Adaptive Mesh Refinement using Subdivision of Unstructured Elements for Conservation Laws[1]

Daniel B. Vollmer

1st September 2003

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

[1]Submitted to the Department of Mathematics, University of Reading, in partial fulfilment of the requirements for the Degree of Master of Science.

Abstract

An adaptive method based on recursive subdivision of unstructured elements for the solution of conservation laws is presented. The refinement of cells is based on regular subdivision into four children as indicated by a gradient-detector and is carried out “Just-In-Time” before the actual computation on that element takes place. In addition to spatial refinement, temporal refinement is carried out in conjunction with “lock-step” time-stepping to guarantee the availability of the proper states at the correct times across the mesh on all scales. The approach uses standard slope-limited finite volume methods of MUSCL-Hancock type with slight modifications to cater for different levels of subdivision in adjacent elements. We present some background to AMR and the finite volume framework, the algorithm itself and conclude with numerical examples of linear and non-linear scalar conservation laws in two dimensions.

Contents

List of Figures

Chapter 1

Introduction

This paper describes an adaptive finite volume scheme for computing solutions to hyperbolic conservation laws on unstructured (triangular) meshes in two dimensions. Solutions to this class of problems have a number of features which distinguish them from other types of partial differential equations (PDEs) — for example elliptic problems. Hyperbolic PDEs often feature shocks and other forms of non-stationary discontinuities in some parts of their domain, whereas other regions are very smooth and easily discretised.

consequentially Lou *et al.*'s [13]) somewhere in the middle.

By contrast, the method we describe leans very much towards the dynamic end of the imaginary scale and even modifies the mesh geometry during the computational phase. Thus, the distinction between adaption / refinement and traditional computation is no longer given — we refer to this refinement strategy as "Just-In-Time"[1]. Furthermore, the whole domain is (at least conceptually) covered by a single mesh instead of multiple overlaid meshes at di□ering resolutions.

Chapter 2

Background

2.1 Adaptive Mesh Refinement

2.1.1 Philosophy

Recently, methods incorporating adaptive mesh refinement (AMR) have become recognised as a reliable and e□cient means to compute numerical solutions to large-scale problems involving (non-linear) hyperbolic partial differential equations. The main idea behind AMR is to concentrate the computational e□ort where it is needed most — e.g. using a high resolution near "interesting" features of the solution and a lower resolution for its smooth regions.

This is somewhat analogous to using mesh generation for steady-state problems (most often in conjunction with finite elements). The better the mesh is adapted to the geometry and features of the problem, the more

2.1.2 Implementations

Most implementations of adaptive mesh refinement (e.g. [6] or [18]) are based on the very general approach developed by Berger and Oliger [4]. Their algorithm works on recursively embedded regular cartesian grids, although some restrictions apply as to how this embedding can be done; usually rotated grids with half the step-size are used.

The outline of their algorithm is as follows (see Figure 2.1 for an illustration):

1. Calculate the solution on the current grid.
2. Estimate the error for all points in the current grid and mark the ones that exceed a predefined threshold. Also check that the resolution of the current mesh is still needed.
3. Cluster the flagged points into new grids such that the unnecessarily refined area is kept to a minimum — but without creating too many

2.1.3 Consequences

From the previous description of the algorithm, we can make a out a few problematic consequences of those above steps, which will be relevant in the later comparison of our approach.

2.1.3.1 “Buffer Zones”

Unless error-estimation and the subsequent refinement are carried out *every* time-step on *every* grid, the clustering algorithm needs to leave a “buffer zone” around every refined grid as some solution features — which are only properly resolved on the fine mesh — may move out of the refined region in between those time-steps. The optimal size of this zone depends on the problem and is difficult to estimate *a priori*.

If this “buffer zone” is small, only few points are unnecessarily refined — thus making the computation of the solution on the refined mesh more efficient — but the expensive regridding step has to be done more often to catch any features leaving the high-resolution mesh. If on the other hand the “buffer zone” is large, then the computation can become the bottleneck as too many points are refined speculatively. But that in turn makes it possible to do the regridding less often.

2.1.3.2 Clustering

The process of taking a point cloud and grouping these efficiently — accord-

The following sections gives a brief overview of finite volume methods and is partly based on the material from [17]. For simplicity, u is treated as a scalar quantity but the key points of the discussion hold true for the case of u being a vector quantity as well. The biggest difference lies in how the solution to the Riemann-problem is computed (see for example van Leer [10] or Roe [15]).

2.2.1 Derivation

Consider the following two-dimensional homogeneous conservation law:

$$\frac{\partial}{\partial t}u(x, y, t) + \frac{\partial}{\partial x}f(u(x, y, t)) + \frac{\partial}{\partial y}g(u(x, y, t)) = 0. \qquad (2.1)$$

If we now proceed to integrate Equation 2.1 over the discrete volume Ω with boundary

1.

$$\sum_{j=0}^{N}$$

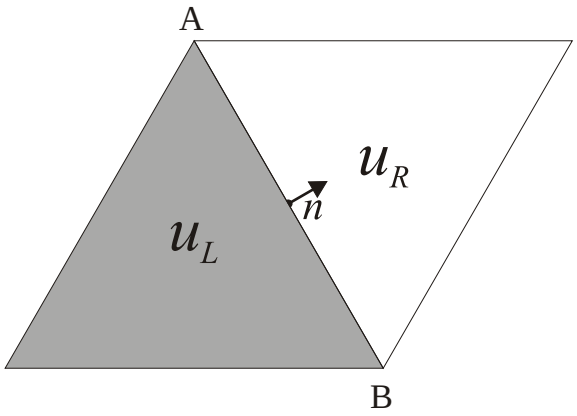


Figure 2.3: Conventions for the flux computation across edge AB (as seen from the shaded element).

the numerical flux is at the mid-point of each edge as that guarantees that Equation 2.3 holds true.

