INTERPOLATION OF SURFACES OVER SCATTERED DATA

Gonzalo A. Ramos
Computer Science Department.
University of Toronto
ramosg@acm.org

Wayne Enright
Computer Science Department.
University of Toronto
enright@cs.toronto.edu

ABSTRACT
We investigate the performance of DEI, an approach [2] that computes off-mesh approximations of PDE solutions, and can also be used as a technique for scattered data interpolation and surface representation. For the general case of unstructured meshes, we found it necessary to modify the original DEI. The resulting method, ADEI, adjusts the parameter of the interpolant, obtaining better performance. Finally, we measure ADEI's performance using different sets of scattered data and test functions and compare ADEI against two methods from the collection of ACM algorithms: Algorithms 752 [10] and 790 [11]. The results show that ADEI is better than, if not comparable to, the best of the compared scattered data interpolation techniques.

KEY WORDS
Scattered Data, Interpolation, Data Visualization

1. Introduction.

Raw numerical data usually originates from a continuous domain, but only a finite number of uniform or non-uniform samples is available. When this data is collected in an irregular fashion, it is said to be scattered, irregular, or random. Scattered data can be found in a number of problems in fields such as earth sciences, meteorology, engineering, and medicine.

For many scientific visualization systems it is desirable to have the input data defined over a regular grid. These systems can use interpolation schemes to generate uniform grid data in cases when scattered data needs to be displayed.

2. Scattered data Interpolation

The scattered data interpolation problem can be defined as: Given a set of n irregularly distributed points

$$P_i = (x_i, y_i)\ \ ,\ i = 1, ..., n \tag{1}$$

over $\Re^2$, and scalar values F_i associated with each point satisfying $F_i = F(x_i, y_i)$ for some underlying function $F(x, y)$, look for an interpolating function $\overline{F} \approx F(x, y)$ such that for $i = 1, \ldots, n$

$$\overline{F}(x_i, y_i) = F_i \tag{2}$$

We assume that all the points P_i (also referred to as nodes or mesh points) are distinct, and that all the points are not collinear. This formulation can be generalized to higher dimensions but, for the remainder of this paper, we will concentrate on the two-dimensional case.

3. Previous Work

The subject of scattered data interpolation is extensive. This section presents an overview of some techniques that are widely used in current methods. Emphasis will be on the techniques and principles behind the two ACM algorithms used later for comparisons.

The *Shepard's Method* is one of the earliest techniques used to generate interpolants for scattered data. It defines an interpolating function that is the weighted average of the value at the mesh points. This method is global because the evaluation of the interpolant requires the evaluation of a function on all given mesh points. Global techniques are expensive for large number of mesh points, but it is possible to apply them to overlapping subsets, and blend the solutions into a single interpolant for the whole set. There are variants to the technique, such as the *Modified Shepard's Method*, which modifies the weighting function to take into account only the points lying in a disc with radius R, centered at the point at which the interpolant is evaluated. The modification improves the method both in addressing shape preservation and in making it local to a neighborhood of points [5] [7]. ACM algorithm 790 (CSHEP2D) [11], used later in Section 6, is a variation of the modified Shepard's method.

Another important method of interpolation is known as *Hardy's Multiquadrics* [6]. In this global method, we consider an interpolating function that is a combination of basis functions. A popular choice of these are the radial basis functions. Franke, in [4], reports the superior performance of multiquadrics compared with other methods.

Another way to construct an interpolant is to consider it to be a piecewise union of patches (usually low degree multivariate polynomials) joined with certain continuity. Examples of this approach are those based on Spline and Bezier patches [1][3], which are extensively used in the area of geometric design and give a user freedom to model and change the shape of an object. ACM Algorithm 752 [13], used later in Section 6, is based on this approach with cubic triangular patches joined with C^1 continuity.

