Certified Meshing of RBF-based Isosurfaces

Amit Chattopadhyay* Simon Plantinga † Gert Vegter‡

Abstract

Radial Basis Functions are widely used in scattered data interpolation. The process consists of two steps: (i) computing an interpolating implicit function the zero set of which contains the points in the data set, followed by (ii) extraction of isocurves or isosurfaces. We focus on the second step, generalizing our earlier work on certified meshing of implicit surfaces based on interval arithmetic. It turns out that interval arithmetic, and even the usually faster affine arithmetic, are far too slow in the context of RBF-based implicit surface meshing. We present optimized strategies giving acceptable running times and better space complexity, exploiting special properties of RBF-interpolants. We present pictures and timing results confirming the improved quality of these optimized strategies.

1 Introduction

RBF-based interpolants. Radial Basis Functions provide a simple meshless method for reconstruction of smooth geometric objects in the plane or in three-dimensional space from a finite point sample $v_1, \ldots, v_n$. The process consists of two steps: (i) computing an interpolating implicit function the zero set of which contains the sample points, followed by (ii) extraction of the isocurve or isosurface.

The Radial Basis interpolant constructed in step (i) is of the form

$$s(\underline{x}) = \sum_{k=1}^{n} w_k \, \varphi(||\underline{x} - v_k||) + p(\underline{x}), \qquad (1)$$

where $\underline{x} \in \mathbb{R}^d$, for $d = 2, 3$, such that s is zero at the sample points (*centers*) v_k. Here p is a polynomial of low degree, cf [5]. The Radial Basis Function (RBF) φ is a univariate function. Some popular RBFs are $\varphi(r) = r^3$ (thin plate spline in 3D), $\varphi(r) = r^2 \log r$ (thin plate spline in 2D), $\varphi(r) = \sqrt{r^2 + c^2}$ (multiquadric), $\varphi(r) = \exp(-r^2)$ (Gaussian).

The second step, namely isosurface extraction, is our main focus. In [9] we use *interval arithmetic (IA)* to extract regular level sets of a general smooth (C^1) implicit function. More precisely, the algorithm computes a piecewise linear surface which is close (isotopic) to the actual zero set, and is guaranteed to have the same topology. It is akin to the Marching Cubes algorithm in the sense that it analyzes the topology of the isosurface on boxes in the plane or in space. If it cannot decide that the topology is correct, it subdivides the box. However, interval arithmetic converges very slowly for implicit functions like (1), i.e., sums consisting of a large number of terms.

Our contribution. Our early experiments show that even the straightforward use of *affine arithmetic (AA)* [4], a fine tuned version of IA, does not improve running times sufficiently. Therefore, we developed an improved strategy uses *linear* upper and lower bounds, exploiting the fact that each term in the sum is of the same form. This strategy works for certain RBFs, and leads to spectacular improvement of the running time, since far less subdivisions of boxes are needed before the algorithm can decide that the topology is correct. Since such linear bounds are not easy to obtain for all types of RBFs we also developed a more general method based on *quadratic* bounding functions, which works for commonly used RBFs. Finally, we give pictures and performance results confirming the improved quality of the optimized strategy in terms of time and space complexity.

Related Work. Current methods for meshing RBF-based implicit surfaces do not come with topological guarantees, since they are usually based on the marching cubes algorithm [7]. Methods for certified meshing of implicit surfaces are presented in [3, 10]. In [9] interval arithmetic is used to extract certified meshing of implicit surfaces. For an overview of interval arithmetic methods and their optimizations we refer to [8]. Affine arithmetic is discussed in [4].

2 Preliminaries

Interval Arithmetic (IA). Interval arithmetic is used to prevent rounding errors in finite precision computations. A *range function* $\Box F$ for a function $F : \mathbb{R}^m \to \mathbb{R}^n$ computes for each m-dimensional interval I (i.e., an m-box) an n-dimensional interval $\Box F(I)$, such that $F(I) \subset \Box F(I)$. A range function is said to be *convergent* if the diameter of the output interval converges to 0 when the diameter of the input interval shrinks to 0. Convergent range functions exist for the basic operators and functions, so all range functions are assumed to be convergent.

