How Efficient are Delaunay Refined Meshes? An Empirical Study

Bruce Simpson

School of Computer Science; University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
rbsimpson@uwaterloo.ca

Summary. Given a data function, $f(x, y)$, defined for (x, y) in a domain,D and an error measure for approximating f on D, we can call a piecewise linear function, $f^{pl}(x, y)$, acceptable if its error measure is less than or equal to a given error tolerance. Adaptive Delaunay Refinement (ADR) is one approach to generating a mesh for D that can be used to create an acceptable $f^{pl}(x, y)$. A measure of the efficiency of methods for generating a mesh, M, for piecewise approximation is the size of M. In this paper, we present empirical evidence that ADR generated meshes can be twice a large as necessary for producing acceptable interpolants for harmonic functions. The error measure used in this study is the maximum of the triangle average L2 errors in M. This observation is based on demonstrating a comparison mesh generating using maximal efficiency mesh theory as reviewed in the paper. There are two different approaches to point placement commonly used in ADR, edge based refinement and circumcenter based refinement. Our study indicates that there is no significant difference in the efficiency of the meshes generated by these two approaches.

1 Introduction

The meshing context of this paper is piecewise linear function approximation on a planar domain D. I.e. Given a function $f(x, y)$ defined for $(x, y) \in D$, create an unstructured triangular mesh, M, for D and the coefficients of a piecewise linear approximation, $f^{(pl)}(x, y) \approx f(x, y)$. Unstructured meshes pose an efficiency-computational cost trade-off. Local computations with unstructured meshes tend to be more complex than with structured grids. However, a given accuracy in $f^{(pl)}(x, y)$ can usually be achieved by an unstructured mesh with significantly fewer vertices that needed by structured grids. So, for global computations, unstructured meshes can be more efficient by virtue of being smaller. Iterative adaptive h-refinement is a long standing mesh generation

techniques that aims to provide this efficiency (since 1970s.) The combination of this technique with Delaunay meshing has been used almost as long (since 1980s). We will use the abbreviation ADR for adaptive Delaunay h-refinement in the sequel. Clearly, there is a limit, for a given f and target error, on how small the meshes that meet this target error can be. A mesh that meets the error target with a minimal number of vertices is a maximal efficiency mesh, Simpson [26]. In this paper, we address questions of how efficient are the meshes generated by adaptive Delaunay refinement, relative to maximal efficiency meshes.

In §2, we review the development of adaptive Delaunay refinement methods and explain the details of the versions that are used in our computations. In particular, we distinguish two choices for the insertion vertex used for refining a triangle, T, the midpoint of a longest edge of T, which we will denoted $LEBis(T)$, and the circumcenter of T, which we will denote by $CC(T)$. In §3, we present in detail a demonstration of ADR for piecewise linear interpolation of a specific harmonic function. This demonstration leads to a discussion of a class of simple meshes consisting of isosceles right angled triangles. This class is closed under adaptive h-refinement or ADR for either $LEBis(T)$ or $CC(T)$ type insertion and any of these methods produce the same result whan applied to an initial mesh in the class.

The demonstration of §3 includes evidence that ADR generated meshes can be roughly twice as large as maximum efficiency meshes, for isotropic data. This evidence is based on a comparison to the size of a highly efficient mesh created by a computation specific to the data function. The computation of the comparison mesh is detailed in §4 and an overview of the theory supporting this computation is given in §5. In §6, a second example is presented in less detail that confirms the features of the example discussed in §3. The results of this study are highly consistent with a similar study by E. F. D'Azevedo, [10], which demonstrates that adaptive meshes created by the program PLTMG [3], discussed below, are about twice as large as specially constructed comparison meshes.

2 Iterative Refinement Methods

Iterative refinement refers to a hierarchy of mesh generation methods, as shown in Figure 1. We will use this hierarchy to discuss related previous work and the methods used in the computations of this paper. Basic iterative h-refinement (BR) requires a size measure for triangles, $size(T)$, an input a mesh M_0 for D, and a maximum size tolerance, $sizeTol$, which may be variable across D. The method then attempts to create a mesh, M for D such that $size(T) \leq sizeTol$ for every $T \in M$. Delaunay refinement methods, (DR), are basic h-refinement methods in which M_0 and M are constrained Delaunay triangulations. Adaptive refinement methods, (AR) are basic h-refinement methods related to specific applications, such as piecewise linear

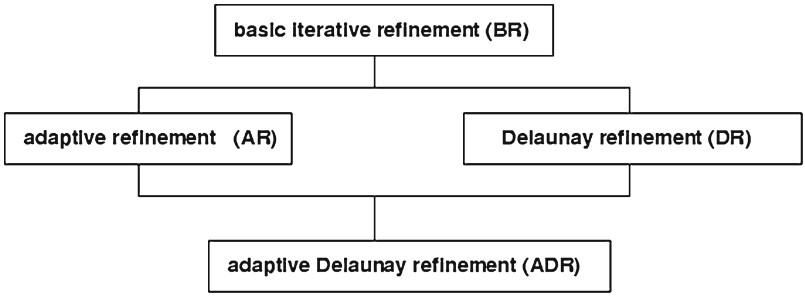


Fig. 1. A hierarchy of iterative refinement methods

approximation. These methods are aware of a data function through an error measure for triangles. They require an error tolerance, $errTol$ and the method attempts to create a mesh such that the error in every triangle does not exceed $errTol$. Adaptive Delaunay refinement (ADR) methods have characteristics both of adaptive h-refinement and Delaunay refinement methods. This requires a reconciliation of the the refinement criterion on $size(T)$ in the case of Delaunay refinement with the triangle error measure in the case of adaptive h-refinement.

2.1 Basic and Adaptive Refinement Methods

To describe the basic attributes of iterative refinement methods, we use the following pseudo code algorithmic description of basic refinement.

```
1: basic iterative refinement(M)
2: initialize S by the triangles of M
3: while S is not empty do
4:   select T using S
5:   if Refine(T) then
6:     V_insert = SelectNewVertex(T, M)
7:     (S_add, S_rem) = Insert(V_insert ,M)
8:     S ⇐ S + S_add − S_rem
9:   else
10:    S ⇐ S − T
11:  end if
12: end while
```
(1)

The algorithm computes a sequence of meshes, $M_n, n = 0, 1, 2, ...$ on the domain D. M_{n+1} is obtained from M_n by selecting a triangle T in M_n such that a method predicate, `Refine(T)`, has value True and inserting a related vertex into M_n, The input parameter is mesh M, which is identified with M_0 of the above mentioned sequence. This description uses several abstract procedures that are incompletely described by their names. Basic refinement methods can

be illustrated by simple model examples of these procedures, which have been associated with early refinement methods. We will describe the methods that we review, and those used in the computations of this paper, by specifying these processes in the child methods of the method hierarchy of Figure 1. The description uses a dynamic set, S, of triangles of M which, at any time contains the triangles of M that may trigger refinement. Perhaps a simpler explanation is that at all times $T \in M - S$ implies `Refine(T) == False`. `Insert`(V, M) takes M_n as input argument for parameter M, and returns M_{n+1} as output argument. The procedure returns two sets of triangles; S_{add} are the triangles in M_{n+1} but not in M_n, and S_{rem} are in M_n but not M_{n+1}.

A simple instance of this basic refinement algorithm uses simple longest edge bisection of T. For this, `SelectNewVertex` of step 6: of (1) returns the midpoint of a longest edge of T for V_{insert} and `Insert` of step 7: splits T into two new triangles T_A and T_B using a new edge from V_{insert} to the vertex opposite the split edge of T. If the edge being split is internal to the mesh, then the same split is performed on the neighbouring triangle, T_{neigh}, producing two more triangles T_C and T_D in the refined mesh. The implication for step 7: of this method is that `Insert` returns $S_{add} = \{T_A, T_B, T_C, T_D\}$ and $S_{rem} = \{T, T_{neigh}\}$. In a 1984 paper, [18], Rivara presented a class of basic h-refinement methods using simple longest edge bisection.

