

# Smoothing and near-interpolatory subdivision surfaces

Scott N. Kersey

**Abstract.** In this paper we describe methods for computing smoothing and near-interpolatory (variational) subdivided surfaces, including those surfaces that meet data to within prescribed tolerances. The theory is based on standard results from constrained optimization combined with existing variational interpolatory subdivision schemes. The particular functional considered later in the paper is a discretization of the thin-plate spline functional. This paper considers the characterization and computational algorithms – numerical and smoothness issues will be deferred to future work.

## §1. Introduction

In this paper, methods are developed for computing “smoothing” and “near-interpolatory” subdivision surfaces. The ideas can be considered a generalization of variational interpolatory subdivision, whereby a functional is being minimized to determine the placement of new points at each level of subdivision, with the old points fixed. However, rather than interpolate the previous vertices, we are including either a global error term on the data (similar to smoothing spline curves) or precise bounds on the error from interpolation (i.e., near-interpolation). For the latter, we are in the realm of constrained optimization, and as such our goal is to derive suitable optimality conditions, and to attempt to solve these (approximately). The results are applied to a particular functional (a discretization of the thin-plate spline functional), like in [7] for interpolatory subdivision. In particular, we generalize their “umbrella” scheme to the smoothing and near-interpolatory setup here.

Let  $q = (q_i)$  be a sequence of vertices  $q_i \in \mathbb{R}^3$  of some polygonal surface  $S_q$ , and let  $p = (p_i : i=1:n)$  be the sequence of vertices of a refined surface

$S_p$  at the next level of subdivision, chosen to minimize some energy function  $E(\cdot)$ . Let  $\{I_1, I_2, I_3\}$  be a partition of the index set  $i=1:n$ , corresponding to those vertices that are interpolated, smoothed/near-interpolated, or unconstrained, respectively. Typically, the unconstrained vertices are the new vertices, the smoothed/near-interpolated vertices may be the original vertices, while the interpolated vertices are the remaining vertices from the previous level of subdivision. Corresponding to each  $i \in I_2$ , let  $w_i \geq 0$  be some non-negative *weight*, and  $\varepsilon_i > 0$  a strictly positive *tolerances*. Let  $|\cdot|$  denote the Euclidean norm in  $\mathbb{R}^3$ . Then, we say that  $S_p$  is a “mixed interpolating/smoothing surface” if  $p$  solves the minimization problem

$$\underset{p}{\text{minimize}} \{E(p) + \sum_{i \in I_2} w_i |p_i - q_i|^2 : p_i = q_i \text{ for } i \in I_1\}, \quad (1)$$

while  $S_p$  is a “mixed interpolating/near-interpolating surface” if  $p$  solves the minimization problem

$$\underset{p}{\text{minimize}} \{E(p) : |p_i - q_i| \leq \varepsilon_i \text{ for } i \in I_2, p_i = q_i \text{ for } i \in I_1\}. \quad (2)$$

Note that (1) reduces to pure interpolation if  $I_2 = \emptyset$  and pure smoothing if  $I_1 = \emptyset$ , while (2) reduces to the usual problem of near-interpolation when  $I_1 = \emptyset$ . The term “mixed” is taken from [1]. In the remainder of this paper, problems (1) and (2) will be referred to as simply “smoothing” and “near-interpolation”, respectively.

In Section 2 of this paper, we characterize the solutions to (1) and (2). Problem (1) is a free minimization problem, and so, relatively straightforward to handle. However, (2) requires results from the theory of constrained optimization. As a by-product of the characterizations, we show that these two problems are equivalent for certain smoothing weights  $w_i$  depending on the tolerances  $\varepsilon_i$  in near-interpolation, which leads to a computational algorithm for computing solutions to (2) based on solutions to (1). In Section 3 of this paper, we apply the results from Section 2 to generalize a particular interpolatory subdivision scheme given in [7] to near-interpolation. Some examples are given in Section 4, followed by some performance numbers in Section 5. Further analysis is deferred to future work.

