Efficiency considerations in triangular adaptive mesh refinement

BY JÖRN BEHRENS¹,* AND MICHAEL BADER²

¹Alfred Wegener Institute for Polar and Marine Research, Am Handelshafen 12, 27570 Bremerhaven, Germany
²Technische Universität München, Boltzmannstrasse 3, 85748 Garching, Germany

Locally or adaptively refined meshes have been successfully applied to simulation applications involving multi-scale phenomena in the geosciences. In particular, for situations with complex geometries or domain boundaries, meshes with triangular or tetrahedral cells demonstrate their superior ability to accurately represent relevant realistic features.

On the other hand, these methods require more complex data structures and are therefore less easily implemented, maintained and optimized. Acceptance in the Earth-system modelling community is still low. One of the major drawbacks is posed by indirect addressing due to unstructured or dynamically changing data structures and correspondingly lower efficiency of the related computations.

In this paper, we will derive several strategies to circumvent the mentioned efficiency constraint. In particular, we will apply recent computational sciences methods in combination with results of classical mathematics (space-filling curves) in order to linearize the complex data and access structure.

Keywords: local triangular mesh refinement; computational efficiency; error estimation; space-filling curve; Sierpinski curve

1. Introduction

Adaptive local mesh refinement (AMR) has proved very helpful in accurately representing multi-scale phenomena numerically. Examples include hurricane-track simulations (Gopalakrishnan et al. 2002), two-dimensional and three-dimensional trace-gas transport simulations (Blayo et al. 1999; Behrens et al. 2000; Hubbard & Nikiforakis 2003), ocean circulation (Behrens 1998; Pain et al. 2005), tsunami wave propagation (George & LeVeque 2006; Pranowo et al. 2008; Shaw et al. 2008) etc. In order to apply AMR successfully, several independent methods need to be combined.

*Author and address for correspondence: Numerical Methods in the Geosciences (Klima Campus), University of Hamburg, Grindelberg 5, 20144 Hamburg, Germany (joern.behrens@awi.de).

One contribution of 10 to a Theme Issue 'Mesh generation and mesh adaptation for large-scale Earth-system modelling'.
(i) An efficient refinement strategy leads to locally and highly dynamically refined but consistent meshes.

(ii) A refinement criterion needs to capture the phenomenon of interest efficiently during run-time.

(iii) Data structures need to support the dynamic hierarchical characteristic of the mesh refinement as well as the linear data access pattern preferred by numerical computations.

In this paper, we focus on the computational efficiency issues. Some introductory remarks will be given on optimal refinement strategies as well as on efficient and reliable refinement criteria in the first subsection below.

(a) Mathematically efficient refinement

In order to achieve mathematical efficiency in dynamic local refinement, two distinct steps need to be considered:

(i) the refinement strategy, i.e. the way in which a criterion for refinement is applied to the mesh;

(ii) the refinement criterion, i.e. the mathematical estimation of error or insufficient resolution.

A very common adaptation strategy is the equi-distribution strategy. A global error $\varepsilon$ is derived by a $p$-norm approach from the vector of local error estimates $\varepsilon_m = (\varepsilon_{\tau_k}), \; k = 1, \ldots, m$, on each element $\tau_k$,

$$
\varepsilon = \|\varepsilon_m\|_p = \left( \sum_{k=1}^{m} \varepsilon_{\tau_k}^p \right)^{1/p}.
$$

The equi-distribution strategy tries to achieve a uniform distribution of local error. In other words, $\varepsilon_{\tau_k} \approx \bar{\varepsilon}$ everywhere. This can be attained by refining those elements $\tau_k$ for which

$$
\varepsilon_{\tau_k} > \frac{\varepsilon}{m^{1/p}}.
$$

A provably optimal adaptation strategy has been studied by Dörfler (1996) in the context of elliptic PDE solutions. Note that, in an a posteriori sense, these methods are also relevant for non-elliptic PDEs. It should further be noted that hyperbolic equations might require additional shock-capturing-based refinement strategies. The idea is to refine a subset of mesh elements, with the sum of their local errors being a fixed part of the total error $\varepsilon$. Then, let $0 < \theta < 1$ be a given parameter; find a minimal set $S = \{\tau_1, \ldots, \tau_k\} \subseteq T$ of the triangulation $T$, such that

$$
\sum_{\tau \in S} \varepsilon_{\tau}^p \geq (1 - \theta)^p \varepsilon^p. \tag{1.1}
$$