It is common to implement the selection of step 4: of (1) so that larger triangles are selected before smaller ones. The need for quick access to the largest triangle in S complicates the data structure for this dynamic set, Shewchuck, [25]. The longest edge propagation path of Rivara, [19, 20], is a heuristic for finding a local maximum edge length in the mesh. For any triangle T_0 in M, the longest edge propagation path of T_0, $Lepp(T_0)$, is the sequence $\{T_j\}_{j=0}^{N}$, where T_j is the neighbor triangle on a longest edge of T_{j-1}, and longest edge $(T_j) >$ longest edge (T_{j-1}), for $j = 1, \ldots, N$. This condition determines $N \geq 0$. Consequently either $Lepp(T_0)$ terminates with T_N that has a longest edge that is a constrained edge of M, (e.g. a mesh boundary edge), or it terminates with a pair of neighbouring triangles, $(T_N, \bar{T})$ such that their common edge is a longest edge of both. For the computations of this paper, these ideas affect step 4: of (1); i.e. a triangle T_0 is initially selected from S and then T is set to T_N of $Lepp(T_0)$. It is common then that T is not itself in S. This is a technicality of the combined use of *Lepp* and longest edge bisection discussed in §2.1 immediately following ; see Rivara, [19], for a discussion of it.

6: V_{insert} = SelectNewVertex(T, M)

The general pattern of this procedure is that it computes a candidate vertex V_{cand} for V_{insert} that will improve the configuration of triangles in the mesh near V_{insert} relative to the refinement criteria. If V_{cand} is so close to a boundary edge, e, that its insertion would result in an obtuse triangle adjacent to e, then V_{cand} is said to encroach on e. Inserting V_{cand} in this case would violate the

assumptions we have placed on the meshes of the refinement sequence. So if V_{cand} does not encroach on any boundary edge, then it can be returned as V_{insert} by `SelectNewVertex(`T, M`)`, otherwise the midpoint of an encroached edge is returned. Two standard choices for V_{cand} are the circumcenter of T, $CC(T)$, or the midpoint of a longest edge of T, which is referred to as longest edge bisection, $LEBis(T)$. We will compare the efficiency of meshes generated by these two possible choices in §3.

Adaptive Refinement

The basic refinement methods that have just been discussed contain the geometric features of adaptive h-refinement methods for triangular meshes which starting appearing in the literature in the late 1970s[1]. The context of these refinement methods was piecewise linear function approximation, $f^{(pl)}(x, y)$, typically by the finite element method, for a data function, $f(x, y)$, typically defined implicitly by a partial differential equation. The implication for our refinement method hierarchy is that the `Refine(T)` predicate uses an error estimate of some measure of the error, $err(x, y) = f(x, y) - f^{(pl)}(x, y)$. The literature on error estimation is large and continues to grow; we do not review it here. Bank and Weiser published an early paper on error estimation for this purpose, [5]. This research was done at an early stage of the sustained development by Bank and collaborators, of the pde solving software PLTMG which incorporates adaptive h-refinement, [3].

2.2 Delaunay and Adaptive Delaunay Refinement Methods

As mentioned above, for Delaunay refinement methods, all the M_n are constrained Delaunay meshes; but in addition, we require that no boundary edge is the longest edge of an obtuse triangle. For Delaunay refinement, step 7: of (1) is the familiar Delaunay vertex insertion into the mesh, M_n. Algorithmically, the update can be accomplished by a simple insertion of V_{insert} followed by a series of edge swaps, or equivalently, as the Delaunay kernel operation of George and Borouchaki, [13], page 55, which is expressed in this reference by $M_{n+1} = M_n - Cav(V_{insert})^2 + Ball(V_{insert})^3$ To relate this operation to the basic refinement algorithm, we identify S_{rem} of step 7: of (/refalgorithm) with $Cav(V_{insert})$ and S_{add} with $Ball(V_{insert})$.

What has been the motivation for developing ADR methods from basic h-refinement methods? The initial uses of iterative Delaunay refinement seem to have been motivated by the use of the circumcentre of T for (V_{insert}). Frey,

[1]Perhaps the earliest reference to adaptive refinement based on LEBis is Sewell, 1979, [23]. See also Bank and Sherman, 1979, [4]; the edge based refinement of this reference is not exactly LEBis

[2]the cavity of V in M_n,$Cav(V)$, is the set of triangles with V in their circumcircle

[3]the ball of V in M_{n+1}, $Ball(V)$, is the set of triangles in M_{n+1} incident on V

1987, [12], promoted it on the basis that $CC(T)$ was a new vertex that was equidistant from the vertices of T, and a longer distance from any other nodes of M, and other authors have concurred, e.g. Peraire et al, 1987, [16].

Another motivation came from proofs that the Delaunay incidences minimized the interpolation error for $f(x, y) = x^2 + y^2$ in a number of error measures, over a given set of vertices, e.g. D'Azevedo and Simpson, 1989, [11] and Rippa, 1992 [17], The implication was that for isotropic errors, a maximal efficiency mesh would be a Delaunay mesh. Of itself, this is not a very convincing motivation because most data, and errors, are anisotropic. The extension of these ideas to anisotropic errors was also recognized in the applications literature e.g. Mavriplis, 1991 [14], and the mesh theory literature. Anisotropic errors are minimized by meshes that are Delaunay in appropriate stretched coordinate systems (see §5).

There are also motivations from the benefits of using Delaunay meshes for the discretization of PDEs by either the FEM or FVM that we do not discuss here, e.g. Shewchuck, 2002 [24], and Sukumar, 2003, [29].

A characteristic of Delaunay meshes is that they contain the most equiangular, and hence most equi-lateral, triangles of any vertex connectivity of a triangular mesh. This is a shape benefit that is not tied to the choice of $CC(T)$ for insertion as was noted by Baker, 1994, [2]. Rivara and Palma, 1997, [21] and [19] reported combining *Lepp* and the choice of $LEBis(T)$ for V_{insert} with Delaunay insertion. Borouchaki and George presented in the same year, [7], an edge based Delaunay refinement scheme that uses many features similar to those that we have discussed above.

Delaunay refinement research was simultaneously being carried forward by the momentum of research in computational geometry. This research extended the `Refine` predicate to include a requirement that the minimum angle in the triangles exceed a specified angle tolerance, $angTol$. I.e.

$$\texttt{Refine(T)} \equiv \texttt{True} \iff size(T) \geq sizeTol \texttt{ or } min\ angle(T) \leq angTol \tag{2}$$

Chew, 1993, [8], Ruppert, 1995, [22], and Shewchuck, 1996, [25], developed Delaunay refinement algorithms based on $V_{cand} = CC(T)$, and suitable encroachment rules, that were proven to terminate with M_N satisfying $min\ angle(M_N) \geq angTol$ for $angTol$ values up to about 30°. Methods proven to terminate satisfying this angle constraint are referred to as quality methods for mesh generation. Based on this research, Shewchuck produced the widely used quality Delaunay refinement code, Triangle, [25]. Using $size(T) = R_{cc}(T)$, $V_{cand} = CC(T)$, and a largest T first ordering for the selection in step 4, Baker, [2], gave an alternative proof that Delaunay refinement with the given encroachment rule, terminates satisfying $min\ angle(M_N) \geq 30°$.