## §2. Characterization and Computation

Let

$$E : \mathbb{R}^{n \times d} \rightarrow \mathbb{R} : p \rightarrow E(p).$$

be some *energy functional*, that we assume to be differentiable. Typically, the functional is quadratic, as for example, a discretization of the “thin-plate spline functional”, as considered in the next section. We denote

partial derivatives of  $E$  by  $D_{p_i}E(\cdot)$ , which is necessarily an element of  $\mathbb{R}^3$  for  $p_i \in \mathbb{R}^3$ .

**Theorem 1.** *Let  $w_i \geq 0$  for  $i \in I_2$ . Suppose that  $p$  solves (1). Then,*

$$\begin{aligned} p_i &= q_i \text{ for } i \in I_1, \\ D_{p_i}E(p) + 2w_i(p_i - q_i) &= 0 \text{ for } i \in I_2, \\ D_{p_i}E(p) &= 0 \text{ for } i \in I_3. \end{aligned} \quad (3)$$

**Proof:** The proof is trivial. Indeed, the first condition comes directly from (1), while the second and third conditions follow by differentiating

$$F(p) := E(p) + \sum_{i \in I_2} w_i |p_i - q_i|^2 \quad (4)$$

with respect to  $p_i$  for  $i \in I_2 \cup I_3$ .  $\square$

**Theorem 2.** *Let  $\varepsilon_i > 0$  for  $i \in I_2$ . Suppose that  $p$  solves (2). Then, for certain non-negative multipliers  $w_i \geq 0$ ,*

$$\begin{aligned} p_i &= q_i \text{ for } i \in I_1, \\ |p_i - q_i| &\leq \varepsilon_i \text{ for } i \in I_2, \\ D_{p_i}E(p) + 2w_i(p_i - q_i) &= 0 \text{ for } i \in I_2, \\ w_i(|p_i - q_i| - \varepsilon_i) &= 0 \text{ for } i \in I_2, \\ D_{p_i}E(p) &= 0 \text{ for } i \in I_3. \end{aligned} \quad (5)$$

**Proof:** The constraints  $p_i = q_i$  and  $|p_i - q_i| \leq \varepsilon_i$  follow directly from the problem statement. To derive the remaining optimality conditions, we will first convert the constrained problem to an unconstrained problem. To do so, we define the *Lagrangian* as the functional

$$L(p, w) := E(p) + \sum_{i \in I_2} w_i (|p_i - q_i|^2 - \varepsilon_i^2), \quad (6)$$

for some *Lagrange multipliers*  $w_i$ . In particular, since  $E$  is differentiable,  $L$  is a differentiable function of  $p_i$ . Moreover, by optimization theory,  $p$  solves (2) iff  $(p, w)$  is a saddle point for (6) (restricted to the subspace  $p_i = q_i$  for  $i \in I_1$ ). On differentiating with respect to  $p_i$ , we have

$$\begin{aligned} D_{p_i}L(p, w) &= D_{p_i}E(p) + 2w_i(p_i - q_i) = 0 \text{ for } i \in I_2, \\ D_{p_i}L(p, w) &= D_{p_i}E(p) = 0 \text{ for } i \in I_3. \end{aligned}$$

Moreover, by analyzing the saddle point condition

$$L(p, \tilde{w}) \leq L(p, w) \leq L(\tilde{p}, w)$$

over all  $\tilde{p} \in \mathbb{R}^{\#I_2 \times d}$  and  $\tilde{w} \in \mathbb{R}^{\#I_2}$ , it follows that  $w_i \geq 0$  and  $|p_i - q_i| - \varepsilon_i = 0$  when  $w_i \neq 0$ , for  $i \in I_2$ . In particular,  $w_i(|p_i - q_i| - \varepsilon_i) = 0$  for  $i \in I_2$ .  $\square$

The fourth condition in the theorem is a so-called *complementarity* or *slack* condition. By this, it follows that the weight  $w_i$  is zero when the constraint is inactive, i.e., when  $|p_i - q_i| < \varepsilon$ .