4. The Differential Equation Interpolant

For most Partial Differential Equations (PDEs) it is common to apply a numerical method that produces an approximate solution at certain discrete points P_i in the domain of interest, which are distributed over a non-uniform triangular mesh that is adapted to the local behavior of the solution. In [2], Enright develops an approach, called Differential Equation Interpolant (DEI), that efficiently approximates the values of the solution of a PDE at off-mesh points. The approach is such that the precision at the resulting off-mesh points has the same order of accuracy as that obtained by the underlying numerical method for the discrete mesh points. The resulting interpolation scheme assumes that the underlying solution u satisfies, for example, a known two-dimensional, second order PDE of the form

$$Lu = g(x, y, u, u_x, u_y) \tag{3}$$

where L is a differential operator defined by

$$Lu = a_1(x, y)u_{xx} + a_2(x, y)u_{yy} + a_3(x, y)u_{xy} \tag{4}$$

where a_1, a_2 and a_3 can be any arbitrary nonlinear function. The approach also assumes that an accurate approximation to the gradient of u at the points $P_i = (x_i, y_i)$ is determined by the numerical method. Such quasi-linear PDEs arise in many application areas in science and engineering (see [2] for a more detailed discussion of these problems) The interpolant is defined as a piecewise surface, where each triangle element e in the domain plane has an associated bivariate polynomial

$$p_{d,e}(x, y) = \sum_{i=0}^{d} \sum_{j=0}^{d} c_{ij} s^i t^j \tag{5}$$

The variables s and t correspond to a mapping of x and y into a unit square where the domain triangle is contained. The mapping is defined by $s = \frac{(x-x_0)}{D_1}$, $t = \frac{(y-y_0)}{D_2}$, where the vertex $P_0 = (x_0, y_0)$ denotes the origin of the unit square $s = 0$, $t = 0$, and D_1, D_2 are appropriate scaling factors that depend on the size of the patch e.

To characterize $p_{d,e}$ we need to find the coefficients c_{ij}; for cubic patches ($d = 3$), we need to find 16 coefficients. We do this by specifying 16 linear equations associated with the coefficients c_{ij}. We determine the first nine equations by imposing the following linear constraints at each of the three points (x_k, y_k) defining e:

$$\begin{aligned} \sum_{i=0}^{3} \sum_{j=0}^{3} c_{ij} s_k^i t_k^j &= F_k \\ \sum_{i=0}^{3} \sum_{j=0}^{3} i c_{ij} s_k^{i-1} t_k^j &= D_1 \frac{\partial F_k}{\partial x} \\ \sum_{i=0}^{3} \sum_{j=0}^{3} j c_{ij} s_k^i t_k^{j-1} &= D_2 \frac{\partial F_k}{\partial y} \end{aligned} \tag{6}$$

where $s_k = \frac{(x_k - x_0)}{D_1}$, $t_k = \frac{(y_k - y_0)}{D_2}$. We construct the remaining seven linear equations by choosing seven 'collocation points' inside the triangle, and by imposing a condition that the interpolant 'almost' satisfy Equation (3) at the chosen points. Using (5) we see that for a fixed value of x and y, $Lp_{3,e}(x, y)$ is a linear combination of the coefficients c_{ij}. Then, for a collocation point $(\widehat{x}, \widehat{y})$ we impose a linear constraint of the form

$$Lp_{3,e}(\widehat{x}, \widehat{y}) = g(\widehat{x}, \widehat{y}, \widehat{F}, \widehat{F_x}, \widehat{F_y}) \tag{7}$$

where $\widehat{F}$, $\widehat{F_x}$ and $\widehat{F_y}$ are approximations to the actual values of F, $\frac{\partial F(x,y)}{\partial x}$ and $\frac{\partial F(x,y)}{\partial y}$ respectively. To determine the coefficients c_{ij}, we use the system of linear equations formed by nine linear interpolation constraints, (6), and linear constraints corresponding to seven collocation points, (7). This system can be expressed as