3 ADR

In this section, we provide a detailed study of ADR using the data function

$$compexp(x, y) = (1 + e^{2\pi x} cos(2\pi y))/(2e^{2\pi}) \tag{3}$$

as the working example. Because this function is the real part of complex valued function $(1 + e^{2\pi z})/(2e^{2\pi})$, we will refer to it as the complex exponential function. $f^{(pl)}(x, y)$ is the piecewise linear interpolant of f and the error measure to be used is

$$|e|_M = \max_{T \in M} |err|_{2,A}(T) \tag{4}$$

where $|err|_{2,A}(T)$ is a computable estimate of the average L_2 error over triangle T,

$$||err||_{2,A}(T) = (\int_T (f - f^{(pl)})^2 dA/A(T))^{\frac{1}{2}} \tag{5}$$

$A(T)$ is the area of T. $|err|_{2,A}(T)$ is computed by estimating $||err||_{2,A}(T)$ using a 7 point order 4 quadrature rule for integration over T which may be found in Strang and Fix, [28], page 184. The criteria used for `Refine(T)` are those of (2) with $|err|_{2,A}(T)$ in place of $size(T)$ and similarly for $errTol$. We set $angTol = 20°$; however, the angle criterion plays very little role in this study.

We introduce several notations:

$M(D, f, errTol)$ – for a mesh generated by ADR on domain D, for data function f using error tolerance $errTol$

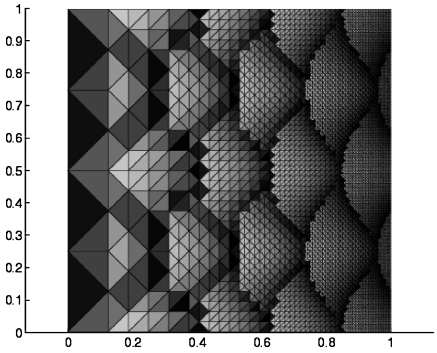
$N_V(M)$ – for the number of vertices in mesh M.

$|err|_{2,A}(T, f)$ – for $|err|_{2,A}(T)$ if we wish to be explicit about dependence on f.

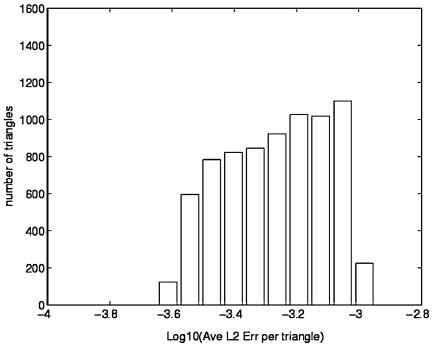
3.1 A Demonstration

For this computation, D is US, the unit square and the initial mesh, M_0 is the two triangle mesh on US. $LEBis$ is used in `SelectNewVertex`. $M(US, compexp, 10^{-3})$ is shown in Figure 2(A); the triangles of this mesh are shaded with a 10 level gray scale based on $log_{10}(|err|_{2,A}(T))$ to show the distribution of errors. A summary histogram of this error data is also shown in Figure 2(B).

It can be seen in Figure 2(A) that only one triangle shape is present in the refined mesh; all the triangles are isosceles, right angled. We will abbreviate 'isosceles, right angled triangle' to IRAT. For an IRAT, the midpoint of the longest edge coincides with the circumcircle center, i.e. $LEBis(T) = CC(T)$. In Lemma 1 below, we use this to infer that, for this M_0, the ADR methods that use either choice of V_{insert} produce the same mesh as an adaptive h-refinement method that uses simple longest edge bisection refinement. The regions of constant triangle shape and size form a series of patches with curved outlines in the mesh. The edge lengths step down by a constant factor



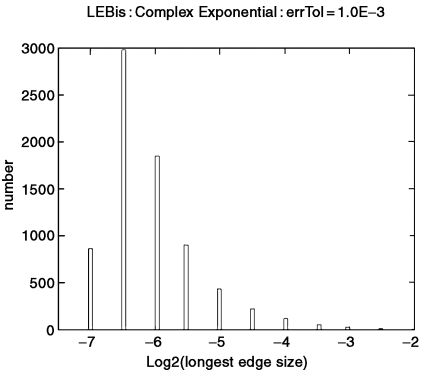
(A) $M(US, compexp, 10^{-3})$ // $N_V(M)$ = 3845 vertices



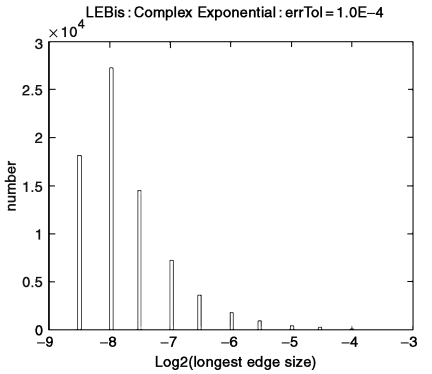
(B) histogram of $|err|_{2,A}(T, ce)$

Fig. 2. ADR mesh for complex exponential, (3), with $errTol = 10^{-3}$, : 3845 vertices

of $1/\sqrt{2}$ on moving from one patch to its neighbour on the right. This pattern can be conveniently summarized by the histogram of the distribution of the log_2(longest edge) shown in Figure 3(A). It shows a discrete spectrum of sizes at the negative half integers. This pattern persists if we decrease $errTol$. Figure 3(B) shows the same histogram for $errTol = 1.0^{-4}$; the corresponding mesh has 37520 vertices, i.e. about 10 times the number in the mesh of Figure 3(A).



(A) $errTol = 1.0^{-}$, for mesh in Figure 2(A)
3845 verticies



(B) $errTol = 1.0^{-4}$, mesh not shown
37520 vertices

Fig. 3. Histograms of $\text{Log}_2(size(T))$ for $M(US, compexp, errTol)$

The grey scale visualization of the distributed triangle interpolation errors of Figure 2(A) shows that on each patch of constant triangle shape and size, the errors are larger for the triangles nearer to the right side of the patch. At

the boundary of two patches, the neighouring triangle of the larger size are shaded dark and those of the smaller size are light indicating a discrete step in error size across this boundary. The error step is of relative size $\approx \frac{1}{2}$ since the triangle edge size change is $\frac{1}{\sqrt{(2)}}$. The gradual darkening of the error grey scale across the patches is reflected in the relatively block shaped histogram of the distribution of $log_{10}(|e|_T)$ shown in Figure 2(B).

We now show that the features of this example have some generality.

3.2 Isosceles, Right Angled Triangle Meshes

We will refer to a mesh in which each triangle is an IRAT as an IRATM. Evidently, an IRATM is a Delaunay mesh that meets the non-obtuse boundary triangle criteria for the Delaunay refinement procedure `DelRefine(M)` of §2. The next lemma shows that for a class of common Delaunay refinement methods, if M_0 is an IRATM, then all M_n of the refinement sequence are IRATMs.

Lemma 1. *Let M be an IRATM, and let T be a triangle selected using by either Lepp or largest triangle first ordering. Let V_{insert} be either $LEBis(T)$ or $CC(T)$, and let M' be the result of Delaunay insertion of V_{insert} into M. Then M' is an IRATM.*

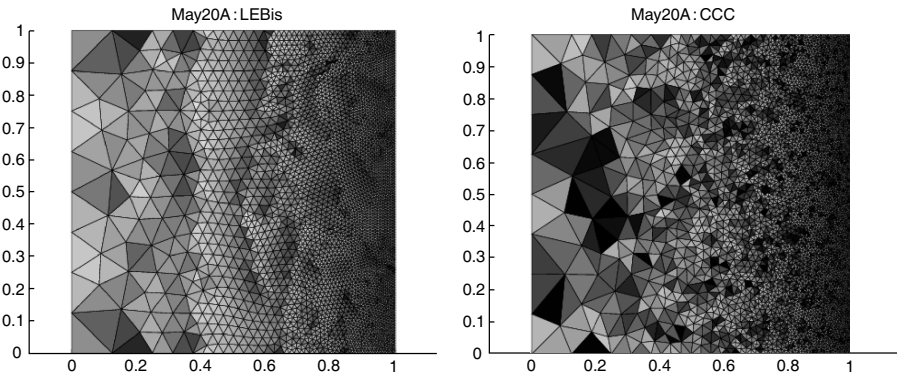
Proof If the longest edge of T is a boundary edge, then the lemma is clearly true. Assume that the longest edge of T is not on the boundary, and let $\overline{T}$ be its neighbour on this edge. Then $\overline{T}$ cannot be smaller than T. But since T has been selected by either *Lepp* or largest first ordering, $\overline{T}$ cannot be larger than T. So we conclude that $\overline{T}$ is the same size as T and the two triangles form a square. `SelectNewVertex` will choose the centre of the square for V_{cand}. V_{cand} does not encroach on any boundary edge, nor does it lie in the circumcircle of any mesh triangle other than T and $\overline{T}$. Hence $V_{insert} = V_{cand}$ and `DelaunayInsert` breaks $T \cup \overline{T}$ into 4 IRATs in M' and all other triangles in M' are unchanged from M.