*Corresponding author. University of Groningen, The Netherlands; email: A.Chattopadhyay@rug.nl
†Email: S.Plantinga@rug.nl
‡Email: G.Vegter@rug.nl

Certified Meshing Algorithm. The certified meshing algorithm [9] subdivides the domain of an implicit function until it can approximate the zero set of the function in each box with a topologically correct piecewise linear surface. The algorithm takes an implicit function F and a box B as input, and computes a piecewise linear approximation of $F^{-1}(0) \cap B$, assuming that the zero set $F^{-1}(0)$ of F contains no singular points of F inside B. It uses range functions for F and its gradient ∇F.

Algorithm: ApproximateCurve(F, B)

1. Initialize quadtree T to B;
2. Subdivide T until for all leaves I:
   $0 \notin \square F(I) \vee \langle \square \nabla F(I), \square \nabla F(I) \rangle > 0$;
3. Mesh(T).

Here, Mesh(T) approximates the zero level set inside the box T by a linear function. The first clause in line 2 discards cells I for which $0 \notin \square F(I)$, i.e., boxes which are guaranteed not to contain part of the zero set of F. The second clause implies that $\langle \nabla F(x), \nabla F(y) \rangle > 0$, for all $x, y \in I$, so the direction of the gradient (and, therefore, of the curve) does not change by more than $\pi/2$ over this box. This implies that the zero set of F is *parametrizable* (i.e., can be written as a function of x or y), which is the key property in the proof of topological correctness of the output. We refer to [9] for details.

3 Range functions for RBFs

Unfortunately, for RBF-based implicit functions s of the form (1) an IA-based implementation of algorithm ApproximateCurve(s, I) has unacceptable running times. Our goal is to improve the performance considerably by optimizing the range intervals $\square s(I)$ and $\square \nabla s(I)$ for such RBF-interpolants s on a box I. We restrict to the two-dimensional case, althought our approach works in any dimension.

Computing $\square s(I)$. Our optimization strategy determines a lower bound $l_k(\underline{x})$ and an upper bound $u_k(\underline{x})$ for $w_k \varphi(||\underline{x} - v_k||)$ on the box I, such that the minimal value $L(I)$ of $l(\underline{x}) := \sum_k l_k(\underline{x}) + p(\underline{x})$ on I and the maximum value $U(I)$ of $u(\underline{x}) := \sum_k u_k(\underline{x}) + p(\underline{x})$ on I are easy to compute. Moreover, taking $\square s(I) = [L(I), U(I)]$ should yield a much better range interval than AI, or even AA.

Our approach is based on the observation that the summand $w_k \varphi(||\underline{x} - v_k||)$ is radially symmetric with respect to the center v_k. We will find *quadratic* upper and lower bounds for the *univariate* function $w_k\, \varphi(r)$ for r ranging over the smallest interval $J_k = [r_1, r_2]$ for which $r_1^2 \leq ||\underline{x} - v_k||^2 \leq r_2^2$, for all $\underline{x} \in I$. See Figure 1. More precisely, the univariate upper bound

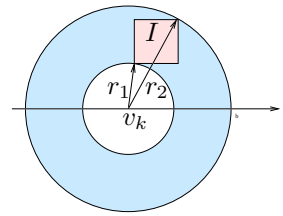

Figure 1: Near and far point of a square interval I. If the center v_k lies inside the box I, then $r_1 = 0$.

of $w_k \varphi(r)$ on J_k is of the form $\alpha_k r^2 + \beta_k$, yielding

$$s(\underline{x}) \leq \sum_{k=1}^{n} \alpha_k ||\underline{x} - v_k||^2 + \sum_{k=1}^{n} \beta_k + p(\underline{x}),$$

for $\underline{x} \in I$. Since, for most RBFs, the polynomial p has degree at most two, the upper bound is a bivariate quadratic function, obtained by adding the coefficients of the upper bounds for each individual summand. Moreover, the maximum value $U(I)$ of this upper bound on the interval I is easily computed. Due to lack of space we just mention that for most RBFs the coefficients α_k and β_k are determined in a rather straightforward way by solving a simple optimization problem, once for each type of RBF. A quadratic lower bound for the RBF-interpolant s on I is determined similarly. In view of the special shape of the quadratic upper and lower bounds this approach is called the *bounding paraboloid strategy (BPARAB)*

