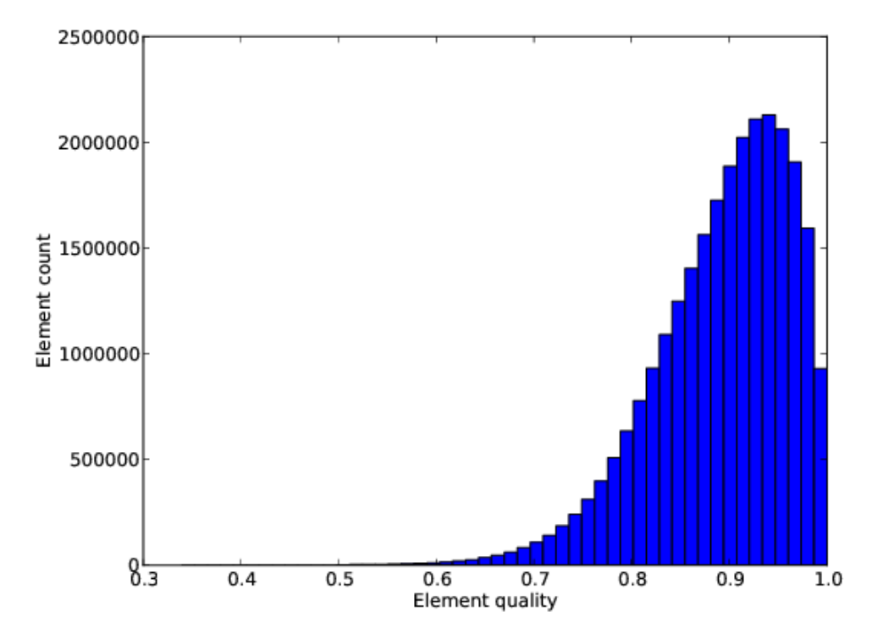

Fig. 10: Aggregated histogram of element qualities

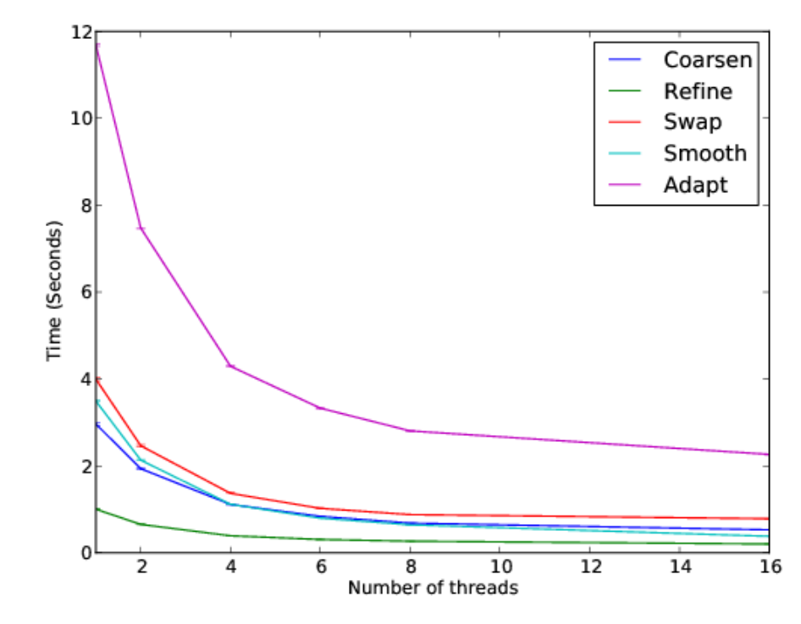

Fig. 11: Total wall time per phase

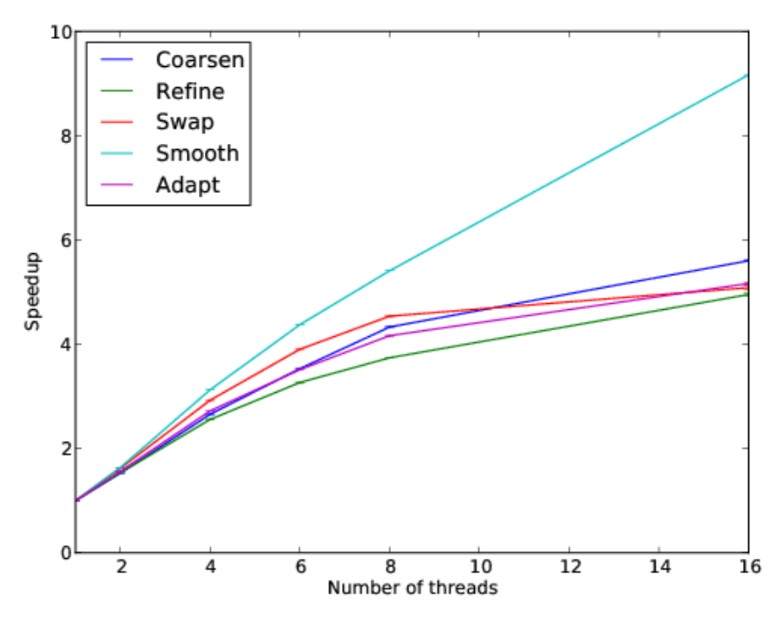

Fig. 12: Speedup per phase

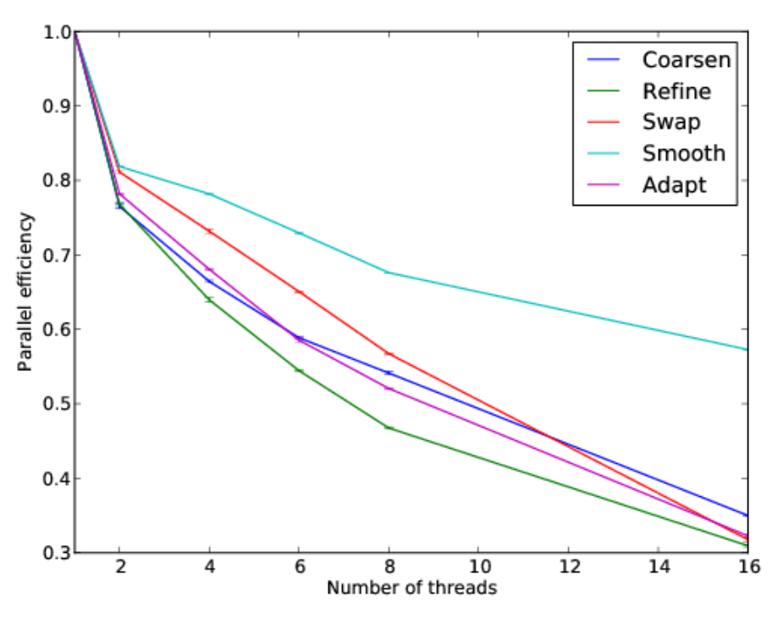

Fig. 13: Parallel efficiency per phase

Conclusions

PRAgMaTIc produces high-quality adapted meshes from typical, auto-generated input meshes. Average element quality is > 0.9, whereas worst element quality is > 0.6 (ideal quality is 1.0). Performance and scalability are high, given the inherent difficulty of parallelising complex, unstructured algorithms which constantly modify mesh topology, therefore rendering classic parallel techniques (Jones-Plassmann colouring, mesh partitioning etc.) inappropriate for such applications. Work on further optimisations like custom for-loop scheduling and 2-stage thread barriers are the subject of current research and preliminary results are very encouraging.

References and Acknowledgements