Corollary 1. *Let M_0 be an IRATM, then the same mesh is obtained by the following three refinement methods applied to M_0:*

i) adaptive h-refinement with simple longest edge bisection
ii), iii) ADR with either $CC(T)$ or $LEBis(T)$ for V_{insert}

provided that either Lepp or largest triangle first ordering is used in the selection of T for refinement and the empty diametral circle encroachment rule (,or no encroachment rule,) is used.

This observation muddies the water for establishing a general merit of combining Delaunay insertion with adaptive h-refinement to get ADR, or some merits of using one of $LEBis(T)$ or $CC(T)$ in preference to the other.



(A) $LEBis(T)$: 3750 vertices (B) $CC(T)$: 3802 vertices

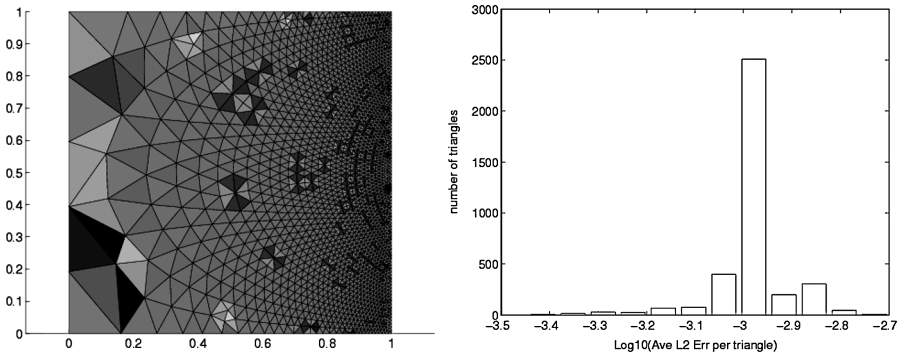
Fig. 4. ADR with strong encroachment test: $errTol = 10^{-3}$

Any such theoretical statements of merit would have to be contingent on the choice of initial mesh.

Before you try this with your favourite ADR code, note that, while mathematically correct, the proof of the lemma depends heavily on exact arithmetic. For IRATs, the circumcircle of T is the diametral circle of its longest edge. So empty circle testing and encroachment testing are basically the same. The binary outcome of the exact arithmetic encroachment test is, however, lost in floating point computation. If inexact arithmetic is used, there are in effect three outcomes:

1. the vertex is definitely not inside the diametral circle
2. it is definitely inside the circle
3. the test is not definitive.

In such arithmetic, the property that ADR maps an IRATM into a bigger IRATM will hold if we define encroachment to occur only if the vertex is definitely inside the diametral circle. We will refer to this as the weak encroachment test. For the meshes created in §3.1, we used this criterion. If, however, we change the criterion to be that encroachment takes place unless the vertex is definitely outside the diametral, circle, then we get the strong encroachment test. Two things ensue from using the strong test in `SelectNewVertex`. We do not get a sequence of IRATM meshes, and there is a difference between using the $LEBis(T)$ choice of V_{cand} and the $CC(T)$ choice. In Figure 4(A) and (B), we show the two meshes generated by the strong test and the $LEBis(T)$ choice of V_{cand} in (A) and the $CC(T)$ choice in (B). While the meshes are evidently not IRATMs, and clearly differ, the sizes are not significantly different, i.e. the variation between the meshes in Figures 2(A), 4(A), and 4(B) is less than 12%.



(A) Comparison mesh - 1938 vertices (B) $Log_{10}(|err|_{2,A}(()T)$ histogram

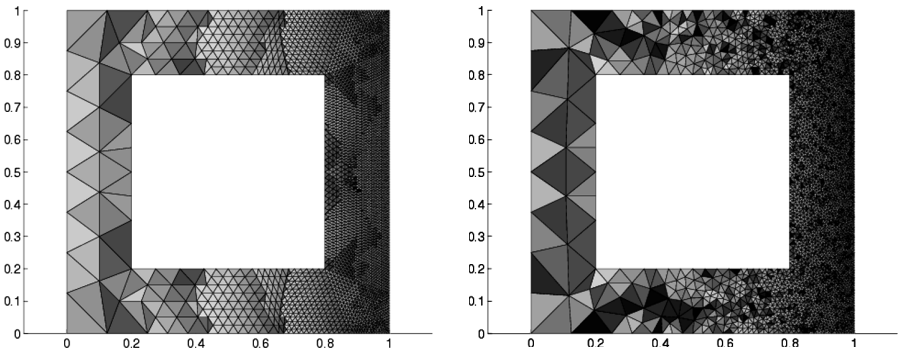
Fig. 5. Comparison mesh for complex exponential; (3)

3.3 A Comparison Mesh

In §3.1, we noted that the light shading of the error gray scale in the triangles on the right sides of the uniform mesh patches of Figure 2(A) indicate some inefficiencies in this mesh. These triangles are smaller than necessary, which also shows in the block structure of the error histogram in Figure 2(B). We can get some insight into the extent of this inefficiency by creating a comparison mesh, M_{comp} using techniques discussed in the next section that are based on some theory of mesh efficiency which is reviewed in §5. For $errTol = 10^{-3}$, M_{comp} is shown in Figure 5(A) and $NV(M_{comp}) = 1938$. A histogram of $log_{10}(|err|_{2,A}(T))$ is shown in Figure 5(B). This histogram shows that the mesh is strongly equidistributing; i.e. that $|err|_{2,A}(T) \approx errTol$ for most of the triangles in M_{comp}. This histogram also shows that M_{comp} is not, strictly speaking, a feasible mesh; i.e. it contains some triangles that do not meet the error tolerance. Meshes that are fully equidistributing to a specified error tolerance are theoretically possible. But constructing them is as difficult as constructing M_{MaxEff}. Here we propose M_{comp} as an indication that a M_{MaxEff} has about 1900 to 2000 vertices. This is evidence for our conclusion that the ADR generated meshes, $M(US, ce, 10^{-3})$ that we have shown are about twice as big as necessary.

3.4 A Modified Domain: The Hollow Square

Perhaps efficiency factor of about 2 that we observed for M_{comp} compared to meshes of Figures 2 or 4 is due to taking D as the unit square and /or M_0 as an IRATM. In this section, we look at an alternative $D = HolloSq$ which is the hollow square created by removing the square from (.2,.2) to (.8,.8) from the unit square as shown in Figure 6. M_0 is the 8 triangle mesh on HS.