If $S$ is being refined, the error will be reduced by a factor depending on $\theta$ and the properties of the problem’s data. Dörfler derives a selection strategy for the elements in $S$ by an inner iteration.
It remains to define $\varepsilon_{\Delta}$. Many applications in geophysical fluid flow apply gradient-based refinement criteria. These criteria are easy to interpret and mostly straightforward to implement. However, they are often not efficient nor reliable. By efficient, we mean that the (unknown) true discretization error is not overestimated; in other words, the refinement area is as small as possible. A reliable criterion does not underestimate the true error, which means that the refinement area is as large as necessary.

From our experience, a simple to derive, yet (for elliptic problems) provably reliable and efficient category of refinement criteria is based on averaging (Carstensen 2004). Let us assume Poisson’s equation

$$-\Delta \rho = f \text{ in } G \subset \mathbb{R}^d, \quad (1.2)$$

where $\Delta$ is the Laplace operator and $f$ a right-hand side in a closed domain $G$. Further, we assume suitably defined boundary and initial conditions on the (unknown) solution $\rho$. Let $\rho_h$ be a numerically computed solution to the discrete analogue of equation (1.2). In order to estimate the true error $\varepsilon = |\Delta \rho - \Delta_h \rho_h|$, $\Delta_h$ being the discrete Laplace operator, we try to approximate $\Delta \rho$ by applying the following scheme for the gradient operator twice:

$$\sigma = \sum_{\{i: \tau_i \text{ in patch of node } i\}} |\tau_i| \cdot \nabla \rho_h|_{\tau_i}. \quad (1.3)$$

Here, $|\tau|$ denotes the area of $\tau$. Extend $\sigma$ to the whole element $\tau$ by linear interpolation. The error estimator is derived from $|\sigma^{(2)} - \Delta_h \rho_h|$. This approach was first proposed by Zienkiewicz & Zhu (1987) and is about the first mathematical error estimator for adaptive finite-element computations.

Note that the derived refinement strategies are usually also applied for coarsening the mesh. Instead of considering an error $\varepsilon_{\Delta} > \text{TOL}$ for refinement, we now consider an error $\varepsilon_{\Delta} < \text{tol}$ for coarsening, where usually tol $< \text{TOL}$. This means that elements with a small error are considered over-refined and can be coarsened.

Now that we have derived refinement strategies and refinement criteria, for handling local or multi-scale features of a geophysical problem efficiently, we need to refine the mesh efficiently.

\((b)\) Geometry of local mesh refinement

For simplicity, we will consider only two-dimensional meshes here. For refinement strategies in three-dimensional space, the reader is referred to Behrens (2006). In order to efficiently refine and de-refine a mesh, hierarchical methods have proved their usefulness. We present two methods here.

Triangular bisection of a marked edge is depicted in figure 1. Each element is equipped with a marked edge. If an element is considered for refinement, only this edge can be subdivided. An initially marked edge is then bisected by a new node. A new edge, connecting the new node with the corresponding opposite vertex, is created and two new triangles, the daughter triangles, are defined. The edges opposite to the new node become the marked edges of the daughter elements. This procedure is defined recursively. Furthermore, it can be embedded in an algorithm for avoiding hanging nodes (yielding a conforming mesh). This algorithm is extremely simple as it states that the refinement is to be repeated.
for all those elements that either are considered for refinement from the error estimation procedure or contain hanging nodes. The algorithm terminates if none of these are left for refinement. Bänsch (1991) proves that this algorithm always terminates and leads to a conforming triangulation. Coarsening is also possible. Under the assumption that only so-called admissible patches are considered, the refinement can be reversed.

A quadrilateral refinement procedure with the same simplicity is hard to find. The simplest geometrical refinement method for quadrilaterals (rectangles) is given in figure 2. While the specification to refine one quadrilateral is still simple, attaining a conforming mesh is not an easy task. We refer to the literature here (e.g. Behrens 2006).