For example, the standard upwind flux — which is the one used by our implementation — across edge AB (as shown in Figure 2.3) takes the form

$$f_{\mathrm{AB}} \cdot n_{\mathrm{AB}} = \begin{cases} f_{\mathrm{AB}}(u_L) \cdot n_{\mathrm{AB}} & \text{if} \quad \lambda_{\mathrm{AB}} \cdot n_{\mathrm{AB}} \geq 0 \\ f_{\mathrm{AB}}(u_R) \cdot n_{\mathrm{AB}} & \text{otherwise} \end{cases} \tag{2.4}$$

for the locally frozen wave-speed

$$\lambda_{\mathrm{AB}} = \begin{cases} f(u_R) - f(u \end{cases}$$

Nonetheless they can be dealt with in exactly the same manner as for non-adaptive methods, see for example van Leer [10] or LeVeque [12] for good discussions of the problem.

2.2.5 Higher-Order Accuracy

Much effort has been invested to achieve better than first-order accuracy with finite volume methods. The first hurdle is Godunov's Theorem, which states that non-oscillatory constant coefficient schemes can be at most first-order accurate. This has been overcome by the introduction of non-linear schemes such as MUSCL, ENO or Flux Corrected Transport (FCT).

One of the more popular ones is van Leer's MUSCL [9] approach, which stands for "Monotonic Upstream-Centered Scheme for Conservation Laws". It belongs to the class of Godunov-type methods, a class of non-oscillatory finite volume schemes that incorporate the (exact or approximate) solution to Riemann's initial-value problem (or a generalisation thereof). Instead of

Equation 2.6 is satisfied if u is of the form

$$u = u + r \cdot L$$

for r being a vector from the centroid of Ω_j and a gradient operator L. Most approaches follow Batten *et al.*'s [3] recommendation to construct a gradient plane through three nearby centroids A, B and C with normal vector

$$n = (P_A - P_B) \times (P_C - P_B), \quad \text{with } P_i = \begin{bmatrix} x_i \\ y_i \\ u_i \end{bmatrix}$$

and the subsequent gradient operator

$$(\text{ABC}) = \begin{cases} \begin{bmatrix} -n_x / n_u \\ -n_y / n_u \end{bmatrix} & \text{if } n_u > \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{otherwise} \end{cases} . \qquad (2.7)$$

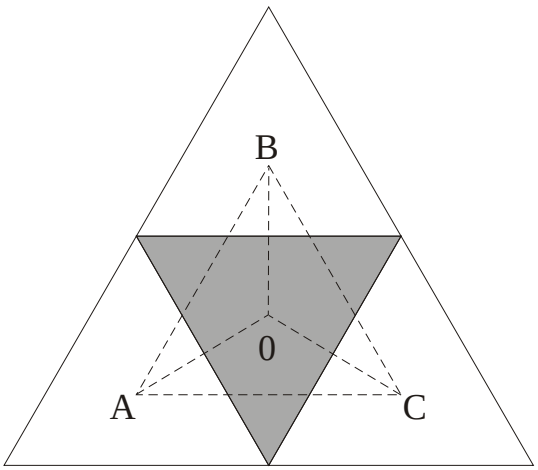


Figure 2.5: Naming Convention for the Limiting Procedure.

The gradient operator defined by Equation 2.7 is not yet limited and as

2.2.3). The limiting of the gradient operator therefore plays an important role as it directly influences the character and accuracy of the solution.

We present a short overview of the limiters used in Chapter 4. The labelling used relative to the "current" element 0 is shown in Figure 2.5; u_0 refers to the value in the current cell whereas u_k refers to the value of the element on the other side of edge k. Similarly, r_{0k} is the vector from the centroid of 0 to the midpoint of edge k.

2.2.5.1 Limited Central Di□erence (LCD)

The LCD-limiter is one of the earliest (and still most widely used) limiters in the context of MUSCL-schemes and is the most di□usive of the limiters presented here — but still an improvement over first order schemes. This limiter's advantages lie in its simplicity and speed.

1. Construct the unlimited gradient operator

$$L = (\text{ABC}).$$

2. For each edge k calculate the scalar

□
□

$$\alpha^k =$$

2. Set

$$L_{\mathrm{MLG}} = L^i \quad \text{such that} \quad |L^i| = \max_{0 \le k \le 3} |L^k|.$$

From this formulation, it is easy to see that the MLG-limiter is slightly more than four times as expensive as the LCD-limiter.

2.2.5.3 Projected LCD (PLCD)

The most recent of the three limiters which have been applied in the context of this paper is Hubbard's [7] Projected LCD-limiter, which relies on the construction of the "Maximum Principle" (MP) region. This region is created from a set of inequalities — precisely one for each edge of the element — around the centroid of the cell. All points lying within this region satify the local maximum principle which in turn guarantees that no over- or undershoots can occur in the linear reconstruction.

1. Construct the unlimited gradient operator

$$L = \quad (\quad \mathrm{ABC}).$$

2. If L does *not* need to be limited, set

$$L_{\mathrm{PLCD}} = L.$$

3. Otherwise construct the MP region defined by

$$\begin{array}{rccccc} & \min(u_k, u_0) & \le & u_0 + r_{0k} \cdot L & \le & \max(u_k, u_0) \\ \Leftrightarrow & \min(u_k - u_0, 0) & \le & r_{0k} \cdot L & \le & \max(u_k - u_0, 0) \end{array}$$

for each edge k, and

4. Project L onto the closest point of the MP region so that

$$L_{\mathrm{PLCD}} = \operatorname*{proj}_{\mathrm{MP}}(L).$$

Although the numerical construction of the MP region is not as expensive in terms of operation count, it is certainly more complicated than programming the MLG-limiter, which is nearly trivial if built onto a working implementation of the LCD-limiter.

Chapter 3

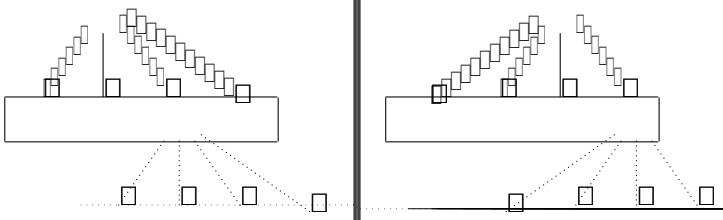
Using Subdivision for Refinement

The proposed algorithm is based on an ordinary unstructured low-resolution mesh which we will refer to as the “top-level” or “base” mesh. Such meshes

Level 0:

Level 1:

Level 2:



for *all* edges of the element.