(A) $LEBis(T)$: 3141 vertices (B) $CC(T)$: 3227 vertices

Fig. 6. $M_{LEBis}(HolloSq, compexp, 10^{-3})$ and $M_{CC}(HolloSq, compexp, 10^{-3})$

Figure 6 (A) shows $M_{LEBis}(HolloSq, compexp, 10^{-3})$ with 3141 vertices,, created using $LEBis(T)$ and Figure 6 (B) shows $M_{CC}(HolloSq, compexp, 10^{-3})$ with 3227 vertices, created using $CC(T)$. $M_{LEBis}(HolloSq, ce, 10^{-3})$ clearly shows the pattern of patches of regular submeshes that characterizes $M(US, compexp, 10^{-3})$ of Figure 2(A) and Figure 4(A). It seems likely that this is related to the shape stability of simple longest edge bisection studied by Adler, [1]. By comparison, $M_{CC}(HolloSq, compexp, 10^{-3})$ in Figure 6(B) shows no such patterns; it, and Figure 4(B),show relatively continuous transitions of triangle edge lengths with significantly more irregularity. These characteristics of the two vertex insertion methods were reported by Baker, [2].

Despite the difference in appearance of the the two meshes in Figure 6, they are essentially the same size i.e. neither $LEBis(T)$ nor $CC(T)$ appear to provide an efficiency advantage over the other. This effect persists for a range of $errTol$. Figure 7 shows that the sizes of $M_{LEBis}(HolloSq, compexp, errTol)$ and $M_{CC}(HolloSq, compexp, errTol)$ are essentially the same for $10^{-4.2} \leq errTol \leq 10^{-2.4}$. A least squares fit to this data produces the linear relationship $N_V(M) \approx 3.15/errTol$.

4 Creating a Comparison Mesh

We discuss the following steps for creating a comparison mesh like that of Figure 5(A).

1) Introduce an appropriate new coordinate system by a transformation

$$(u, v) = G(x, y) \tag{6}$$

which maps D in the (x, y) plane 1-1 onto a domain $\overline{D}$ in the (u, v) plane. Let the inverse transformation be $(x, y) = g(u, v)$.

2) Define data function

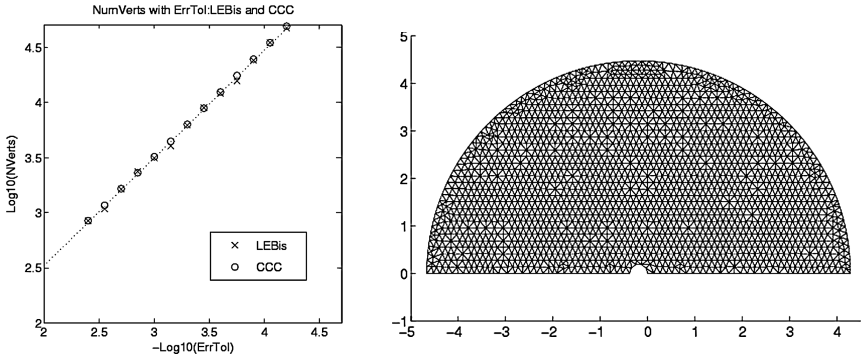
$$F(u,v) = f(g(u,v)) \;\; for (u,v) \in \overline{D} \tag{7}$$

for $(u,v) \in \overline{D}$ to be the function corresponding to f invariant under (6). Create an appropriate mesh, $\overline{M}_{comp} = M(\overline{D}, F, \overline{errTol})$

3) Using $(x,y) = g(u,v)$, map $\overline{M}_{comp}$ to M_{comp} on D.

Key to getting a suitable comparison mesh are the interpretations of 'appropriate' as it appears in steps 1) and 2) of (7). These explanations are based on some theory of maximal efficiency meshes which is, unfortunately, incomplete and technically complex, even for two dimensions. We give an overview of this theory in §5. The choices of appropriate $G(x,y)$ and $\overline{M}_{comp}$ are quite problem specific and customized; so these techniques for creating M_{comp} are not practical general mesh generation techniques.

Note that in looking for efficient meshes for pwlinear interpolation, we are not concerned with whether they are Delaunay meshes or not. While we do create a Delaunay mesh for $\overline{M}_{comp}$ in step 2, the inverse mapping of step 3 does not ensure that M_{comp} is Delaunay, nor do we care.



(A) $N_V(M(HS, ce, errTol)) = 3.15/errTol$ for either $LEBis$ or CC (B) $\bar{M}_{Comp}$, $N_V(\bar{M}) = 1938$ vertices

Fig. 7.

For the complex exponential, D'Azevedo, [9], showed that an appropriate transformation is[4]

$$\begin{aligned} u = G_1(x,y) &= \sqrt{20}/e^{\pi}\ (e^{\pi\, x} cos(\pi\, y) - 1) \\ v = G_2(x,y) &= \sqrt{20}/e^{\pi}\ e^{\pi\, x} sin(\pi\, y) \end{aligned} \tag{8}$$

[4]Actually (8) is the transformation computed by D'Azevedo rotated through 90°.

The image of the unit square in the (x, y) plane under this transformation is shown in Figure 7(B). It is a semi-ring in the upper half of the (u, v) plane, centred on $(-c, 0)$, for $c = \sqrt{20}/e^{\pi} = .1933\ldots$ and having inner radius c, and outer radius $2\sqrt{5} = 4.472\ldots$.

Figure 7(B) also shows the version of $\overline{M}_{comp}$ used as per step 2 of (7) to create M_{comp}, shown in Figure 5(A), as per step 3. To create $\overline{M}_{comp}$, we use a small amount of ADR applied to an initial Delaunay mesh, $\overline{M}_0$. $\overline{M}_0$ is constructed from a regular grid of vertices controlled by a grid spacing parameter, h. The boundary vertices form a uniformly spaced partition of $\partial\overline{D}$ of spacing $\approx h$, and the internal vertices lie on a grid $(j\,h, k\,\sqrt{5}h)$ for integers j and k corresponding to grid points inside $\overline{D}$ at a distance greater than h from the boundary. ADR is then used to reduce the size of the triangles of $\overline{M}_0$ to meet an error criterion, $\overline{errTol}$ for F, while retaining much of the desired shape in the resulting triangles. We pick h large enough that $\overline{M}_0$ requires at least every internal triangle to be refined at least once.

This example can provide some simple intuitive insight into the term 'appropriate' for G of step 1 and $\overline{M}_{comp}$ of step 2 in this case. We know that the mesh spacing in M should be small on the $x = 1$ boundary of the unit square (for accuracy) and should be large on the $x = 0$ boundary (for efficiency). If such a mesh is to be the transform of an essentially uniform mesh on $\overline{D}$, then the image of the $x = 1$ boundary must be much longer that the image of the $x = 0$ boundary. G accomplishes this by mapping the $x = 1$ boundary to the large outer circle, and the $x = 0$ boundary onto the small inner circle, of the boundary of $\overline{D}$.

As described in §5, the ideal comparison mesh would be a regular mesh of essentially uniform triangles of appropriate shape. Exact uniformity is not possible because of the geometry of $\overline{D}$ and not useful because of the approximate relation between $|err|_{2,A}(\overline{T}, F)$ and $|err|_{2,A}(T, f)$ when T and $\overline{T}$ are images under (8). Step 2 at (7) has two continuous control parameters, h and $errTol$, that can be tuned to control the size of M_{comp} and the spectrum of $|err|_{2,A}(T)$. However, the variation of these outcomes with the control parameters is only approximately continuous; there are discrete jumps in behaviour due to the 'stiffness' of essentially uniform meshes. For this reason, we have presented the mesh of Figures 5(A) and 7(B) as a comparison mesh even though it is not feasible, rather than a feasible comparison mesh with a spectrum that peaks significantly below the error tolerance of 10^{-3}.