Several other methods to refine cells are common and used. Intersecting a triangle by bisecting each edge is called regular triangular refinement. Very commonly, quadrilaterals are subdivided into nine or more daughter elements, and even completely free simplicial meshes built from polygonal cells are used (e.g. Weller & Weller 2008).

(c) Data structures and the problem of efficiency

Locally adaptive meshes require involved data structures. While hierarchical mesh refinement strategies are best mapped to tree-like data structures (figure 3), numerical computations are not necessarily efficient on these. In fact, most floating-point units on current high-performance processors require data streams in consecutive arrays.

In order to efficiently manage refinement and de-refinement of hierarchically generated locally refined meshes, tree data structures are natural. In figure 3, two small meshes with local refinement are presented, together with their tree representation. Each cell is assigned a tree node. If the element is refined, it
is the root of a subtree. The corresponding tree would be stored in an object-oriented linked manner. In order to do computations, an algorithm would have to traverse the tree.

On the other hand, numerical computations require consecutive storage access. This is evident for vectorized or pipelined processing units. But even with cache-based memory designs, consecutive data access patterns prove to be beneficial because each time data are accessed, a whole (consecutive) cache line is loaded.

The consequence is a dilemma: either the mesh management part or the numerical computational part of an adaptive algorithm can be optimized. Most current adaptive codes solve the problem by optimizing the numerical part and allow mesh adaptation only in larger intervals, using sophisticated predictions of mesh changes (e.g. Power et al. 2006). Another approach is taken in the author’s work, where the two phases—grid manipulation and numerical computation—are clearly separated and a gather/scatter step is performed in between these steps (Behrens et al. 2005). The principle is illustrated in figure 4. For grid management (left side), object- and tree-oriented data structures allow for optimized mesh handling. In the computation phase (right side), vector-like data structures allow for high throughput in current floating-point processing units. A gather and scatter step maps from one data structure to the other. It is important to note that a requirement for this principle to work is that neighbourhood relations in the grid (which usually dominate the numerical computations) are preserved in the gathered vector-like data structure. In many applications of the mesh generator amatós that implement this paradigm, the overhead imposed by the gather/scatter step is below 1 per cent of the computation time.

It turns out that this paradigm helps to gain some efficiency and also helps in parallelizing the dynamically adaptive code in a straightforward manner. Note, however, that true efficiency can only be achieved when data access patterns preserve neighbourhood relations. This will be elaborated in §2, after introducing an example of an adaptive tsunami wave propagation code.
\( \frac{\partial t}{v} + (v \cdot \nabla) v + f \times v + g \nabla \zeta + \frac{n^2 |v|}{H^{1/3}} - \nabla \cdot (A_h \nabla v) = 0 \) \hfill (1.3)

and

\( \frac{\partial t}{\zeta} + \nabla \cdot (v H) = 0, \) \hfill (1.4)

where \( v = (u(t, r), v(t, r)) \) denotes the horizontal depth-averaged velocity and \( H = h(r) + \zeta(t, r) \) denotes the total water depth, with \( \zeta \) the time-dependent sea surface elevation relative to the mean water depth \( h \); \( f \) is the Coriolis parameter, \( g \) the gravitational constant, \( n \) the Manning roughness and \( A_h \) a viscosity coefficient.

---

Phil. Trans. R. Soc. A (2009)
These equations are discretized by a conforming/non-conforming linear finite-element method introduced in Hanert et al. (2005). Modifications are applied to the advection operator and an explicit leap-frog time-stepping scheme is applied. Details are described in Androsov et al. (2008).

Currently, automatic mesh generation for complex ocean basins is being developed. More accurate boundary fitting elements are achieved by optimization of the node placements. Inundation is being tested. The (global) time-step length is automatically controlled utilizing an estimated Courant number criterion. More precisely, the time step $\Delta t$ is set to

$$\Delta t = \theta \frac{g H_\tau}{h_\tau},$$

with $\theta < 1$ a user-defined threshold, $H_\tau$ the mean water depth in element $\tau$ and $h_\tau$ the width of $\tau$. However, a big challenge is still to find an adequate refinement strategy and criteria for the adaptive mesh size control. Several other examples of adaptive mesh applications are given in Behrens (2006).
2. Efficient tree-based algorithm

We will now take a closer look at the methods to gain higher efficiency. It will be the key topic of this section to derive a cache-aware ordering and access scheme for general adaptively refined meshes. Additionally, we will propose a new radical way to overcome large overheads imposed by extensive data management requirements in dynamically refined meshes.