Note that we could have used other standard results in optimization to prove Theorem 2, such as Kuhn Tucker theory. However, by constructing the Lagrangian (6) explicitly, we gain some insight to the connection between smoothing and near-interpolation. Indeed, the Lagrangian is nearly identical to the smoothing functional (4) in the proof of Theorem 1. Moreover, on comparing the optimality conditions in these two theorems, we conclude the following:

**Corollary 1.** *Suppose the  $p$  solves (2), and that the  $w_i$  are set to the Lagrange multipliers in (5). Then, for these weights  $w_i$ ,  $p$  also solves (1).*

Depending on the objective functional  $E$ , solving problem (1) can be straight-forward. In particular, if  $E$  is quadratic, then (3) reduces to a linear system. But solving (2) has the added complication of the inequality constraints, which would typically be handled by techniques in constrained optimization. However, for the specific constraints here, we have had success with the following iteration:

**Algorithm 1.**

1. Choose tolerances  $\varepsilon_i > 0$ . Initialize the weights  $w_i$ , say  $w_i = 1$ .
2. With the weights fixed, solve (1) for the vertices  $p_i$ .
3. Update the weights according to

$$w_i \leftarrow \frac{|p_i - q_i|}{\varepsilon_i} w_i \tag{7}$$

for  $i \in I_2$ .

4. Iterate between steps 2 and 3.

The rationale behind the algorithm is that, the slack conditions in (5) are satisfied at a fixed point  $w_i$  of (7) in step 3, while the remaining optimality conditions in (5) are satisfied by solving (1) in step 2. The intuitive idea is to pull  $p_i$  closer to  $q_i$  by increasing the weight  $w_i$  when  $|p_i - q_i| > 0$ , and reduce the weight if  $|p_i - q_i| < 0$ . In particular, if a constraint is inactive at a solution, then the iteration will force the corresponding weight to zero. If it is clearly inactive, one can simply set the weight to zero. The iteration is based on one given in [9] for choosing smoothing spline weights, and has been applied to near-interpolation in [2, 3].

### §3. The Discretized Thin-Plate Spline Functional

The *thin-plate spline* minimizes the functional

$$\int \int f_{uu}^2 + 2 f_{uv}^2 + f_{vv}^2 \, dA, \quad (8)$$

over all functions  $f : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$  that interpolate prescribed data. Various discretizations of (8) can be found in the literature, e.g., [6, 7, 8, 10, 11, 12]. Here, we are assuming first that the piecewise linear surfaces  $S_p$  have triangular faces only, and we approximate (8) by the functional

$$F(p) := \sum_i \frac{1}{m_i} \sum_{j=1}^{m_i} \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} |p_{ijk} - 2p_{ij} + p_i|^2. \quad (9)$$

The points  $p_{ij}$  are the direct neighbors of  $p_i$ ,  $j=1:m_i$  (valence  $m_i$ ), and  $p_{ijk}$  are the direct neighbors of  $p_{ij}$ ,  $k=1:m_{ij}$  (valence  $m_{ij}$ ). One may note that what appears in the summand are second divided differences of points on the surface – an approximation to the second derivatives in the thin-plate spline functional.

Our goal is to approximate solutions to the system of equations in (5). To do so, we will iterate by varying just one vertex at a time, holding the others fixed. To this end, let  $r_i$  be a copy of  $p_i$ , and define

$$\tilde{F}(p, r) := \sum_i \frac{1}{m_i} \sum_{j=1}^{m_i} \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} |r_{ijk} - 2r_{ij} + p_i|^2.$$

In particular,  $\tilde{F}(p, p) = F(p)$ . We require that  $D_{p_i} \tilde{F}(p, r) = 0$  for each  $i$ . As it happens (and as we verify in Lemma 1 below), this leads to the equations

$$U^2(p_i) := \frac{1}{m_i} \sum_{j=1}^{m_i} U(p_{ij}) - U(p_i) = 0, \quad (10)$$

with

$$U(p_i) := \frac{1}{m_i} \sum_{j=1}^{m_i} p_{ij} - p_i. \quad (11)$$

These are the second and first order *umbrella operators*, respectively, used in [7]. Our goal here is to generalize their interpolatory scheme to a near-interpolatory scheme.