Computing $\square \nabla s(I)$. To find optimal ranges $\square s_x(I)$ and $\square s_y(I)$ for the components of the gradient of the RBF-interpolant (1), first note that s_x is given by

$$s_x(\underline{x}) = \sum_{k=1}^{n} w_k \, \frac{\varphi'(||\underline{x} - v_k||)}{||\underline{x} - v_k||} \, (x - v_{kx}) + p_x(\underline{x}), \quad (2)$$

where $\underline{x} = (x, y)$ and $v_k = (v_{kx}, v_{ky})$. Applying the same approximation strategy as before we find quadratic lower bounds on the one-dimensional interval J_k for each of the *univariate factors* $w_k\, \varphi'(r)/r$, leading to a bivariate *cubic* lower bound $L_k(\underline{x})$ on the box I for the k-th summand in (2) of the form

$$L_k(\underline{x}) = a_k\, x\, (x^2 + y^2) + Q_k(\underline{x}),$$

where a_k is a real constant, and $Q_k(\underline{x})$ is a quadratic polynomial. A cubic upper bound $U_k(\underline{x})$ of this form is found similarly. A straightforward derivation yields the minimal value of $\sum_k L_k(\underline{x}) + p_x(\underline{x})$ and the maximal value of $\sum_k R_k(\underline{x}) + p_x(\underline{x})$ on I, and, hence, a good interval $\square s_x(I)$. A good interval $\square s_y(I)$ is computed similarly.

As we will show in Section 4, this strategy improves the performance of the certified meshing algorithm ApproximateCurve considerably for various RBFs.

Bounding plane strategy for the cubic RBF. The cubic RBF, given by $\varphi(r) = r^3$, corresponds to the thin plate spline in 3D, which is used widely in reconstruction of geometric surfaces from scattered point samples. Therefore, for this case we tried to design an even better strategy based on special properties, like convexity, of the RBF. More precisely, using well-chosen *linear* upper and lower bounds, we were able to improve the running time even further in some cases. For experiments with this *bounding plane strategy (BP)*, corroborating this improvement, we refer to Section 4.

4 Experimental results

We present some 2D experiments with algorithm ApproximateCurve, implementing range functions based on IA, AA, BPARAB and, for the cubic RBF, the BP-strategy. We extract the zero sets of various RBF-interpolants, and compare the number of leaves (NOL) of the subdivision tree and the CPU-time (CPU). The RBF-interpolants are constructed using uniform sample interpolation points extracted from several well-known functions, cf. [6], over a bounded domain.

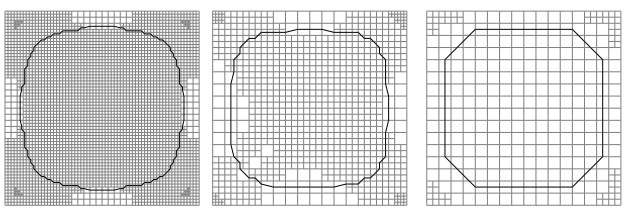

Figure 2: Isocurve extraction for a cubic RBF-interpolant (100 centers) of the function $xy(x-1)(y-1)-0.02$, sampled uniformly on the square $[0,1]\times[0,1]$ using strategies: (i) AA , (ii) BP and (iii) BPARAB.