(a) Refinement tree to bit stream

The first observation we will use states that a mesh can be represented as a tree and that this tree can then be represented by a bit stream. This is illustrated in figure 6. The triangulation on the left is represented by the tree to the right. Each node of the tree is assigned a bit:

$$\text{bit}_{\text{node}} = \begin{cases} 0, & \text{if node is a leaf}, \\ 1, & \text{else}. \end{cases}$$

The corresponding bits are indicated in the figure. Now, traversing the tree (by convention) in a depth-first sequence defines the following bit pattern:

1110100111000100 . . .

Note that the bit pattern is uniquely defined by the node’s bit assignments once the traversal sequence is established. Therefore, knowing the coordinate of the original triangle, the geometrical refinement method and the bit pattern allow for a complete representation of the mesh. This is a radical simplification compared with the current implementation of mesh-generation tools like amatos (Behrens et al. 2005).

(b) Neighbourhood relations—space-filling curves

In figure 6, it is noted that the access sequence corresponding to depth-first traversal corresponds to the Sierpinski curve, a space-filling curve (SFC). SFCs have very interesting properties: they map a multi-dimensional domain to one-dimensional curves. For the given types of recursively refined meshes, there exist
SFCs that map the mesh to a (consecutive) one-dimensional sequence of elements. It can be shown that neighbourhood relations in the mesh are preserved to a large extent on the curve. In other words, if elements are neighbours in the mesh, they are also most likely to be close to each other in the consecutive sequence of the SFC. The neighbourhood preservation establishes our second important observation.

This can be utilized with great benefit for efficient load balancing and partitioning (Behrens & Zimmermann 2000; Mitchell 2007). Not only does domain partitioning work well, but matrix structure can be optimized and cache utilization maximized by this reordering scheme (Behrens 2005).

Having stated that the neighbourhood relations are preserved under the depth-first traversal of the refinement tree, and after observing the correspondence to an SFC, we can now see that the tree traversal also establishes a mesh traversal. In this mesh traversal, a new mesh element is always accessed from an already touched one. Thus, for computations, we always step forward from known elements to unknown ones over common edges.

(c) Tabulated access patterns

The third important observation is that the access patterns to elements as well as vertices corresponding to these elements can be tabulated in a systematic way. As a new element is always accessed through a known one, vertices are either common or new. Depending on the orientation and type of the known element from which the algorithm approaches a new element, a table stores the type (known or unknown) and orientation of the new element and its vertices. From this table, it can also be derived whether a vertex will ever be accessed again or not.

With this information, there are three different types of unknowns/vertices:

(i) those that are new (input stream);
(ii) those that will never be accessed any more (output stream);
(iii) those that are neither new nor finished (intermediate storage).

Details of the tabulation and types are given in Bader et al. (2006). We just state that the intermediate storage can be organized in an optimal way by stack-type intermediate storage. The table of access types also determines the way in which known intermediate elements are retrieved from and stored on the correct stack location. A small number of stacks is necessary in order to maintain optimal access. With this tabulated scheme, one can guarantee that either each data item needed for the computations is a new one that can be accessed from the input stream, or it will be finalized after the current step, or it can be accessed from the top of one of the (tabulated) stacks. So, data access takes place either in a consecutive (pre-fetchable) way from/to input/output streams or in a cacheable way to extremely efficient intermediate storage.

(d) Experiments and results

Preliminary tests have been performed with the above-described algorithm for a stationary problem (Bader et al. 2006, 2008). We solve Poisson’s equation

\[-\Delta \rho = f \text{ in } G \in \mathbb{R}^2, \quad \rho|_{\partial G} = 0,\]

Phil. Trans. R. Soc. A (2009)
Figure 7. Example domain with a re-entrant corner and locally refined triangular mesh.

Table 1. Memory requirements and execution times for different adaptively refined mesh sizes. Adapted from Bader et al. 2008.