This serves as a reminder that refinement is a *necessary* step of the algorithm (to maintain accuracy) whereas derefinement is an *optional* step. This becomes an important consideration when dealing with implicit subdivisions in Section 3.4.4.

3.4 The Algorithm

This section describes the steps carried out for an adaptive computation using our subdivision method. This process is governed by three user-defined parameters (in addition to the other problem-dependent data such as the mesh, the top-level time-step ΔT, the particular fluxes used — which in turn describe the equation to be solved, and so on):

1. $_r$ — The sensitivity of the gradient detector to refinement.
2. $_d$ — The insensitivity of the gradient detector for derefinement.
3. N — The maximum level of subdivision carried out on the mesh.

Because we do not want to unnecessarily refine and derefine the same elements over and over again, it is a requirement that $_r$ is *strictly less* than $_d$. As a rough guideline $_d \approx 2\ _r$.

3.4.1 Set-Up

These tasks are only executed once before the computational loop and as such are not *quite* as sensitive to optimisation as some of the other steps that are carried out thousands of times for each time-step.

3.4.1.1 Top-Level Mesh

The first requirement is a top-level mesh on which the subsequent adaptive mesh refinement can be carried out. This base mesh is never modified (although it “acquires” many children) and its vertices never move. This mesh

same time-step. In spite of all adaptivity, the base mesh also has to provide enough initial resolution so that that the gradient detector will be able to properly estimate where the mesh — due to the initial conditions — needs subdivision.

3.4.1.2 Initial Conditions

The next step is to populate this mesh with initial conditions. These are given as a function of x and y in the form $u(x, y, 0)$. As the values within each cell are area / volume averages, they cannot simply be sampled at the centroid of the cell. To compute these volume averages analytically, one would have calculate the area under an arbitrary triangular element placed on top of the initial conditions. This is far from impossible but very much dependent on the shape and storage of the initial data.

To facilitate conditions of a more general nature, we have adopted a stochastic approach which samples the initial conditions at a fixed number of (pseudo-)random points within the triangle and then averages these samples to give an estimate of the proper average. The points in the triangle are generated in barycentric coordinates to guarantee a normal distribution as presented by Turk [19] while the pseudo-random number sequences are

During the subdivision process the proper nesting has to be enforced so that neighbours in the computational mesh will only di□er by *at most* one level of subdivision. This necessitates what we call implicit subdivisions (see Section 3.4.4) which come from nesting rather than computational accuracy requirements (i.e. the gradient detector).

Figure 3.3: A low-resolution base mesh and the same mesh adapted to the initial condition of the rotating slotted cylinder problem with 5 levels of subdivision.

3.4.2 Advancing Time

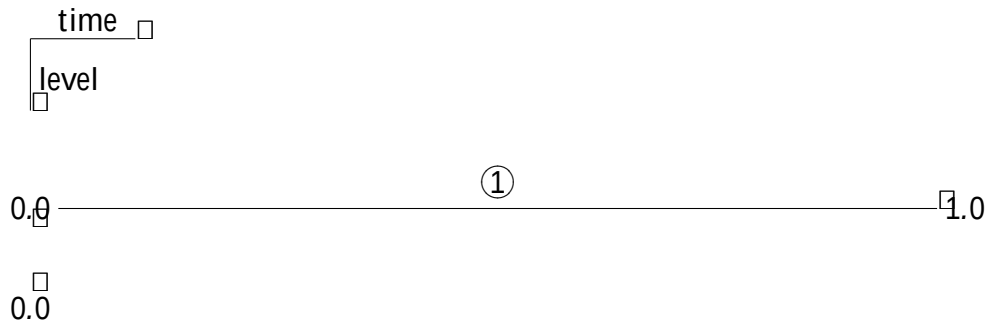
resolution — and thus might be updated at a di□erent frequency than the current cell.

Thus we need to (at least) halve the time-step for each additional level of refinement, i.e.

$$\Delta t(M) = 2^{-M} \Delta T,$$

where ΔT is the time-step used on the top-level mesh.

The formulation of Equation 3.2 implies that for the update of a particular cell one needs to access the states in the neighbouring cells. These neighbouring cells — due to our nesting requirement — may be of a higher (by one level), lower (again by one level) or of the same resolution. The time-stepping of each level therefore has to be done with great care so that the neighbouring elements from an adjacent level are also evolved to the proper point in time. We refer to this as "lock-step" time-stepping and its pyramid-structure is depicted in Figure 3.5.



subdivision. We keep a list for each level of refinement that contains the triangles of that level of subdivision. This allows us to efficiently iterate over all the elements in a level without having to traverse the whole quad-tree data structure. These lists are incrementally updated to reflect promotions across levels as more and more elements are subdivided. The only time the lists are reconstructed from scratch is after a derefinement operation has been completed.

The only obstacle to the direct application of Equation 3.2 is the possibly different resolution of the neighbours used for the numerical fluxes. The three possibilities are depicted schematically in Figure 3.6.

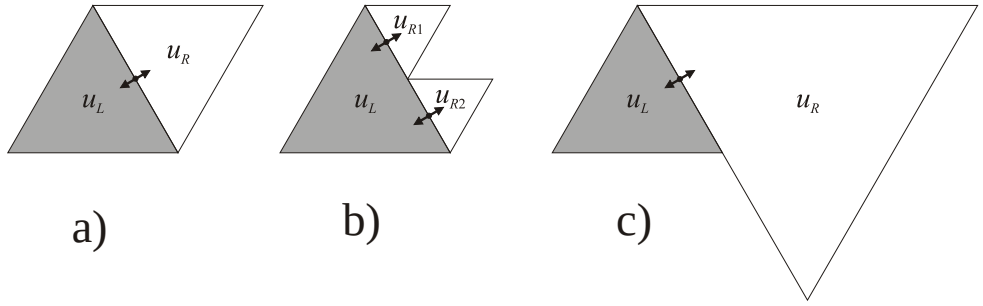


Figure 3.6: Flux Computation for a) a neighbour of the same resolution, b) neighbours of higher resolution and c) a neighbour of lower resolution. No other cases can occur because of the nesting requirement which does not allow the level of subdivision of neighbours to differ by more than one.