**Lemma 1.** *The Euler equation  $D_p \tilde{F}(p, r) = 0$  at  $r = p$  leads to the Umbrella conditions (10). In particular,  $D_{p_i} \tilde{F}(p, r)|_{r=p} = 2U^2(p_i) = 0$  for all  $i$ .*

**Proof:** Note that the vertex  $p_i$  appears only in the  $i$ -th summand. On taking variations with respect to  $p_i$ , we get:

$$\begin{aligned}
D_{p_i} \tilde{F}(p, r) &= \frac{2}{m_i} \sum_{j=1}^{m_i} \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} (r_{ijk} - 2r_{ij} + p_i) \\
&= \frac{2}{m_i} \sum_{j=1}^{m_i} \left( \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} (r_{ijk} - r_{ij}) - \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} (r_{ij} - p_i) \right) \\
&= \frac{2}{m_i} \sum_{j=1}^{m_i} \left( \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} (r_{ijk} - r_{ij}) - (r_{ij} - p_i) \right) \\
&= \frac{2}{m_i} \left( \sum_{j=1}^{m_i} \frac{1}{m_{ij}} \sum_{k=1}^{m_{ij}} (r_{ijk} - r_{ij}) - \sum_{j=1}^{m_i} (r_{ij} - p_i) \right) \\
&= 2 \left( \frac{1}{m_i} \sum_{j=1}^{m_i} U(r_{ij}) - \frac{1}{m_i} \sum_{j=1}^{m_i} (r_{ij} - p_i) \right).
\end{aligned}$$

Replacing  $r$  by  $p$  leads to

$$\begin{aligned}
D_{p_i} \tilde{F}(p, r)|_{r=p} &= 2 \left( \frac{1}{m_i} \sum_{j=1}^{m_i} U(p_{ij}) - \frac{1}{m_i} \sum_{j=1}^{m_i} (p_{ij} - p_i) \right) \\
&= 2 \left( \frac{1}{m_i} \sum_{j=1}^{m_i} U(p_{ij}) - U(p_i) \right) \\
&= 2U^2(p_i).
\end{aligned}$$

On setting  $D_{p_i} \tilde{F}(p, r)|_{r=p} = 0$ , we arrive at the umbrella conditions  $U^2(p_i) = 0$ .  $\square$

It follows by (3) and (5) that solutions  $p$  to (1) and (2) must satisfy the linear equations

$$\boxed{
\begin{aligned}
p_i &= q_i, \text{ for } i \in I_1 \\
U^2(p_i) + w_i(p_i - q_i) &= 0 \text{ for } i \in I_2 \\
U^2(p_i) &= 0 \text{ for } i \in I_3
\end{aligned}
} \tag{12}$$

for some non-negative weights  $w_i$ . We cycle through each equation in (12), solving the  $i$ -th equation for  $p_i$  with the other vertices fixed (Gauss-Seidel iteration on the linear system). Of course, nothing needs to be done to satisfy the interpolatory constraints  $p_i = q_i$  for  $i \in I_1$ . This leads to averaging rules (a smoothing scheme) that can be applied locally. This turns out to be a modification of the Umbrella scheme. To derive it, note that (10) can be written as

$$U^2(p_i) := \left(1 + \frac{1}{m_i} \sum_j \frac{1}{m_{ij}}\right) p_i + S,$$

with  $S$  containing the terms that do not involve  $p_i$ . Let  $\nu_i := 1 + \frac{1}{m_i} \sum_j \frac{1}{m_{ij}}$ . Then, we have

$$U^2(p_i) + w_i(p_i - q_i) = \nu_i p_i + S + w_i p_i - w_i q_i = 0.$$

Collecting terms gives

$$(\nu_i + w_i) p_i + S - w_i q_i = 0,$$

and so

$$\begin{aligned} p_i &= \frac{1}{\nu_i + w_i} (w_i q_i - S) \\ &= \frac{1}{\nu_i + w_i} (w_i q_i - (U^2(p_i) - \nu_i p_i)) \\ &= \frac{1}{\nu_i + w_i} (w_i q_i + \nu_i p_i - U^2(p_i)). \end{aligned}$$