5 An Overview of Some Theory of Maximal Efficiency Meshes

In the introduction, we describe a goal of adaptive h-refinement as producing meshes that locate the vertices efficiently for the control of the piecewise linear approximation purpose of the mesh. We also mention that there is a limit to how efficiently this can be achieved. The mathematical formulation of this

limit is a maximal efficiency mesh, M_{maxEff}, which minimizes the number of vertices over the set of meshes for which $\max_{T\in M}(||err||_{2,A}(T)) \leq errTol$. We can view ADR as a method for generating meshes, $M(D, f, errTol)$ that are feasible meshes for the constrained optimization problem of determining M_{maxEff}. In this section, we give an overview of some of the theory pertaining to M_{maxEff}. We start with the simple case of quadratic f, then discuss what can be said for more general functions, and then look at harmonic functions in particular.

5.1 The Quadratic Data Function Model

Much of our understanding of max efficiency meshes is guided by analysis of $f(x, y)$ as a quadratic polynomial, which has produced some rigorously correct results and helpful insights. Three key components of the study of optimal meshes for quadratic f are

a) an explicit formula for $||err||_{2,A}(T, f)$ that shows its dependence on the size and shape of T
b) an affine transformation to a new coordinate system that reduces the formula of a) to one of two canonical cases.
c) a characterization of the maximal efficiency triangle shapes for the two canonical cases.

The transformation of b) is then used to reduce optimal meshing problems to one of two canonical optimal meshing problems. The characterization of c) provides some indication of the nature of solutions to these problems, including some special case, non-typical solutions.

Component a): Error in Linear Approximation of Quadratic Data

Let $f(x, y) = \frac{1}{2}H_{1,1}\,x^2 + H_{1,2}\,xy + \frac{1}{2}H_{2,2}\,y^2$ where H is the constant Hessian of f. In developing a formula for $||err||_{2,A}(T, f)$ for this quadratic f, Nadler [15], introduces quantities, d_j, associated with the jth edge of T which can be expressed in native coordinates, (x, y), by the quadratic form

$$d_{j+1} = \frac{1}{2}(P_{j+1} - P_j, H(P_{j+1} - P_j)) \text{ for } P_j = (x_j, y_j) \tag{9}$$

The error is then given by

$$||err||_{2,A}(T, f) = \frac{1}{180}((d_1 + d_2 + d_3)^2 + d_1^2 + d_2^2 + d_3^2). \tag{10}$$

See also Berzins, [6]. Note from (10) that d_j has units of error; so they are invariant under affine changes of coordinates.

Component b): Canonical Forms

We now develop an affine transformation from native coordinates, (x, y) to coordinates, (u, v) in which the d_j have a canonical form. These new coordinates are commonly called stretched coordinates. Because H is symmetric, there is an orthogonal matrix, M such that $M^t\, H\, M$ is diagonal, D; the diagonal entries of D, D_k $k = 1, 2$ being the eigenvalues of H. Consequently, we can rewrite (9) as

$$d_{j+1} = \frac{1}{2}(M\ (P_{j+1} - P_j), D\ M\ (P_{j+1} - P_j))$$

$M\, P_j$ are rotated coordinates of P_j which we will denote by (p_j, q_j). Assuming that $|D_1| \leq D_2$, we can then write

$$d_{j+1} = \frac{1}{2} D_2\ (a_2(p_{j+1} - p_j)^2\ +\ (q_{j+1} - q_j)^2) \tag{11}$$

where $a_2 = D_1/D_2$, so $|a_2| \leq 1$. For simplicity, we assume $a_2 \neq 0$, avoiding the degenerate case.

The transformation to stretched coordinates takes the form

$$\begin{pmatrix} u \\ v \end{pmatrix} = \sqrt{D_2} \begin{pmatrix} \sqrt{|a_2|} & 0 \\ 0 & 1 \end{pmatrix} M \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} G_1(x, y) \\ G_2(x, y) \end{pmatrix}. \tag{12}$$

It is useful to introduce the anisotropy ratio, a, which carries the sign of a_2, as $a\ = \text{sign}(a_2)\ \sqrt{|a_2|}$ Under this transformation, we have the following forms for d_j

$$\begin{array}{ll} \text{if } a > 0\ d_{j+1} = \frac{1}{2}(\ (u_{j+1} - u_j)^2 + (v_{j+1} - v_j)^2\ ) & \text{the definite case} \\ \text{if } a < 0\ d_{j+1} = \frac{1}{2}(\ -(u_{j+1} - u_j)^2 + (v_{j+1} - v_j)^2\ ) & \text{the indefinite case} \end{array} \tag{13}$$

Stretched coordinates are useful here because the maximal efficiency meshing problem is invariant under (12) in the sense that a solution in one coordinate system transforms to a solution in the other. Let us note the details involved in this assertion. If we continue to use $F(u, v) = f(g(u, v))$ as introduced in §3.3, we can regard functions (f, D) and $(F, \overline{D})$ as one abstract quadratic function, invariant under (12). However, by virtue of (13), we can see that from the point of view of error behaviour, we can assume

$$\begin{aligned} F(u, v) &= (u^2 + v^2)/2 \ \text{ if } a > 0 \\ &= (-u^2 + v^2)/2 \ \text{ if } a < 0 \end{aligned} \tag{14}$$

which we will denote by $F(u, v) = (\pm u^2 + v^2)/2$.

We can extend the mapping between D and $\overline{D}$ to be a mapping of an arbitrary mesh M on D to mesh $\overline{M}$ on $\overline{D}$ in which each triangle $T \in M$ is mapped exactly to a triangle $\overline{T} \in \overline{M}$. In this sense, we could say that M and $\overline{M}$ are invariant under (12). Consequently, the piecewise linear function

$F^{(pl)}(u,v)$ can be defined either as the piecewise linear interpolant of F on $\overline{M}$, or the transform of $f^{(pl)}(x,y)$; they are equivalent. Using this invariance of $f^{(pl)}(x,y)$ and the invariance of the d_k terms of Nadler's formula, we can see that $||err||_{2,A}(T,f)$ is invariant, i.e. can be calculated in either coordinate system.

Consequently, we have the following equivalences

$$\begin{aligned} M(D,f,errTol) &\iff M(\overline{D},F,errTol) \\ M_{maxEff}) &\iff \overline{M}_{maxEff} \end{aligned} \tag{15}$$

meaning that if meshes M and $\overline{M}$ are images under (12), then M meets the error tolerance, $errTol$ if, and only if, $\overline{M}$ does, and M is a maximal effiency mesh if, and only if, $\overline{M}$ is.

Component c): Maximal Efficiency Triangles

A triangle is a maximal efficiency triangle, T_{maxEff}, if it maximizes $A(T)$ over the set of triangles such that $||err||_{2,A}(T) \leq errTol$. The maximal efficiency triangle for a general quadratic f is the transform of its canonical case. Both D'Azevedo, [9], and Nadler [15], identified the shapes of maximal efficiency triangles for the canonical cases. For the definite case, the maximal efficiency triangles are equilateral with any orientation. For the indefinite case, there are several isosceles shapes for T_{maxEff}; however, their optimality is orientation dependent, which complicates the construction of $\overline{M}_{maxEff}$ for the indefinite case. One of these shapes is the isosceles triangle with horizonal base and height to base ratio $\sqrt{5}/2$.

If, per chance, domain $\overline{D}$ could be tiled with maximal efficiency triangles for $errTol$, then the resulting mesh would be $\overline{M}_{maxEff}$ for $\overline{D}$. Then, in turn, the mesh obtained by transforming $\overline{M}_{maxEff}$ to the domain D would be M_{maxEff} for $(D,f,errTol)$.

5.2 More General Data Functions

If $f(x,y)$ is not quadratic, then, assuming that f is smooth, it can be approximated near P by a quadratic based on the Hessian, $H_f(P)$. It is common in meshing algorithms to use the quadratic f theory locally, i.e. for triangles that are small enough to qualify as being 'near P'. However, this is of limited value in the global view of meshing needed for the theory of maximal efficiency meshes. So we turn to some classical differential geometry for the tools we need.