3.4.3.1 Edge Fluxes

a) This is the only trivial case as it is identical to the application of a finite volume scheme on a regular unstructured mesh. Therefore one can directly compute the flux through edge k of element 0 as

$$f_k(u_0^{n+1/2} + r_{0k} \cdot L_0^n, u_k^{n+1/2} + r_{k0} \cdot L_k^n) \cdot n_k. \qquad (3.4)$$

b) The second case is treated as if edge k were two edges, k_1 and k_2, with half the original length each. In Figure 3.6b, k_1 would separate states u_L and u_{R1} whereas k_2 would separate u_L and u_{R2}. This of course makes changes to the limiting procedure necessary as the gradient operator L_0 is not evaluated at r_{0k} (the midpoint of edge k) anymore but rather

at r_{0k_1} and r_{0k_2}. The numerical flux can then be written as

$$\begin{aligned} & f_{k_1}(u_0^{n+1/2} + r_{0k_1} \cdot L_0^n, u_k^{n+1/2} + r_{k_10} \cdot L_{k_1}^n) \cdot n_{k_1} \\ + & f_{k_2}(u_0^{n+1/2} + r_{0k_2} \cdot L_0^n, u_k^{n+1/2} + r_{k_20} \cdot L_{k_2}^n) \cdot n_{k_2}. \end{aligned} \tag{3.5}$$

Because these fluxes are written in terms of the "new" normal vectors n_{k_1} and n_{k_2}, no averaging has to be done because the normal vectors are scaled by the length of the respective interface.

c) This situation is in some sense the inverse of case b) and as such, special care has to be taken while constructing and limiting the gradient operator L_k because itlyT.9a794T.9resptoroersetpthe g1.9555Tf9.2010T122.53

resolution) “bu□er zones” to cope with events until the next remeshing step takes place.

3.4.3.2 Changes to Gradient Operators

As alluded to in the previous section, the gradient operators need to be modified so that the points where the fluxes are computed — which are not restricted to the midpoint of the edges anymore — are properly limited. These changes are two-fold. On the one hand, it is necessary to decide how the unlimited gradient operator is constructed, given that there may be more than three neighbouring states. And on the other hand, we need to make sure that these are properly limited at the points where they are evaluated.

The only di□culty in the construction of the unlimited gradient operator (ABC) occurs when any of the neighbours have a higher resolution than

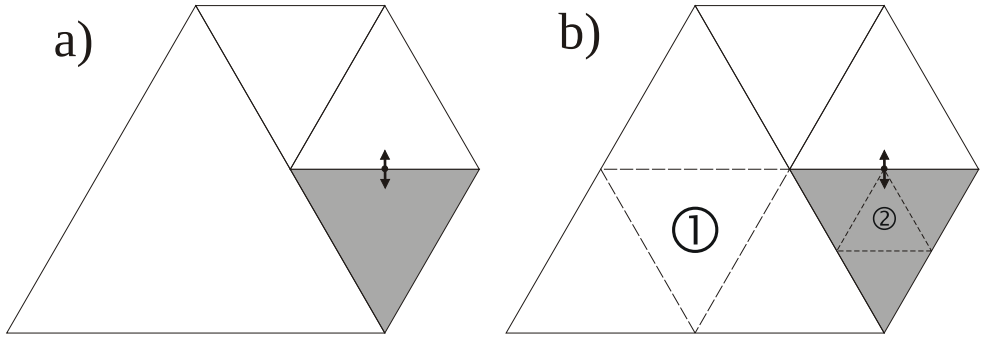


Figure 3.7: a) An accuracy based subdivision becomes necessary for the current (shaded) triangle at the indicated edge and b) makes an implicit subdivision (1) necessary which has to be completed before the original refinement (2) can take place.

2. Allocate space for the children and generate their vertices.

3. Compute the states in the children. This is done by evaluating the usual gradient operator of the parent at the children's centroids. This ensures proper conservation and retains more information than simply duplicating the constant parent state to all children. As the gradient operator is not necessarily correctly limited at the children's centroids this may cause under- or overshoots to appear. But these violations of the maximum principle are unlikely in practice as the children's centroids are rather close to the parent's centroid and therefore only incur small changes from the constant state.

4. Lastly, compute (in the case of the newly generated children) or update (for all neighbours of the parent) the connectivity information so that all neighbour-pointers now refer to the more finely resolved children instead of their parent.

After this process is completed, the parent effectively becomes dormant as no more computations are carried out on it; it is only needed in the hierarchy to provide information about its children.

It is essential that Step 1 takes place first so that all the implicit subdivisions take place before the real ones. If, for example, the normal subdivision labelled (2) in Figure 3.7 were to be executed first, then the mesh would end up in an invalid state because the refinement levels of two adjacent triangles would differ by *two*.

Chapter 4

Results

To verify the validity of our approach, the proposed adaptive scheme has been implemented in the C++ programming language, whose object-oriented approach seems to be favoured by many implementations of AMR (for example Hornung and Trangenstein [6]) over the more traditional FORTRAN. The code used to obtain these results allows for arbitrary unstructured meshes with no-flux or periodic boundary conditions. The problem specification and the associated fluxes are kept separate from the algorithm in order to facilitate its use for different types of computations.

The computation of errors in the L_1 norm against known solutions is somewhat problematic on unstructured meshes. The approach we have adopted is to interpolate from the irregular AMR mesh to a regularly spaced grid and compute the norm from there. This introduces an additional interpolation error to those estimates; but as the size of the regular mesh increases, the interpolation error tends to 0. On a regular grid with N points we thus compute the L_1 error as

$$\frac{1}{N} \sum_{i,j} |u_{i,j} - u_{\text{exact}}(x_i, y_j)|.$$

4.1 Linear Advection

Problems of this type are rarely solved in practice as the analytic solution is known for most advection profiles. Nevertheless, they are a valuable and well-understood tool used for testing and comparing different numerical schemes.

4.1.1 Estimating Order of Accuracy

A similar set-up to Batten *et al.* [3] has been used to estimate the order of accuracy of our scheme. The estimate is computed by solving

$$u_t + u_x + u_y = 0$$

(i.e. linear advection with wave-speed $(1, 1)^T$) with the initial data

$$u(x, y, 0) = \sin(2\pi x) \sin(2\pi y).$$

on a mesh consisting of right-angle triangles on the region $[0, 1] \times [0, 1]$ with different levels of refinement and then comparing the respective L_1-errors.