$$Wc = b \tag{8}$$

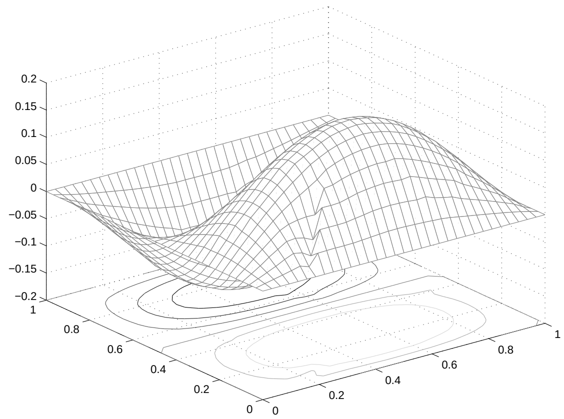
where $c = \{c_{ij}\}$, and W is a matrix that depends on the mesh points, the collocation points, and the definition of L. The vector b depends on the mesh data and the approximations $\widehat{F}$, $\widehat{F_x}$ and $\widehat{F_y}$ associated with the collocation points. To ensure that there is a solution the collocation points must be chosen such that W is nonsingular.

5. Enter: ADEI

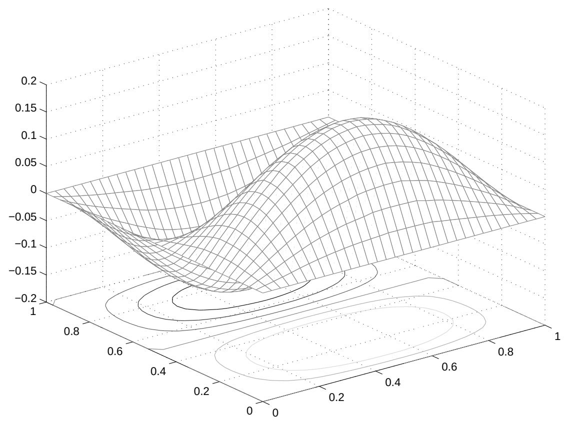
Although DEI has an accuracy comparable to the numerical method that provided the data, it does not necessarily generate a continuous piecewise interpolant. For a choice of collocation points, this can be observed in the form of oscillations in the approximation near the boundaries of the triangles e, due to the fact that on the boundaries, the function $p_{d,e}$ may well be a high degree polynomial. For the purposes of visualization, we would like to obtain smooth surfaces, so it is desirable to attenuate the discontinuities between patches as much as possible. To achieve this, we develop a test that rejects choices of collocation points that produce noticeable discontinuities among patches. Analyzing several examples, we observe that patches with non-negligible discontinuities have associated coefficients, c_{ij}, with a magnitude larger than expected. This behavior is reflected in the norm of the vector c associated with the patch (8). The magnitude of the norm is directly related to the value of $norm(c)$:

$$norm(c) = \sum_{i,j \in [0,3]} (c_{ij})^2 / 16 \tag{9}$$

We observed that the noticeable discontinuities correspond to patches with large values of $norm(c)$. To attenuate the discontinuities, we developed an algorithm that detects the offending patches and decreases their associated $norm(c)$ by choosing alternate collocation points, and therefore a different vector c. This algorithm is described in Algorithm 1.



a



b

Figure 1. DEI and ADEI for a 6x6 regular mesh; a- the interpolant obtained with the first choice of collocation points, where discontinuities can be seen; b- the corresponding interpolant after the application of Algorithm 1.

The choice of the values α and $MAXTRIES$ affects the performance of the algorithm, both in how long it takes to find a suitable set of collocation points, and how much the discontinuities are attenuated. Although the best choices for these values will depend on the function, F, and the location of the scattered data, we observed that overall good results were obtained for $\alpha = 1.5$ and $MAXTRIES = 100$. We call the interpolant defined with this set of alternate collocation points the Alternate Differential Equation Interpolant (ADEI). The result of applying the algorithm to the data in Figure 1-a can be seen in Figure 1-b.