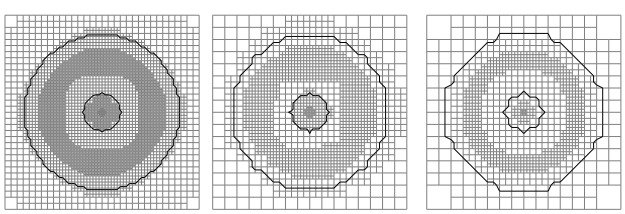

Figure 3: Isocurve extraction for a cubic RBF-interpolant (100 centers) of the function $(x^2+y^2)(1-\sqrt{x^2+y^2})-0.04$, sampled uniformly on the square $[-1.2,1.2]\times[-1.2,1.2]$ using strategies: (i) AA , (ii) BP and (iii) BPARAB.

We used the `Boost` library [1] for IA, and the library [2] for AA. All experiments have been performed on a 3GHz Intel Pentium 4 machine under Linux with 1 GB RAM using the `g++` compiler, version 3.3.5.

Experiments with Cubic RBF. Our first sequence of experiments has been performed using cubic-based interpolants. In other words, we used the RBF given by $\varphi(r) = r^3$. Tables 1–3 presents the measured performance for different optimization strategies. Figure 2–4 contain the corresponding isocurves, together with the boxes corresponding to the leaf-nodes of our subdivision tree.

Note that Table 1 shows that straigthforward use of IA does not lead to convergence (in reasonable time), except in trivial cases. Therefore, we discard IA from our remaining experiments.

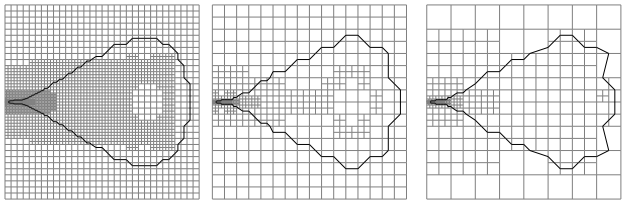

Figure 4: Isocurve extraction for a cubic based RBF-interpolant (100 centers) of the function $4y^2-(x+1)^3(1-x)$, sampled uniformly on the square $[-1.1,1.1]\times[-1.1,1.1]$ using strategies: (i) AA , (ii) BP and (iii) BPARAB.

Experiments with Multiquadric RBF. Next, we show some more experimental results using the multiquadric RBF given by $\varphi(r) = \sqrt{1+r^2}$. Table 4 compares the performance of the AA and BPARAB strategies for this case. Figures 5 and 6 contain the corresponding isocurves.

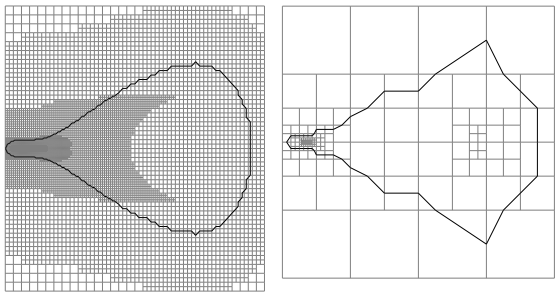

Figure 5: Isocurve extraction for a multiquadric RBF-interpolant (49 centers) of the function $4y^2-(x+1)^3(1-x)$, sampled uniformly on the square $[-1.1,1.1]\times[-1.1,1.1]$ using strategies: (i) AA and (ii) BPARAB.

Conclusion. Our experiments show that IA has unacceptable performance, that AA converges in most experiments with the cubic RBF but fails for the multiquadric-based interpolants, that BPARAB is a general and fast method, and that the BP-strategy for cubic RBFs does not perform better than BPARAB.

References

[1] `Boost` interval arithmetic library. `www.boost.org`.

	IA		AA		BP		BPARA	
NOC	NOL	CPU	NOL	CPU	NOL	CPU	NOL	CPU
25	138616	6.54s	1264	1.6s	280	0.22s	154	0.15s
49	484660	39.6s	2140	5.3s	325	0.49s	274	0.49s
100	1726852	4m13s	3856	25.8s	898	2.19s	304	1.19s
225	6757600	36m47s	5848	1m56s	1156	8.74s	769	6.30s
400	-	-	12880	9m11s	2008	19.1s	1081	16.5s
625	-	-	16408	27m59s	3676	56.6s	1120	27.6s
900	-	-	17980	629s	4237	1m46s	1636	49.5s
1156	-	-	19084	98m55s	4435	2m39s	2032	1m11s

Table 1: Space and time complexity corresponding to Figure 2.