The given problem is not well-suited to adaptive computation

The contour plots in Figure 4.2 and 4.3 support the L_1-results. Both the MLG- and the LCD-limited solutions exhibit grid based distortion, although it is far less pronounced with the MLG-limiter. The PLCD-limiter does very well — the peaks have nearly the same magnitude as the highly compressive MLG-scheme and no distortion is evident. This matches the observations made by Batten *et al.* [3] and Hubbard [7].

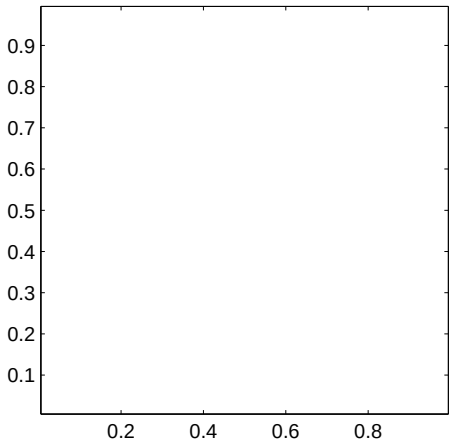


Figure 4.2: Contours of the exact (left) double sine wave function and the MLG-solution. The MLG-solution shows a directional bias which leads to a subtle distortion in the lower-left quadrants vs the upper-right ones.

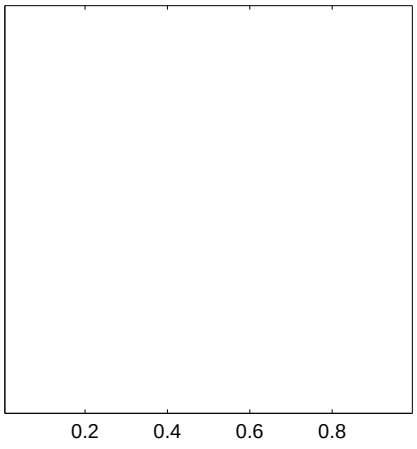


Figure 4.3: Contours of the PLCD-solution (left) and the LCD-solution. Note the smooth profile of the PLCD-limiter in contrast to the heavily distorted LCD-solution.

4.1.2 Rotating Slotted Cylinder

Zalesak [21] first presented the slotted cylinder in 1979 and it is regarded as one of the hardest benchmarks for advection schemes. The version used here is scaled and shifted but otherwise identical. The problem is to solve

$$u_t + \left(y - \frac{1}{2}\right) u_x - \left(x - \frac{1}{2}\right) u_y = 0 \tag{4.1}$$

with initial condition

$$u(x, y, 0) = \begin{cases} 1 & \text{if } r > 0.15 \quad (|x - 0.5| \leq 0.025 \quad y - 0.5 \leq 0.1) \\ 0 & \text{otherwise} \end{cases}$$

for $r = \sqrt{(x - 0.5)^2 + (y - 0.5)^2}$. Equation 4.1 describes a clock-wise rotation of the whole region about the point $(0.5, 0.5)^T$. A whole revolution of the region is completed at $t = \pi$ so that the solution is periodic with period π. This also means that the exact solution at $t = \pi$ is equal to the initial data.

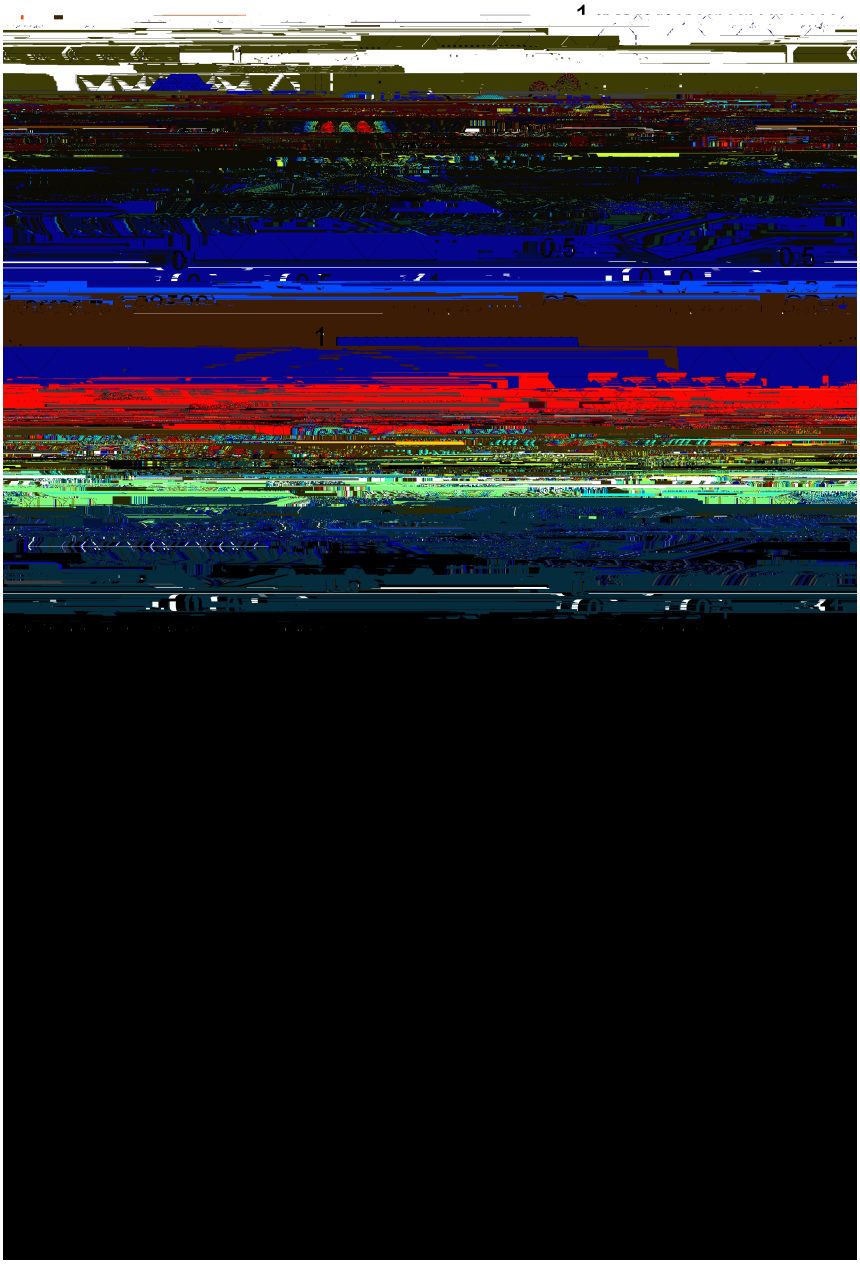
The problem is solved in the region $[0, 1] \times [0, 1]$ with no-flux boundary conditions on a top-level mesh with 48 equilateral triangles as seen in Figure 3.3. The time-step used is $\Delta t = 0.05$ which upon initial observation is outside the stability-region defined by Equation 3.3. This is no cause for concern though as the advection-vector can safely be set to 0 for $r \geq 0.3$ where $u(x, y, t)$ is identically zero. This extends the stable region to $\Delta t \leq 0.063927$. For the results in Figure 4.4 a refinement threshold of $_r = 0.025$ and a derefinement limit of $_d = 0.05$ were used in conjunction with five levels of refinement.