6. The Test: Functions and Data.

We compare ADEI against two ACM algorithms for the interpolation of scattered data, algorithms 752 (acm752) and 790 (acm790). The tests were performed on three functions: F_1 from [2], F_2 from the test suite used by Renka in [12], and F_3, which serves to evaluate the performance of the methods on functions without significant features. To make the comparison as fair as possible, we provide acm752 the gradients at the mesh points in the same way that they were provided to ADEI. Because the routines in acm790 do not allow a direct input of the gradients, they were not used with this algorithm.

Algorithm 1 Algorithm to find a set of coefficients c_{ij} that will attenuate the oscillations in the boundaries of the triangles e.

for every e **do**
  Compute an initial set $c = \{c_{ij}\}$
end for
$success = true$
loop
  Compute the norm vector: $nc(e) = norm(c)$
  Compute threshold: $thresh = \alpha \cdot mean_{90\%}(nc)$
  Compute set of offending triangles: $T = \{e : nc(e) > thresh\}$
  if $T = \emptyset$ or $\neg success$ **then**
    FINISH
  end if
  $success = false$
  for each $e \in T$ **do**
    $times = 0$
    $PartialSuccess = false$
    repeat
      $times = times + 1$
      Choose alternate set of collocation points for e
      Compute $\widetilde{c} = \{\widetilde{c_{ij}}\}$
      if $norm(\widetilde{c}) < nc(e)$ **then**
        Update coefficient with better values: $c = \widetilde{c}$
        $success = true$
        $PartialSuccess = true$
      end if
    until $PartialSuccess$ or $times = MAXTRIES$
  end for
end loop

The test functions are:

$$F_1(x, y) = cos(10y) + sin(10(x - y)) \tag{10}$$

$$F_2(x, y) = e^{(-\frac{(5-10x)^2}{2})} + 0.75e^{(-\frac{(5-10y)^2}{2})} + 0.75e^{(-\frac{(5-10x)^2}{2})}e^{(-\frac{(5-10y)^2}{2})} \tag{11}$$

$$F_3(x, y) = sin(2\pi y) * sin(\pi x) \tag{12}$$

For ADEI, we must introduce a differential operator, L, for each test function. We used $Lu = u_{xx} + u_{yy}$, and the corresponding functions g_i such that $LF_i = g_i(x, y, u, u_x, u_y)$.

The data is divided into three node sets: adaptive random samples, truly random samples, and regular samples. The adaptively random class (Node Set I) has for every test function two data sets of 100 points, referred to as Data Sets I and II. The truly random class (Node Set II) consists of

Table 1. The parameter r^2 can be seen as being obtained from a least-squares fit from a constant function to the data. The interpretation associated with its value is summarized in this table.

r^2	INTERPRETATION
0	no accuracy
0.9	fair fit
0.95	good fit
0.99	very good fit
0.999	excellent fit
0.9999	almost perfect fit (negligible error in empirical data)

three sets of 100 randomly chosen points, referred to in this paper as Data Sets III, IV, and V. The third class (Node Set III) is a regular grid that divides the unit square into an 8 by 8 subdivision, referred to as Data Set VI. Also, we group the functions into two classes: Those with many features (F_1 and F_2), and those without many features (F_3).

All random points are generated in the square [-0.1,1.1]×[-0.1,1.1], making most of the thin triangles of the associated triangulation lie outside the unit square [0,1]×[0,1], where all the error measurements are taken.

We apply the error measure used by Renka and Brown in [12] to compare a function u and its approximation $\widetilde{u}$. It is defined by

$$\begin{aligned} error(u,\widetilde{u}) &= \frac{\sum_{p\in NodeSet}(u(p)-\widetilde{u}(p))^2}{\sum_{p\in NodeSet}(u(p)-\overline{u})^2} \\ &= \frac{SSE}{SSM} \end{aligned} \tag{13}$$

where $NodeSet$ are the points from a 33 by 33 uniform grid defined on the unit square, that lie in the convex hull defined by a Node Set (I, II, or III).