<table>
<thead>
<tr>
<th>number of elements</th>
<th>number of unknowns</th>
<th>memory (MB)</th>
<th>run-time per iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>double precision</td>
<td>integer</td>
</tr>
<tr>
<td>357 588</td>
<td>179 439</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>863 296</td>
<td>432 640</td>
<td>56</td>
<td>12</td>
</tr>
<tr>
<td>1 915 368</td>
<td>959 190</td>
<td>96</td>
<td>24</td>
</tr>
<tr>
<td>5 184 010</td>
<td>2 594 456</td>
<td>219</td>
<td>61</td>
</tr>
<tr>
<td>12 902 390</td>
<td>6 455 242</td>
<td>510</td>
<td>222</td>
</tr>
</tbody>
</table>

where $f$ is a given right-hand side and $\partial \mathcal{G}$ is the boundary of the closed domain $\mathcal{G}$, representing a square with a re-entrant corner (figure 7). The solution of the linear finite-element discretization is sought with a hierarchical-basis multi-level pre-conditioning scheme applied to a conjugate gradient iterative method (Griebel 1994).

Tests have been conducted on a personal computer with an Intel Pentium 4 processor (Prescott, 3.4 GHz, 1 MB level-2 cache) and 2 GB of main memory. We consider double-precision computations in the floating-point parts. Results are given in table 1. Note that the memory requirements even for a 12 million element grid fit easily in a common PC’s main memory. This is due to the fact that most of the required memory scales with the number of unknowns. Additional memory for grid management is (almost) abandoned.

At least as important is the run-time behaviour. Computational time per iteration and unknown is almost constant. This indicates that linearization of data access leads to an algorithm that is not slowed when the cache fills.
Pre-fetching can be realized by automatic compiler optimizations. In fact, for the level-2 cache of the Intel architecture, hit rates of over 99 per cent were observed.

These results are extremely convincing, and it is our aim to develop AMR tools that rely on the proposed techniques, yet offer simple interfaces for the application, as amatos does. First time-dependent problems have been solved (linear shallow-water equations) with similar results.

It should be noted that the algorithm allows for vectorization as well as parallelization. As the algorithm proceeds along the lines of depth-first traversal, it induces an SFC that can be used for partitioning. At the same time, the input stream is a vector (of arbitrary length). Some additional intermediate storage locations are needed, which can be located in fast registers or caches. Therefore, the algorithm is well suited for the current computer architectures used in Earth-system modelling.

3. Challenges

Some challenges remain. The efficient algorithm explored in §2d for solving partial differential equations as they occur in dynamical cores of current Earth-system modelling tools on dynamically adaptive meshes is still restricted to certain (and at the moment two-dimensional) types of meshes. The method described in this paper can only be applied to meshes created by hierarchical triangular bisection. A similar method for quadrilateral meshes, refined by edge bisection of the elements, has been proposed in Mehl et al. (2006).

The methods have proved their efficiency only on simple geometries. Currently, an automatic meshing tool for complex ocean basins is being developed, which takes the uninterrupted SFC ordering into consideration. A mesh generated by this generator is shown in figure 8.

Figure 8. Complex geometry of the Indian Ocean represented in an automatically generated locally refined mesh (detail).
Furthermore, the methods presented have only been applied to low-order finite elements. It remains to be demonstrated how these methods can be extended to high-order elements or to finite-volume-type computations. We see no major obstacle to this extension. However, simple interfaces and flexible extensible libraries still need to be developed.

The authors would like to gratefully acknowledge contributions by their (former) students Stefanie Schraufstetter, Csaba A. Vigh (both TU München), Widodo S. Pranowo (United Nations University, Bonn, and Alfred Wegener Institute, Bremerhaven), Oliver Kunst, Jan Schlicht and Corinna Ziemer (all three University of Bremen). Parts of this work have been funded by the German Ministry for Research and Education (BMBF) under grant no. 03TSU01.

References


Behrens, J. 2006 Adaptive atmospheric modeling—key techniques in grid generation, data structures, and numerical operations with applications. Lecture Notes in Computational Science and Engineering, no. 54. Berlin: Springer.


Carstensen, C. 2004 All first-order averaging techniques for a posteriori finite element error control on unstructured grids are efficient and reliable. Math. Comput. 73, 1153–1165. (doi:10.1090/S0025-5718-03-01580-1)