4.1.2.1 Quality of Solution

As one can see from the L_1-errors in Figure 4.4 the limiters play a crucial role: All of them[1] limit correctly across the adaptive mesh; no oscillations are present in the solution, which would certainly not be the case with more traditional second order methods used on regular grids such as Lax-Wendro, Warming & Beam or — to a lesser extent — Fromm. If the limiting is removed for any of the linear reconstructions then the solution grows unboundedly within a *single* top-level time-step due to the sharp discontinuities in the initial data.

1st Order (L1 error = 0.04477)

1st Order



From the first-order scheme to the MLG-operator each successive limiter *halves* the error in this example, resulting in a ten times more accurate solution with the MLG limiter than the first-order scheme. This stresses the importance of a good reconstruction- / limiting-procedure over "raw" resolution. Nevertheless, the MLG-limiter seems to introduce a very slight distortion to the top-right of the slot and to the bottom-right "tip". This minuscule distortion is also evident in higher-resolution computations with further subdivisions.

The Projected LCD-slope is slightly more diffusive than the Maximum Limited Gradient — the edges of the "discontinuity" are now spread out over four of the most-refined triangles in contrast to three on the MLG-limited computation. But the projected limiter is less susceptible to grid based distortion as the solution looks perfectly symmetric.

The widely used Limited Central Difference-approach is even more diffusive but it manages preserves most of the original shape — although the top of the cylinder is now smoothly rounded instead of flat and the slot is beginning to close (or at least rise).

The solution produced by the first-order scheme is nearly unrecognisable. It is not immediately apparent which side of the cylinder had the slot embedded in it and which had not. Many triangles are refined due to the overly diffusive nature of the method.

4.1.2.2 Impact of Limiters

But accuracy is not the only reason for which it is important to choose a good reconstruction- / limiting-scheme. The compressiveness of the limiter directly influences the cost of the adaptive computation. If the scheme used is too dispersive, the discontinuities get spread out over a larger area which then needs more refined triangles.

This is very well illustrated by Figure 4.5 which depicts the cost of each top-level time-step (similar to the measure defined in Section 4.2.1 but only for a *single* top-level time-step instead of the whole computation). The cost for the Maximum Limited Gradient-limiter stays nearly constant during the whole computation, which means that the information in the solution is well retained and that the derefinement removes about as many "old" elements as new ones are being refined.

The PLCD-solution is not quite as efficient as it causes slightly more elements to be subdivided at a an additional cost of about 28% for each time-step near the end of the computation. The number of elements only seems to increase at a very low rate however.

Both the first-order upwind update and the LCD-scheme are not a very

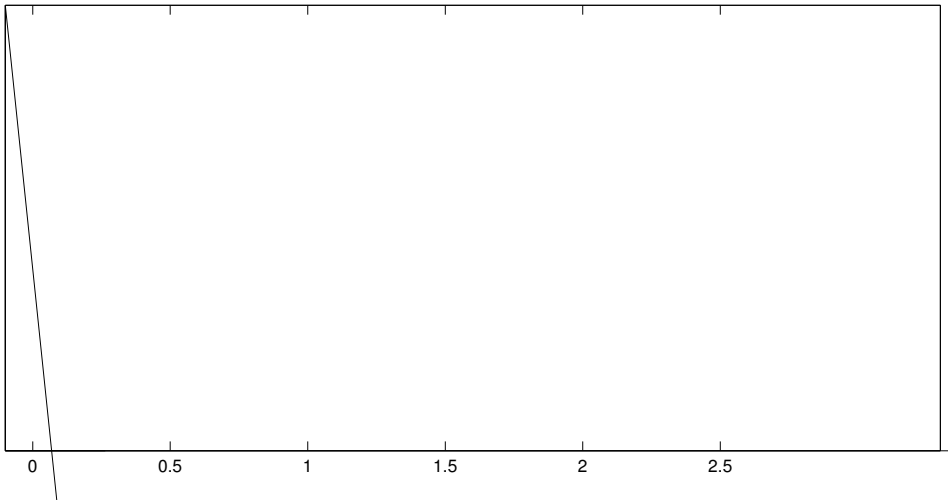


Figure 4.5: The workload / cost per top-level time-step during the 5-level solution of the rotating slotted cylinder.

good choice for adaptive computations in general and problems that contain discontinuities in particular. The use of the first-order method makes the whole computation (i.e. the area under the respective graph) more than twice as expensive as with the most compressive limiter. Nonetheless, they are an important foundation on which to formulate more accurate methods — the MLG-operator for example uses the steepest of four different LCD-gradients.