We summarize the results given by the error measures computing the values $r^2 = 1 - Avg(error)$, which is called the *coefficient of determination*. Its interpretation [12] is summarized in Table 1.

7. Results

We compute[1] for a given a set of values of r^2 the *expected fit* for a particular interpolation method. Table 2 shows how we can categorize by numbers the magnitude r^2. For a given set of values r^2, it is possible to count how many of them belong to each *fitness level* (interval), and to obtain the relative frequency $rf(i)$ for a particular fitness level i. We define the *expected fit* as

$$ExpectedFit = \sum_{i=0,5} rf(i)i \tag{14}$$

For the functions with many features (F_1 and F_2), ADEI has better accuracy than acm752 (even with the supplied gradients) and acm790 for Node Sets I, II, IV, and V. For Node Sets III and VI, ADEI is comparable to acm752. Figure 2-a shows the expected fit of the methods for the functions with many features. For function F_3, all the methods exhibit an excellent to almost perfect fit. Again, the two methods with higher accuracy in this case are ADEI and acm752. Figure 2-b shows the expected fit of the methods for F_3.

[1]Detailed tables for values of r^2 can be found in [8]

Table 2. Fitness levels intervals. A value r^2 belongs to one of these five categories.

INTERVAL	FITNESS LEVEL	FIT
0-0.9	0	No
0.9-0.95	1	Fair
0.95-0.99	2	Good
0.99-0.999	3	Very Good
0.999-0.9999	4	Excellent
0.9999-1	5	Almost Perfect

ADEI has the better accuracy when interpolating F_1, where it always generates an excellent fit for adaptive, truly random, and uniform meshes. On the other hand, for F_2 the best method is acm752, with ADEI being comparable for the random sets only. For F_3 all the methods exhibit excellent to almost perfect fitting characteristics, again, when ADEI is not the best, it is comparable to acm752. Figure 3 shows the expected fit over Data Sets I-III, for functions F_1, F_2, and F_3.

It is clear that acm790 didn't perform comparably to the other two methods because no gradient data was supplied to it. We also observed that acm752 improved its performance using true gradient information instead of gradient estimation techniques.

The cost of evaluating the interpolant in acm790 is different than in ADEI and acm752, although all methods share an initial stage where they must find the cell (or triangle[2]) that contains the point to be evaluated. The final evaluation on ADEI and on acm752 requires the evaluation of a cubic polynomial. However, the final evaluation in acm790 involves several evaluations of cubic polynomials, one for each mesh point in a disc centered in the evaluation point, making the evaluation not as fast as the methods based on triangulations. The optimization algorithm described in Algorithm 1 makes the estimation of the overall cost of ADEI difficult, because we cannot tell in advance how many times alternate collocation points have to be chosen, or how many patches will present non-negligible discontinuities.

8. Conclusion

For the given data sets and functions, ADEI performed an excellent to near perfect fit with better, if not compara-

[2]The expected cost of locating a point in a triangulation is $O(log(n))$, whereas for the cell method on acm790 is $O(1)$ with a $O(n)$ worst case [9].

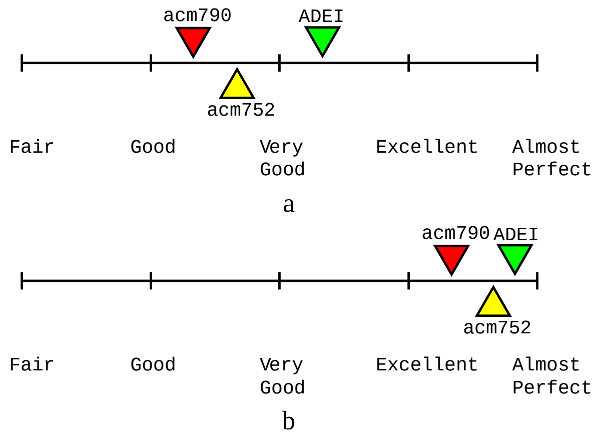


Figure 2. a- Expected fit over functions F_1, F_2 on node sets I-VI; b- Expected fit over function F_3 on node sets I-VI.