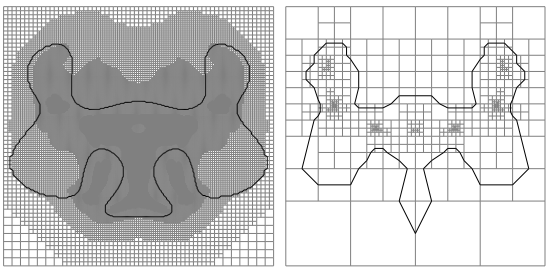

Figure 6: Isocurve extraction for a multiquadric RBF-interpolant (25 centers) of the function $(y-x^2+1)^4+(x^2+y^2)^4-1=0$, sampled uniformly on the square $[-1.2,1.2]\times[-1.4,1.0]$ using (i) AA and (ii) BPARAB.

	AA		BP		BPARAB	
NOC	NOL	CPU	NOL	CPU	NOL	CPU
25	2908	4.07s	640	0.51s	448	0.388s
49	6820	16.5s	1237	1.51s	580	1.10s
100	9052	55.7s	1741	4.10s	1156	3.70s
225	18808	5m36s	3580	19.8s	1528	10.5s
400	24988	17m16s	4492	41.8s	1972	29.0s
625	31492	46m57s	5338	1m9s	3652	1m10s
900	34888	108m	6565	2m6s	4120	1m46s
1156	37288	198m	7663	3m53s	4540	2m40s

Table 2: Space and time complexity corresponding to Figure 3.

	AA		BP		BPARAB	
NOC	NOL	CPU	NOL	CPU	NOL	CPU
25	934	1.29s	181	0.152s	142	0.152s
49	1810	5.96s	376	0.66s	337	0.63s
100	3592	25.9s	784	2.77s	589	2.27s
225	7072	2m39s	1483	11.7s	1183	10.1s
400	11956	2m24s	2350	32.3s	1492	23.2s
625	18766	34m	3691	1m15s	2104	51.2s
900	25252	88m	5122	2m30s	3136	1m45s
1156	27910	154m	5668	3m25s	3895	2m39s

Table 3: Space and time complexity corresponding to Figure 4.

	AA		BPARA	
NOC	NOL	CPU	NOL	CPU
25	1216	1.48s	52	0.068s
49	7726	20.8s	154	0.30s
100	115312	13m36s	274	1.12s
225	—	—	289	2.71s
400	—	—	520	8.66s
625	—	—	598	15.8s
900	—	—	610	22.7s
1156	—	—	874	41.6s

Table 4: Complexity of Multiquadric-based meshing corresponding to Figure 5.

[2] `C++` affine arithmetic library. `savannah.nongnu.org/projects/libaffa`.

[3] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surface meshing. *Discrete and Computational Geometry*, 39:138–157, 2008.

[4] L. H. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms.*, 00:1–13., 2003.

[5] A. Iske. Scattered data modelling using radial basis functions. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, Mathematics and Visualization, pages 287–315. Springer-Verlag, Heidelberg, 2002.

[6] H. Lopes, J. Oliveria, and L. Figueiredo. Robust adaptive polygonal approximation of implicit curves. *Computers and Graphics*, 2002.

[7] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics (Proceedings SIGGRAPH 1987)*, 21(Annual Conference Series):163–169, 1987.

[8] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Comput. Aided Geom. Des.*, 19(7):553–587, 2002.

[9] S. Plantinga and G. Vegter. Isotopic meshing of implicit surfaces. *The Visual Computer*, 23:45–58., 2007.

[10] B. Stander and J. Hart. Guaranteeing the topology of an implicit surface polygonizer for interactive modeling. In *Proceedings SIGGRAPH*, pages 279–286, 1997.