Note that this reduces to

$$p_i = p_i - \frac{1}{\nu_i} U^2(p_i)$$

when  $w_i = 0$ , as when  $i \in I_3$ . Hence, to solve (1) iteratively,  $p_i$  is updated according to

$$\boxed{\begin{aligned} p_i &= q_i, \text{ for } i \in I_1 \\ p_i &\leftarrow \frac{1}{\nu_i + w_i} (w_i q_i + \nu_i p_i - U^2(p_i)), \text{ for } i \in I_2 \\ p_i &\leftarrow p_i - \frac{1}{\nu_i} U^2(p_i), \text{ for } i \in I_3. \end{aligned}} \quad (13)$$

To solve (2), we apply Algorithm 1 to the setup here, which leads to the iteration:

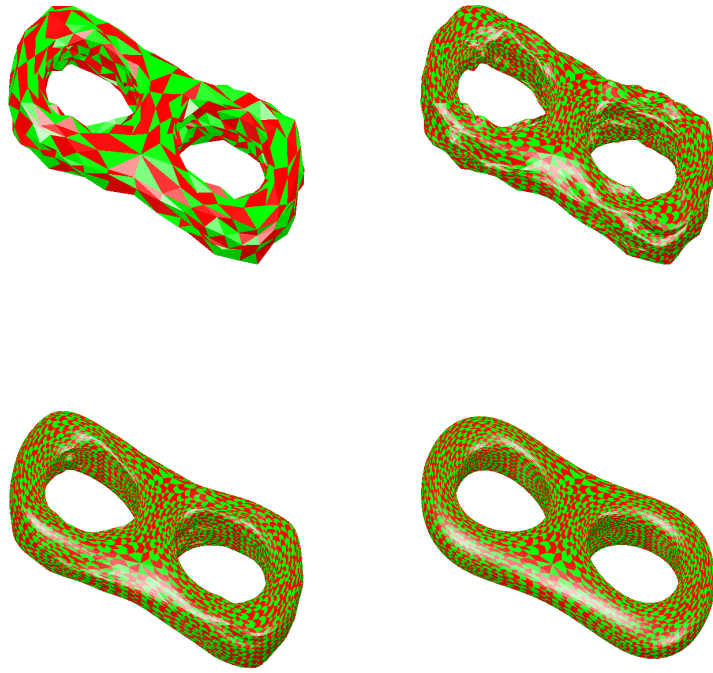
$$\boxed{\begin{aligned} p_i &= q_i, \text{ for } i \in I_1 \\ p_i &\leftarrow \frac{1}{\nu_i + w_i} (w_i q_i + \nu_i p_i - U^2(p_i)), \text{ for } i \in I_2 \\ p_i &\leftarrow p_i - \frac{1}{\nu_i} U^2(p_i), \text{ for } i \in I_3 \\ w_i &\leftarrow \frac{|p_i - q_i|}{\varepsilon_i} w_i, \text{ for } i \in I_2. \end{aligned}} \quad (14)$$

In practice, we iterate on the first three equations in (14) several times to find the “smoothing-spline surface”, following by an update of the weights in the forth equation. Then repeat. Note that (13) reduces to the interpolatory variational scheme in [7] when  $I_2 = \emptyset$ . Moreover, one can approach interpolatory subdivision in (13) by letting  $w_i \rightarrow \infty$ . In near-interpolation (14), this corresponds to  $\varepsilon_i \rightarrow 0$ .

#### §4. Examples

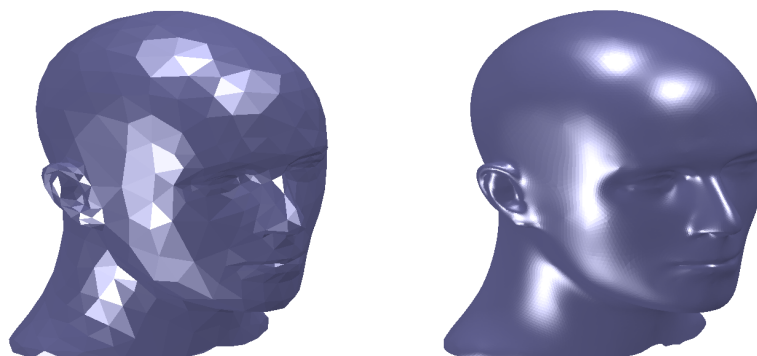
In this section, algorithm (14) is tested on a couple data sets. The goal is to show how near-interpolated surfaces can often produce better results than interpolated surfaces on noisy data sets.