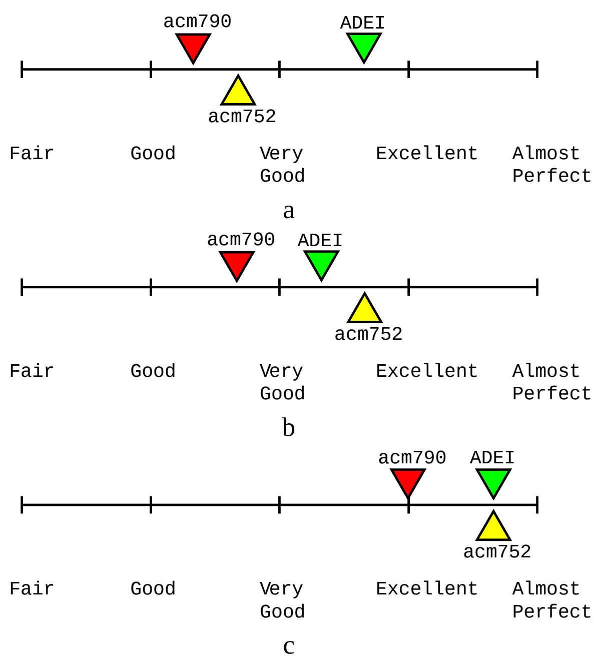


Figure 3. a- Expected fit over Data Sets I-III, on function F_1; b- Expected fit over Data Sets I-III, on function F_2; c- Expected fit over Data Sets I-III, on function F_3.

ble, accuracy results than the algorithms it was compared against, in particular acm752.

Our work only begins an investigation into the sensitivity of the choice of collocation points of the interpolant generated by ADEI. Although there are known cases where the choice will generate a bad solution, we still cannot see a clear way to generate good collocation points. Future work in this area will address this issue, as well as the question of how ADEI and the optimization scale to higher dimensions. Other future research venues include the study of this technique as an effective scheme for the surface representation of large sets of scattered data.

References

[1] R. BARTLES. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Inc, 1987.

[2] W. H. Enright. Accurate approximate solution of partial differential equations at off-mesh points. *ACM Transactions on Mathematical Software*, 26(2):274–292, June 2000.

[3] G. FARIN. *Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide. Fourth Edition*. Academic Press, 1997.

[4] R. Franke. Scattered data interpolation: Test of some methods. *Mathematics of Computation*, 38(175):181–199, 1982.

[5] R. Franke and G. Nielson. Smooth interpolations of large data sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15:1691–1704, 1980.

[6] R.L. Hardy. Theory and applications of the multiquadric-biharmonic method. *Computers Math. Applic.*, 19(8-9):163–208, 1990.

[7] G.M. Nielson. Scattered data modeling. *IEEE Computer Graphics and Applications*, January 1993.

[8] G.A. Ramos. Scattered data interpolation using an alternative differential equation interpolant. Master's thesis, University of Toronto, 2001.

[9] R. J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software*, 14(2):139–148, June 1988.

[10] R. J. Renka. Algorithm 752; srfpack: software for scattered data fitting with a constrained surface under tension. *ACM Transactions on Mathematical Software*, 22(1):9–17, March 1996.

[11] R. J. Renka. Algorithm 790: Cshep2d: Cubic shepard method for bivariate interpolation of scattered data. *ACM Transactions on Mathematical Software*, 25(1):70–73, March 1999.

[12] R. J. Renka and R. Brown. Algorithm 792: Accuracy tests of acm algorithms for interpolation of scattered data in the plane. *ACM Transactions on Mathematical Software*, 25(1):78–94, March 1999.

[13] R. J. Renka and A.K. Cline. A triangle based c1 interpolation method. *Rocky Mt. J Math*, 14(1):223–237, 1984.