[1] ı. V. ÇAtalyüRek, J. Feo, A. H. Gebremedhin, M. Halappanavar, and A. Pothen. Graph coloring algorithms for multi-core and massively multithreaded architectures. *Parallel Comput.*, 38(10-11):576–594, Oct. 2012.

[2] L. Freitag, M. Jones, and P. Plassmann. An efficient parallel algorithm for mesh smoothing. In *Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories*, pages 47–58. Citeseer, 1995.

[3] L. F. Freitag, M. T. Jones, and P. E. Plassmann. The Scalability Of Mesh Improvement Algorithms. In *IMA VOLUMES IN MATHEMATICS AND ITS APPLICATIONS*, pages 185–212. Springer-Verlag, 1998.

[4] X. Li, M. Shephard, and M. Beall. 3d anisotropic mesh adaptation by mesh modification. *Computer methods in applied mechanics and engineering*, 194(48-49):4915–4950, 2005.

[5] G. Rokos, G. J. Gorman, J. Southern, and P. H. J. Kelly. A thread-parallel algorithm for anisotropic mesh adaptation. *CoRR*, abs/1308.2480, 2013 (Under review, `http://arxiv.org/abs/1308.2480`).

We would like to thank Fujitsu Laboratories of Europe Ltd and EPSRC grant number EP/I00677X/1 for supporting this work.