The first data set is displayed in the top left image in Figure 1. The data is particularly “rough”, and, as a consequence, undesirable artifacts appear on the interpolated surface, as clear from the top right image. In the bottom two images, near-interpolation was used to smooth out the data. With the larger tolerances, the bottom right surface is the “fairest”,



**Fig. 1.** Interpolatory and near-interpolatory subdivision to data with random noise. The original data is given in the top left image. The top right image was produced by interpolatory subdivision, and the bottom two images by near-interpolation, with larger tolerances for the bottom right surface.





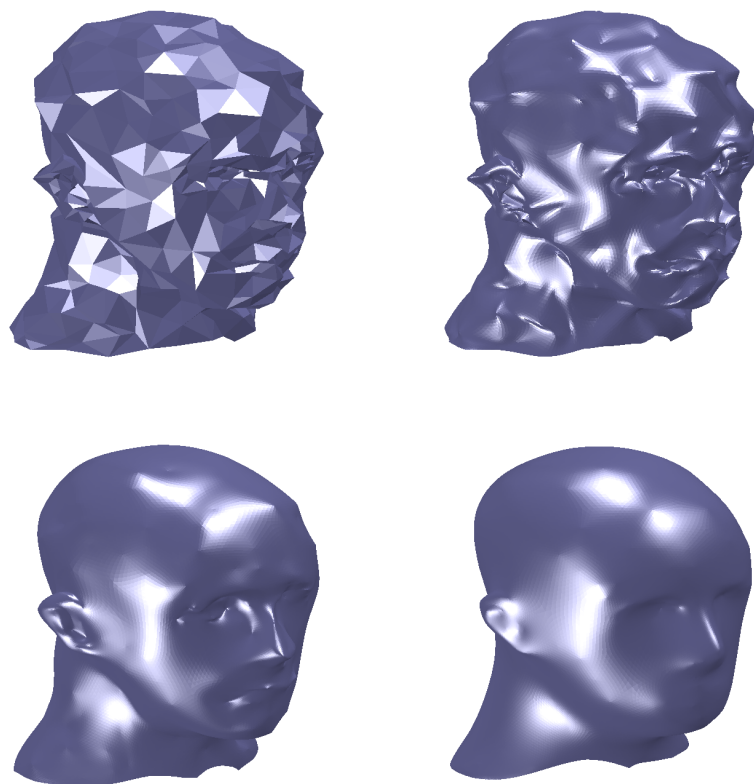
**Fig. 2.** Head data: Original and Interpolated.

while the bottom-left image would more faithfully reproduce the noisy data.

The data in Figure 2 (top left) is nice in the sense that (variational) interpolatory subdivision [7] produces a “nice-looking” surface (top right), albeit, away from the extraordinary points. In Figure 3, error is introduced into the data set. As a result, the interpolated surface (top right) is not particularly appealing. To better recover the original smooth surface, we apply iteration (14), to produce the bottom two images. With smaller tolerances, the bottom-left image perhaps does a better job of reproducing the features of the original (smooth) surface, while the price for a smoother surface, in the bottom right image, is that features have been “washed out”.

## §5. Computation

To give a feel for the performance of the algorithm, the results of a few test runs on a Pentium III computer are given in the following tables. In Figure 4, cpu times (in seconds) are listed for three levels of subdivision for the “head” data (Figure 2), for both interpolatory subdivision (as in [7]) and near-interpolatory subdivision (by (14)). We used 10 weight iterations (of Algorithm 1) for near-interpolation, requiring 9-10 times the number of computations at each level over interpolation. While this is a fixed number, not depending on the number of vertices, it seems a reasonable tradeoff for getting fairly precise tolerances and possibly smoother surfaces. The major downside in the algorithm is that, due to the 4-way triangular split (the number of vertices increases by a factor of four between levels), the computational growth rate between levels of subdivision



**Fig. 3.** Scattered head data: Interpolated and Near-interpolated.

is geometric. As a consequence, it is not practical to subdivide to more than several levels. This is also true for interpolation.