If the line segment from P_j to P_{j+1} is considered to be infinitely short, i.e. by setting $P_{j+1} = P_j + dP$ for differential dP, then, in native coordinates, (9) becomes the differential edge length formula

$$ds^2 = (dx, H_f(P)\,dx) \tag{16}$$

In the case that $H_f(P)$ is positive definite, (16) is the Riemannian geometry formula for differential arc length for metric tensor H_f. For the case in which $H_f(P)$ is indefinite, (16) is the formula for differential arc length in a Minkowski, or hyperbolic, geometry. Classical differential geometry has established that for particular classes of metric tensors, it is possible to define stretched coordinates globally for the domain by transformations $(u, v) = G(x, y)$. The differential edge length formulae in the (u, v) coordinates are either Euclidean, i.e. $ds^2 = du^2 + dv^2$, or canonical hyperbolic, i.e. $ds^2 = -du^2 + dv^2$, Sokolnikoff, [27], Chapter 2. These are the same two canonical cases as we identified at (13).

In [9], D'Azevedo determined sufficient conditions on H_f for a global transformation to stretched coordinates to exist, and describes a procedure for constructing $G(x, y)$. These conditions are met, in particular, by harmonic functions and the procedure applied to the complex exponential data function, (3) results in essentially the transformation (8).

Since these transformations to stretched coordinates are not generally affine, triangles in stretched coordinates are mapped onto 'curved triangles' i.e. three sided patches connecting the images of the vertices with curved sides. In particular, $||err||_{2,A}(T, f)$ is not invariant. Consequently, the problems of determining M_{maxEff} for D, f and $errTol$ and $\overline{M}_{maxEff}$ for $\overline{D}, F$ and $errTol$ no longer enjoy the equivalence that we noted at (15) for quadratic data functions. It is not clear whether such equivalences would hold in some asymptotic sense as $errTol \to 0$. We have observed in calculations that the ratio $|err|_{2,A}(\overline{T}, F)/|err|_{2,A}(T, f) \approx .75$ as T becomes small. This suggests that $|err|_{2,A}(\overline{T}, F)$ may not be even a consistent approximation to $|err|_{2,A}(T, f)$ for small T. These observations have implications for the construction of comparison meshes that we present in §4. We introduce a separate error tolerance for tuning the comparison mesh and we report the error statistics for M_{comp} in native coordinates, (x, y).

5.3 Application to Harmonic Functions

For harmonic data functions $f_{yy}(x, y) = -f_{xx}(x, y)$ so H_f has the special form

$$H_f = \begin{pmatrix} r & s \\ s & -r \end{pmatrix} \tag{17}$$

for $r = f_{xx}(x, y)$, $s = f_{xy}(x, y)$. The eigenvalues of (17) are $D_2 = \sqrt{r^2 + s^2}$, $D_1 = -D_2$; hence, non-degenerate $H_f(x, y)$ is always indefinite. The anisotropy ratio, a, of (11) is exactly -1, so, in this sense, the error metric is isotropic. However, because H_f is always indefinite, the error will show some directional dependence and maximal efficiency triangles have orientation restrictions and are not equilateral as discussed in §5.1.

The D'Azevedo global transformation to stretched coordinates can be computed for harmonic functions, and, in particular,for the complex exponential,

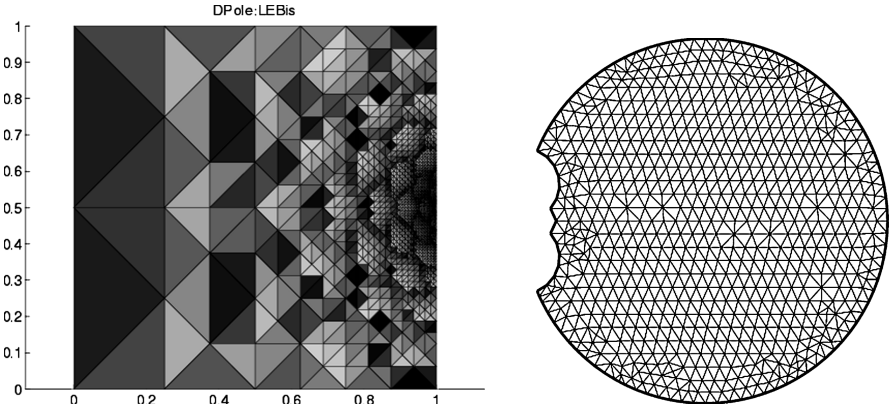
(3) as per (8) which was used to construct the image domain, $\overline{D}$ in the (u, v) plane shown in Figure 7(B). Using ADR, $\overline{M} = M(\overline{D}, (-u^2 + v^2)/2, \overline{errTol})$ was created as described in §4. The initial mesh for $\overline{M}$ had a regular interior grid which, if refined once, would be composed of maximal efficiency triangles for the indefinite case in stretched coordinates.

6 The Double Pole: A Second Example Data Function

If we look at the mesh characteristics and efficiency of ADR applied to another harmonic data function, we find that our observations of §3 for the complex exponential are confirmed. In this section, we demonstrate this using the 'double pole' data funtion

$$dp(x, y) = Re(1/(z-c)^2) = ((x-c_1)^2+(y-c_2)^2)/((x-c_1)^2+(y-c_2)^2)^2 \quad (18)$$

on the unit square, for pole at $c = (1.1, .5)$.



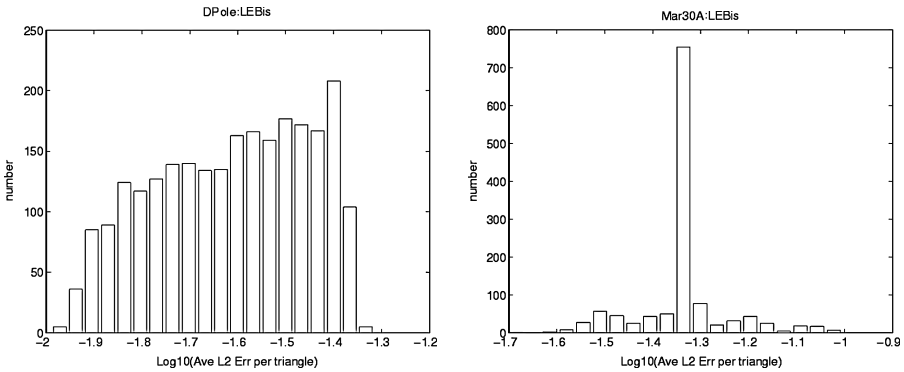
(A) $M(US, dp, .0045)$: 1281 vertices (B) $\overline{M}_{comp}$: 677 vertices

Fig. 8. ADR and comparison meshes for double pole data function

For a two triangle initial mesh, with $errTol = .0045$, and the weak encroachment criterion, all methods produce the IRATM shown in Figure 8(A) The D'Azevedo transformation for a double pole at (c_1, c_2) is

$$u = \sqrt{6}(1-(x-c_1))/d^2 \;\; ; v = \sqrt{6}(y-c_2)/d^2 \;\; \text{for } d^2 = (x-c_1)^2+(y-c_2)^2 \quad (19)$$

The domain, $\overline{D}$ in the (u, v) plane corresponding to D= the unit square is shown in Figure 8(B). It is bounded by arcs from four circles. These circles have centers denoted (uc_k, vc_k) and radii denote by r_k for $k = 1 \ldots 4$ in the following table



(A) IRATM : histogram of $|err|_{2,A}(T)$ (B) M_{comp} : histogram of $|err|_{2,A}(T)$

Fig. 9. Error spectra for adaptive and comparison meshes for double pole data function

$$(uc_1, vc_1) = s(1, -1/(2 * c_2)) \; ; \; r_1 = s/(2c_2)$$
$$(uc_2, vc_2) = s(1 + 1/(2(c_1 - 1)) \; ; \; r_2 = s/(2|c_1 - 1|)$$
$$(uc_3, vc_3) = s(1, 1/(2 * c_2) \; ; \; r_3 = s/(2c_2)$$
$$(uc_4, vc_4) = s(1 + 1/(2c_1) \; ; \; r_4 = s/(2c_1)$$

where $s = \sqrt{6}$.