Taken together, this implies that — particularly in the context of adaptive methods — a more expensive but less diffusive limiter may well offset the higher cost of its construction during a whole computation. If one takes the cost of the computation of the Limited Central Difference as a base-line, then the MLG-operator is slightly more than four times as expensive to construct while the PLCD-limiter needs about two to three times as much. We have found it essential to cache limited gradient-operators for each triangle to avoid their recomputation as they are needed at least four times — once for the computation in the current triangle and then once for each neighbour's computation.

meshes or solve existing problems with higher accuracy. It is an important

solution on there. The adaptive computation has to be used as the "baseline-cost" to compare against because it is hard to predict the cost *a priori* due to the variable number of cells.

4.2.1 Estimating Cost

The cost of the AMR computation is defined to be

$$C_{\mathrm{AMR}} = \left\lceil \frac{T}{\Delta t} \right\rceil \cdot \sum_{\text{all tris}} 2^{M}$$

where T is the final time and M represents the level of subdivision of each triangle. For this particular example, $T = \pi$ and $\Delta t = 0.063927$ so that C_{AMR}

4.2.2 Resulting Errors

Using Equation 4.6 to estimate an equal cost fixed mesh gives $N_e \approx 6700$. This number of elements has a cost of 4909548 ticks due to the various approximations used. The actual number of elements has consequently been fixed at $N_e = 5376$ for which $C_{\mathrm{FXD}} = 3404863$ ticks with a time-step of $\Delta t = 0.00496$.

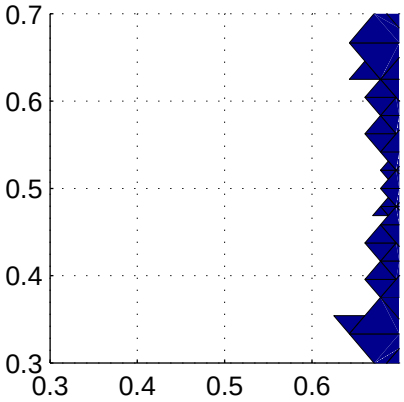


Figure 4.7: Magnification of the results of the AMR and fixed mesh computations at $t = \pi$ for approximately equal cost.

The adaptive computation has an L_1 error of 0.0043 whereas the fixed mesh results in an L_1 error of 0.0158. This means that the AMR method is nearly four times more accurate (according to the L_1 norm) for this particular problem and for a similar amount of computational resources consumed. The comparison is not entirely on equal grounds though as the cost of subdivisions and derefinement are ignored for the AMR scheme. These are not particularly large but hard to quantify in terms of the above-mentioned cost-metric and have thus been omitted.

4.3 Nonlinear Problems

The solution of *nonlinear* hyperbolic partial differential equations is probably one of the largest application areas for finite volume methods.

4.3.1 Burger's Equation

The numerical solution of Burger's equation is considerably more difficult than the previous advection problems. Suddenly, one is confronted with shocks appearing and disappearing, smooth initial data turning into discontinuities and other difficulties. But it is also these features that make the presented adaptive method worthwhile.

The inviscid Burger's equation in two dimensions is defined as

$$u_t + uu_x + uu_y = 0 \tag{4.7}$$

which implies that

$$f(u) = g(u) = \frac{1}{2}u^2$$

when Equation 4.7 is written as

$$u_t + f(u)_x + g(u)_y = 0$$

to match the definition of a conservation law in Equation 2.1.

The method itself stays exactly the same compared to earlier applications — the only thing that needs to be adapted is the numerical fluxes in the problem description. These need to be defined carefully to ensure conservation and thus correct shock speeds.

Equation 4.7 is solved on the region $[0, 1] \times [0, 1]$ with $\Delta t = 0.05$ on the previously used equilateral triangle mesh with five levels of subdivision and the initial data

$$u(x, y, 0) = \begin{cases} \frac{1}{2} & \text{if } x < \frac{1}{2} \text{ and } y < \frac{1}{2} \\ -\frac{1}{2} & \text{if } x > \frac{1}{2} \text{ and } y > \frac{1}{2} \\ \frac{1}{4} & \text{otherwise} \end{cases}. \tag{4.8}$$

This is the same initial data as used by Berger and Oliger [4], although we apply different — namely periodic — boundary conditions. Also, the properly limited schemes implemented do not need numerical work-arounds such as adding artificial viscosity to dampen oscillations that were employed by them.

Due to the different boundary conditions used, no analytic solution to compare against was available. Much progress has been made in recent years in the classification and analytic solution of two-dimensional Riemann-problems (for example Wagner's paper [20]), but work remains to be done until those solutions are easier to construct. In this particular example it

was possible to verify that the general shape of the solution is correct. The shocks and expansion-waves also seem to move with the correct speeds from previous experience in the one-dimensional case.

As seen in Figure 4.8, the adaptive scheme with the MLG-operator gives very good resolution of shock-fronts and chooses appropriate resolutions for features at all levels: constant states use very little computational resources as one would expect and expansion-wave gradients are properly resolved without being overly fine. The other limiters (not shown) do surprisingly well considering the difficult nature of the problem, although the expansion fans seem to be slightly elongated when using the first-order scheme and (to a lesser degree) the LCD-limiter. The stronger dispersion in those limiters is the most likely cause for that.

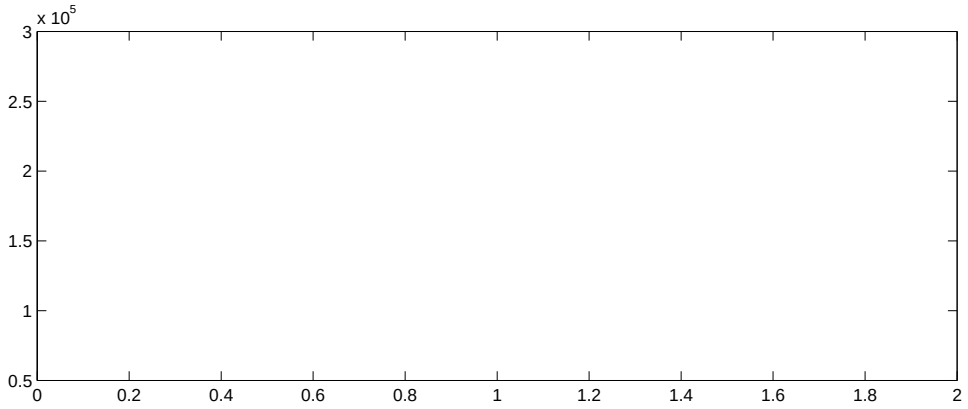


Figure 4.9: The workload / cost per top-level time-step during the 5-level solution of Burger's equation. The whole computation took less than 30 seconds on the author's machine for the MLG-case.