#	Interpolatory $U^2(p)$	Near-Interpolatory $U^2(p)$
1	.16	1.55
2	.74	6.67
3	3.01	26.88

**Fig. 4.** Computation times for three levels of subdivision, in cpu-seconds.

In (14), the weights are adjusted in the outer loop in an attempt to achieve prescribed error bounds  $|p_i - q_i| \leq \varepsilon_i$  at each point. The more iterations, the more accurate one expects the results to be. In Table 5, uniform tolerances of .3, .05, 1 and .3 are given for two data sets. The goal is to achieve these uniform tolerances by iterating on the weights using Algorithm 1 (i.e., the outer loop in (14)). The data displayed is the maximum error  $\max_i |p_i - q_i|$ . As indicated, one gets to within a couple decimal places after 10 or so iterations. Hence, the method does a pretty good job of producing surfaces that approximate solutions to (2). However, for a faster rate of convergence (beyond linear), this iteration would not be suitable. In this case one could resort to standard methods of optimization.

# iter	Torus $\varepsilon_i = .3$	Torus $\varepsilon_i = .05$	Head $\varepsilon_i = 1$	Head $\varepsilon_i = .3$
1	.183	.183	4.22	4.22
2	.246	.0782	1.31	.492
3	.275	.0544	1.095	.410
4	.289	.050658	1.085	.363
5	.306	.050198	1.165	.331
6	.317	.050115	1.170	.315
7	.318	.050080	1.140	.307
8	.315	.050059	1.112	.303
9	.310	.050045	1.114	.301
10	.306	.050035	1.103	.3008
11	.303	.050028	1.086	.30045
12	.301	.050023	1.067	.30036

**Fig. 5.** Convergence to tolerances using Algorithm 1.

**Acknowledgments.** I would like to thank Leif Kobbelt for permission to use the data set in Figure 2.

## §6. References

1. Bezhav, A. Yu. and V. A. Vasilenko, *Variational Spline Theory*, Kluwer Academic/Plenum Publishers, New York (2001).
2. Kersey, S. N., Near-interpolation, *Numer. Math.* **94(3)** (2003), 523–540.
3. Kersey, S. N., On the problems of smoothing and near-interpolation, *Math. Comp.* **72(244)** (2003), 1873–1885.
4. Kersey, S. N., Near-interpolatory subdivided curves, manuscript (available at author's home page), March (2003), 1–23.
5. Kobbelt, L., A variational approach to subdivision, *Comput. Aided Geom. Design* **13** (1996), 743–761.
6. Kobbelt, L., Discrete fairing, In *Proceedings of the seventh IMA conference on the mathematics of surfaces*, (1997), 101–131.
7. Kobbelt, L., S. Campagna, J. Vorsatz and H.-P. Seidel, Interactive multi-resolution modeling on arbitrary meshes, *Proceedings of SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, (1998), 105–114.
8. Kobbelt, L., Discrete fairing and variational subdivision for free-form surfaces design, *The Visual Computer* (2000).
9. Shikin, E., and A. Plis, *Handbook On Splines For The User*, CRC Press, Boca Raton, FL (1995).
10. Weimer, H., and J. Warren, Subdivision schemes for thin plate splines, *Computer Graphics Forum* **17(3)** (1998), 303–313.
11. Welch, W., and A. Witkin, A free-form shape design using triangulated surfaces, *Proceedings of SIGGRAPH 1994*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, (1994), 247–256.
12. Xie, H., and H. Qin, A physics-based framework for subdivision surface design with automatic rules control, *Pacific Graphics* (2002).

Scott N. Kersey  
Georgia Southern University  
Statesboro, GA 30458-8093  
[skersey@georgiasouthern.edu](mailto:skersey@georgiasouthern.edu)  
<http://www.georgiasouthern.edu/~skersey>