The error histograms for $M(US, dp, .0045)$ of Figure 8(A)) and M_{comp}, transformed from Figure 8(B), are shown in Figures 9(A) and (B) respectively.

7 Observations and Conclusions

There are several observations that we have made that are consistent across the computations that we have described, and others that we performed in the course of this study. A primary objective was to estimate the efficiency of ADR applied to isotropic data functions and using Euclidean geometry Delaunay meshes. We observed that the meshes are typically about twice the size of maximal efficiency meshes for these cases. It seems reasonable to us that a similar efficiency would be obtained by ADR for anisotropic data, using appropriate stretched coordinates.

Our review of §2 identified several versions of adaptive refinement and in particular, we reported on computations using the alternative choices of insertion vertex, $LEBis(T)$ and $CC(T)$. In §3, we noted that the different versions all produce that same resulting mesh, if the initial mesh is an IRATM, and if the implementations in inexact arithmetic use a weak encroachment criterion. The relevance of this observation for our efficiency study is that any efficiency differences between the versions must come from features associated with the initial mesh. If the initial mesh is not an IRATM, or a strong encroachment

criterion is used, then difference in the resulting meshes were quite apparent, but differences in efficiency were insignificant, (Figures 4, 6).

The form of the error histograms for the ADR meshes provide some insight into the nature of their efficiency. Figures 2(B) (complex exponential) and 9(B) both show that $log_{10}(errTol) - .6 \leq log_{10}(|err|_{2,A}(T)) \leq log_{10}(errTol)$ and $10.^{-.6} \approx 1/4$. I.e.

$$errTol/4 \ \leq \ |err|_{2,A}(T) \ \leq errTol$$

The lower bound of $errTol/4$ for $|err|_{2,A}(T)$ was universally observed in our computations, and the distribution of $|err|_{2,A}(T)$ in the interval ($errTol/4$, $errTol$) is typically fairly uniform as shown in Figures 2(B) and 9(B). So the average error is about $5errTol/8$, which is roughly consistent with the mesh having twice as many triangles as would be needed to meet the error criterion of $errTol$.

Perhaps this amount of inefficiency in ADR meshes is acceptable for many purposes. But if we want to achieve more efficiency, where should we look? Perhaps some form of smoothing could focus the error histogram distribution at its average value enough to be a significant remedy. However, it does not seem obvious how this would be achieved, and at what price. Note that D'Azevedo, [10], demonstrated a similar level of efficiency in the meshes generated by PLTMG, which incorporated a smoothing.

We have presented efficiency of ADR for refinement based on the L_2 average error, $||err||_{2,A}(T)$. It would be interesting to know the extent to which our observations would carry over to an energy average norm, i.e. $(\int_T (\partial err/\partial x)^2 + (\partial err/\partial y)^2 dA/\ A)^{1/2}$. It would also be very interesting to know about the efficiency of ADR for three dimensions. However, there are substantial difficulties in exending the technique of this paper to 3-D. In particular, neither the theory of maximal efficiency triangles for quadratic data nor techniques for computing a global transform to stretched coordinates are known.

References

1. A. Adler. On the bisection method for triangles. *Mathematics of Computation*, 40:571–574, 1983. similarity classes.
2. T. J. Baker. Triangulations, mesh generation and point placement strategies. In D. Caughey, editor, *Computing the Future*, pages 61–75. John Wiley, 1994.
3. R. E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0.* SIAM, Philadelphia, 1998.
4. R. E. Bank and A. H. Sherman. The use of adaptive grid refinement for badly behaved elliptic partial differential equations. In R. Vichnevetsky and R. S. Stepleman, editors, *Advances in Computer Methods for Partial Differential Equations-III*, pages 33–39. IMACS, 1979.
5. R. E. Bank and A. Weiser. Some a posteriori error estimators for elliptic partial differential equations. *Math Comp*, 44(170):283–301, 1985.

6. M. Berzins. A solution-based triangular and tetrahedral mesh quality indicator. *SIAM J. Sci. Stat. Comput.*, 19:2051–2060, 1998.
7. H. Borouchaki and P. L. George. Aspects of 2-d delaunay mesh generation. *International Journal for Numerical Methods in Engineering*, 40:1957–1975, 1997.
8. L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th Annual Comp. Geometry*. ACM Press, 1993. QA448.D38S87x.
9. E. F. D'Azevedo. Optimal triangular mesh generation by coordinate transformation. *SIAM J. Sci. Stat. Comput.*, 12:755–786, 1991.
10. E. F. D'Azevedo. On adaptive mesh generation in two-dimensions. In S Owen, editor, *Proceedings: 8th International Meshing Round Table*. Sandia National Laboratories, 1999.
11. E. F. D'Azevedo and R. B. Simpson. On optimal interpolation triangle incidences. *SIAM J. Sci. Stat. Comput.*, 10:1063–1075, 1989.
12. W. H. Frey. Selective refinement: A new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24:2183–2200, 1987.
13. P. L. George and H. Borouchaki. *Delaunay Triangulation and Meshing*. Hermes, 1998.
14. D. J. Mavriplis. Adaptive mesh generation for viscous flows using Delaunay triangulation. *Journal of Computational Physics*, 90:271–291, 1990.
15. E. Nadler. *Piecewise Linear Approximation on Triangles of a Planar Region*. PhD thesis, Brown University, 1985. order Number DA8519885.
16. J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72, 1987.
17. S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Analysis*, 9:257–270, 1992.
18. M.-C. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International Journal for Numerical Methods in Engineering*, 20:745–756, 1984.
19. M. C. Rivara. New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. *Int. J. Num. Methods*, 40:3313–3324, 1997.
20. M. C. Rivara and N. Hitschfeld. LEPP-Delaunay algorithm: a robust tool for producing size-optimal quality triangulations. In *Proc. of the 8th Int. Meshing Roundtable*, pages 205–220, October 1999.
21. M.-C. Rivara and M. Palma. New LEPP-Algorithms for quality polygon and volume triangulation: Implementation issues and proctical behavior. In *Trends in Unstructured Mesh Generation*, volume AMD-Vol. 220, pages 1–8. American Society of Meshanical Engineers, 1997. The Joint ASME/ASCE/SES Summer Meeting, Evanston, Illinois, USA, July 1997.
22. J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensi onal mesh generation. *J of Algorithms*, 18:548–585, 1995.
23. G. Sewell. A finite element program with automatic user-controlled mesh grading. In R. Vichnevetsky and R. S. Stepleman, editors, *Advances in Computer Methods for Partial Differential Equations- III*, pages 8–10. IMACS, 1979.
24. J. R. Shewchuck. What is a good linear element? interpolation, conditioning, and quality measures. In S. Owen, editor, *Proceedings: 11th International Meshing Round Table*. Sandia National Laboratories, 2002.
25. J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In ACM, editor, *First Workshop on Applied Computational Geometry*, pages 124–133. (Philadelphia, Pennsylvania), 1996.

26. R. B. Simpson. Anisotropic mesh transformations and optimal error control. *Applied Num. Math.*, 14:183–198, 1994.
27. I. S. Sokolnikoff. *Tensor Analysis. Theory and Applications to Geometry and echanics of Continua,*. Wiley, 2nd edition, 1964.
28. G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*. Prentice Hall, 1973.
29. N. Sukumar. Voronoi cell finite difference method for the diffusion operator on arbitrary unstructured grids. *International Journal for Numerical Methods in Engineering*, 57:1–34, 2003. Laplacian.