The "adaptive" cost advantage shows up in a slightly different form. The initial data given in Equation 4.8 has *four* plateaus separated by discontinuities. The final solution at $t = 2$ only contains *two* distinct states and therefore contains much fewer highly refined edges where those states meet.

From the computing cost displayed in Figure 4.9 it is immediately apparent that the computation speeds up drastically towards the end. The time needed per time-step near the end of the computation is reduced to nearly a quarter of the time used during its early phase. This results in a noticeable shortening of the computation-time after the 50%-mark. The different limiters have a comparable cost to before; although there is not much separating the MLG-operator from the PLCD-gradient this time around.

In regard to the solution of this problem, Berger and Oliger [4] have remarked that *"this problem is a hard test for mesh refinement because such a large fraction of the region is refined."* Due the ability of our subdivision-based refinement to allow for locally higher resolution without affecting the rest of the mesh (other than through implicit subdivisions of course), a weakness of traditional adaptive mesh refinement has been turned into an advantage.

Chapter 5

Conclusions

An adaptive algorithm for the efficient solution of hyperbolic conservation laws on unstructured meshes has been introduced. The proposed scheme is formulated in terms of standard finite volume methods on triangular elements, although some extensions have been made to allow for inter-level updates with those methods which involve polygons with more than three edges. The simple quad-tree data-structure is essential to the performance of the algorithm as it facilitates the atomic operations such as subdivision into four children, derefinement and neighbour queries on which the method relies. It also restricts neighbouring triangles to have a refinement-difference of at most one level which ensures that there are no abrupt changes in resolution. Another advantage of using subdivision of elements is that no monitoring or adjustment of individual element's anisotropy is necessary if the original top-level mesh is well-formed (e.g. a Delaunay triangulation). Furthermore, the introduction of "Just-In-Time" refinement makes it possible to defer any decisions about the resolution at which computations are to be done until they are actually about to happen. This is achieved by very fine-grained adaptivity: The "adaptiveness" of the scheme is not something that is done every now and then between traditional computations — it is an integral part of the algorithm.

The mesh refinement method was implemented in two dimensions with a variety of [9] slope-limiters and has been successfully used to solve both traditional linear problems as well as non-linear ones. These numerical experiments have shown the superior efficiency of the scheme compared to computations on fixed meshes. They have also demonstrated the necessity of high-quality finite volume methods of at least second order accuracy for ef-

the diffusive behaviour of *any* scheme is lessened by the use of our method compared to fixed mesh computations. We have found Batten *et al.*'s [3] Maximum Limited Gradient to be very good at resolving discontinuities or shock fronts — although it is not immune to mesh distortion — and Hubbard's [7] Projected LCD-operator excels on continuous problems and shows no dependency on the underlying mesh geometry.

5.1 Further Work

There are quite a few distinct areas which may prompt further research. Foremost would be the extension to non-linear systems of equations — for example using Roe's approximate Riemann-solver [15] in conjunction with the proper averages. There are already quite a few reformulations of "popular" PDEs into conservation form for genuinely higher-dimensional finite volume methods on unstructured grids, [7] for example shows how they can be applied to the Shallow Water equations. This should be fairly straightforward to do as our adaptive method does rely on largely unmodified finite volume schemes.

The extension of the scheme to three dimensions is another interesting subject, although we do not foresee any new topological difficulties in the process. The main changes would be to replace triangles with tetrahedra and to modify the subdivision and data-structures accordingly.

It may very well be worthwhile to replace our rather crude threshold-based gradient detector with something more sophisticated as the linear reconstruction of the second-order schemes allows us to compute accurate solutions for linear data *without* the need for much refinement. The work of Barth and Larson [2] on error estimates for finite volume methods may be relevant in that regard.

The behaviour of different gradient-operators for particular types of data prompts the question whether it may be appropriate to use different operators at different levels of refinement or for exceeding different thresholds in the gradient-detector. One could for example apply the PLCD-limiter on the children of a triangle if the refinement was a "close call" (i.e. relatively smooth states) and use the MLG-operator if the region was determined to contain discontinuities — while still maintaining proper conservation.

Appendix A

Acknowledgments

Many thanks to my supervisor, Dr. P. K. Sweby, for many helpful suggestions and hints as well as general guidance towards the topic of this paper.

Prof. M. J. Baines has also been very generous in answering many of my questions.

I am also grateful to the EPSRC for providing partial financial support.

Last but not least, I wish to thank my parents for putting up with me for the past 24 years and making all this possible.

I would also like to thank the following for keeping me (remotely) sane during all this time:

- *Björk*, *Elton John*, *Lamb*, *Last Days of April*, *Mazzy Star*, *The Notwist*, *The Prodigy* and *Yoko Kanno*.
-

Appendix B

Bibliography

[1] M. Ahmadi and W. S. Ghaly: "A Finite Volume Method for the Two-Dimensional Euler Equations with Solution Adaptation on Unstructured Meshes", *6th ASME International Congress on Fluid Dynamics and Propulsion, Egypt* (1996)

[2] T. J. Barth and M. J. Larson: "*A Posteriori* Error Estimates for Higher Order Godunov Finite Volume Methods on Unstructured Meshes", *NASA Technical Report* NAS-02-001 (2002)

[3] P. Batten, C. Lambert and D. M. Causon: "Positively Conservative High-Resolution Convection Schemes for Unstructured Elements", *International Journal for Numerical Methods in Engineering* 39 (1996)

[4] M. J. Berger and J. Oliger: "Adaptive Mesh Refinement for Hyperbolic Partial Di□erential Equations", *Journal of Computational Physics* 53 (1984)

[5] M. Berzins, R. Fairlie, S. V. Pennington, J. M. Ware and L. E. Scales: "SPRINT2D: Adaptive Software for PDEs", *ACM Transactions on Mathematical Software* 24-iv (1998)

[6] R. D. Hornung and J. A. Trangenstein: "Adaptive Mesh Refinement and Multilevel Iteration for Flow in Porous Media", *Journal*

A POS

[9] B. van Leer: