Butterworth Filtering and Implicit Fairing of Irregular Meshes

Hao Zhang
School of Computing Science
Simon Fraser University, Canada

Eugene Fiume
Department of Computer Science
University of Toronto, Canada

Abstract

In this paper, we propose efficient numerical techniques for Butterworth filtering and implicit fairing of large irregular triangle meshes, where the corresponding filters are rational polynomials and the resulting large linear systems need to be solved iteratively. We show that significant speed-up can be achieved for Butterworth filtering by factorizing the linear system in the complex domain. As for implicit fairing, with our estimate of the optimal extrapolation parameter ω, successive overrelaxation (SOR) offers great improvements, both in speed and space usage, over the more familiar conjugate gradient type solvers.

1. Introduction

Large and complex mesh models are often obtained from points sampled over real-world 3D objects. But due to the inevitable physical noise added by a scanning device, these point samples often do not reflect their correct locations, resulting in meshes containing undesirable rough features. Mesh fairing (smoothing or denoising) is a process dedicated to the removal of such rough features. Our ultimate goal is to produce highly smooth meshes efficiently, for rendering, modeling, and visualization, while preserving the basic shape and/or features of the original model.

A signal processing approach to mesh fairing was first proposed by Taubin [8], where the mesh geometry is represented as a 3D signal defined over the vertices of the underlying mesh graph. Compared to traditional techniques relying on nonlinear geometric optimization [5], Taubin's application of *polynomial* low-pass filters [8, 9] is known for its simplicity and efficiency. Such an approach builds upon the premise that the geometric irregularities over a mesh have an intuitive frequency-domain characterization. For the ideal low-pass filter, all frequencies higher than a tunable *cut-off frequency*, are eliminated from the signal, while all other frequencies, ones within the *pass-band*, are left unchanged. Thus its impulse response is a step function. Since computing the frequency spectrum of large irregular meshes is impractical, we typically use either polynomials or rational polynomials to approximate the ideal filter.

The well-known Laplacian smoothing [3] applies a polynomial filter which attenuates all but the zero frequency, causing severe shrinkage and shape distortion. Taubin [8] remedies this with the λ-μ filter, where pass-band frequencies are amplified with increasing degree of the λ-μ polynomial. However, λ-μ filters have relatively low smoothing speed [2, 4], and frequency amplification may create ripples over the mesh surface. Taubin et al. [9] then combine Chebyshev polynomial approximation with the classical filter design using windows [6] to derive much better approximations of the ideal low-pass filter, where five to ten-fold speed-up over λ-μ filtering can be obtained easily.

While polynomial filters have finite impulse responses (FIR), infinite impulse response (IIR) filters using *rational polynomials* are also common in digital filter design [6]. Desbrun et al. [2] appear to have been the first to consider IIR filters for mesh fairing. They pose the problem in the context of solving a diffusion PDE, and use implicit integration, called *implicit fairing*, to achieve efficiency, stability, and large time-steps. Implicit fairing of large meshes requires the iterative solution of a large, sparse linear system. Desbrun et al. [2] report that using the biconjugate gradient (BCG) solver with a diagonal preconditioner results in significant savings, in terms of *iteration count*, against Laplacian smoothing or explicit integration. We have observed however that in terms of execution time, the improvement would hardly be obvious since one BCG iteration is about three times as costly as one explicit integration step [1].

Desbrun et al. [2] also suggests that filtering results may be improved by increasing the degree of the denominator of the IIR filter for implicit fairing (we call it the *IF* filter in this paper) with a significant increase in computational cost. It turns out that these filters are the well-known *Butterworth filters* [6]. Higher-order Butterworth filters approximate the ideal low-pass filter better and often result in less shape distortion and better mesh quality up to a certain point, since an overly sharp transition about the cut-off frequency may introduce *ringing* [7]. On the other hand, although the IF filter is not really low-*pass*, as it attenuates all but the zero frequency, it is comparatively more efficient to implement and suitable to use when aggressive fairing is required.

Contribution: In this paper, we address the computational difficulties involved with the realization of both IF and Butterworth filters for large irregular triangle meshes. Our contributions can be summarized as follows:

- We propose an estimation of the optimal extrapolation parameter ω for SOR to be used for the IF filters in implicit mesh fairing. This results in about *two to three-fold speed-up* over implementations using conjugate gradient type solvers for typical smoothing tasks.

- With a factorization of the linear system involved in the complex domain, a $2N$-th order Butterworth filter may be computed as a series of N first-order filters. We demonstrate that the biconjugate gradient stabilized (BCGS) method [1] is suitable to use for the resulting complex systems. For second-order filters, we can already obtain about *four to five-fold speed-up* over the same solver which does not use factorization.

The result of our comparison between Butterworth filtering and Taubin's Chebyshev polynomial approximation [9] is not as conclusive. With the current implementation, our experiments show that to produce meshes of similar smoothness quality, the execution times for the two approaches are comparable. In terms of iteration counts however, Butterworth filtering should definitely be preferred. Due to lack of space, we report these findings in detail in an extended version of this paper [12].

Notations: We represent an irregular triangle mesh M with n vertices by its coordinate signal $\mathbf{x} \in \mathbf{R}^{n\times 3}$, where the k-th row of $\mathbf{x}$ gives the 3D coordinates of vertex k. The *centroid matrix* C of M is such that $C_{jk} = 1/|N_1(j)|$ if (j,k) is an edge and $C_{jk} = 0$ otherwise, where $N_1(j) = \{k|(j,k) \text{ is an edge}\}$ is the set of neighbors of j. We reserve the letter i for the imaginary unit, and I for identity matrices. Eigenvalues (frequencies) are denoted by σ with subscripts. The *spectrum* of a matrix A, i.e., the set of eigenvalues of A, is denoted by $\sigma(A)$. The *spectral radius* $\rho(A)$ of A is the largest magnitude of the elements in $\sigma(A)$.

2. Mesh fairing through IIR filtering

The signal processing approach to mesh fairing relies on a shape decomposition of a mesh with respect to a suitably defined discrete Laplacian operator L. If the eigenvectors $e_1, \ldots, e_n$ of L are linearly independent, then any mesh $\mathbf{x}$ can be expressed as a linear sum $\mathbf{x} = \sum_{j=1}^{n} e_j X_j$, called an *eigenvalue decomposition*, where the set of weights $X_1, \ldots, X_n$ form the mesh signal transform, mimicking the effect of Fourier transform of mesh geometry [9].

In this paper, we consider only the discrete uniform Laplacian operator $U = I - C$, a natural choice for discrete fairing purposes [4, 8]. It can be shown that the eigenvalues of C and U are all real, with $\sigma(C) \subseteq [-1,1]$, $\rho(C) = 1$, and $\sigma(U) \subseteq [0,2]$. Also, the eigenvectors of U are linearly independent. One can regard the eigenvalues of U as the natural *vibration frequencies* of a mesh, and the eigenvectors as its *vibration modes* [8].

Mesh fairing can be achieved through rapid attenuation of the high-frequency contributions in the mesh signal through filtering. Applying a filter $f(\sigma)$ to the mesh $\mathbf{x}$ results in the mesh $f(U)\mathbf{x}$. For example, the (polynomial) Laplacian smoothing filter is of the form $f_{\text{LAP}}(\sigma) = (1-\lambda\sigma)^N$, where $0 < \lambda < 1$ is a time-step parameter [2] controlling the degree of smoothing. So one step of Laplacian smoothing corresponds to multiplying the matrix $I - \lambda U$ to $\mathbf{x}$. If the filter f is a rational polynomial $f = g/h$, then to obtain $f(U)\mathbf{x}$, we need to solve the linear system $h(U)\mathbf{y} = g(U)\mathbf{x}$ for $\mathbf{y}$.

A Butterworth filter of order $2N$ is of the form

$$f_{\text{BUT}}(\sigma) = \frac{1}{1+(\sigma/\hat{\sigma})^{2N}} = \frac{1}{1+(\lambda\sigma)^{2N}}, \qquad (1)$$

where $\hat{\sigma} > 0$ gives the cut-off frequency. The IF filter for implicit fairing[1] [2] happens to be a degenerate case of the Butterworth filters with $2N = 1$. Since this is typically not classified as a Butterworth filter [6], we shall make a distinction by insisting that N be a positive integer for (1) to be a Butterworth filter. For convenience however, we use $\lambda = 1/\hat{\sigma}$, the time-step parameter for implicit integration, to define both the Butterworth and IF filters.

Butterworth filters have several desirable properties. Since $f_{\text{BUT}}(0) = 1$, the DC value of a mesh signal is preserved. As $0 \le f_{\text{BUT}}(\sigma) \le 1$ for all $\sigma \ge 0$, there is no over- or under-shooting. In the pass-band, $f_{\text{BUT}}(\sigma)$ is maximally flat [6] in that its first $2N-1$ derivatives are zero at $\sigma = 0$. Also, its approximation to the ideal low-pass filter is monotonic and as N grows, it gets progressively better. Note that Taubin's Chebyshev polynomial approximations only possess the last of these properties. The IF filter, on the other hand, behaves more like the Laplacian smoothing filter.

3. Numerical techniques for IIR filtering

When applying the IIR filters we have seen so far to mesh fairing, we need to solve the linear system

$$B_N\mathbf{x} = [I + (U/\hat{\sigma})^N]\mathbf{x} = [I + (\lambda U)^N]\mathbf{x} = \mathbf{b}, \qquad (2)$$

where U is our Laplacian operator, and $\mathbf{b}$ is the initial mesh. Since U is not symmetric in general, neither is the linear system. Let us refer to (2) with $N = 1$ as the *IF system*, and the *N-th order Butterworth system*, if $N \ge 2$ is even.

[1]Desbrun et al. [2] consider the diffusion PDE $\partial\mathbf{x}/\partial t = -\eta\Delta(\mathbf{x})$, where $\mathbf{x}$ is a time-dependent signal defined over the mesh and the Laplacian Δ is taken to be U. By integrating the PDE implicitly, we have $(I + \eta dt U)\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)}$, where $\lambda = \eta dt$ is the time step. Given $\mathbf{x}^{(n)}$, this is equivalent to applying an IIR filter $1/(1+\lambda\sigma)$ to $\mathbf{x}^{(n)}$.

3.1. Solving IF systems via SOR

Let us first consider the simpler IF system

$$B_1\mathbf{x} = (I + \lambda U)\,\mathbf{x} = \mathbf{b}. \tag{3}$$

When measuring execution times, we only account for *CPU* time spent on solving the systems. Since for a typical mesh, the number of non-zero entries per row in B_1 is about seven, we store B_1 in an adjacency-list structure to save storage.

We have experimented with twenty mesh models on a 1.8GHz Pentium IV with 256MB RAM. The face count of these meshes range from 23,000 to over 375,000. Seven of them, including the well-known horse, bunny, Igea, and Isis models, are obtained from public-domain mesh libraries, while the rest are generated through decimation.

BCG vs. BCGS (BCG stabilized): Biconjugate gradient, or BCG, is a variant of the conjugate gradient method to handle nonsymmetric systems. Its convergence is often observed for a variety of problems, but few theoretical results are available. Experiments show that the convergence patterns of BCG may be quite irregular and can even break down. The BCGS, or BCG stabilized, method is quite effective in avoiding such pitfalls, but at a slightly higher computational costs per iteration [1]. Our experimental results for IF filtering confirm the above assessments. Compared with BCG with a diagonal preconditioner, as in [2], BCGS exhibits much smoother convergence and achieves about *two-fold speed-up* over BCG quite consistently.

SOR for IF filtering: The SOR method uses an extrapolation parameter ω to accelerate Gauss-Seidel iterations [10]. It is trivial to implement SOR so that its per-iteration cost is only about $1/3$ and memory requirement about $1/2$ (assuming that the average mesh vertex degree is about six) of that of the BCG type solvers [1]. Also, SOR is much preferred in a parallel environment over BCG and its variants since it does not require the computation of inner products, which are communication-intensive [1].

The success of SOR depends critically on the choice of the relaxation parameter ω. In the classical work of Young [11], the optimal ω, one which achieves the fastest convergence, for the so-called *consistently ordered* systems is derived. Although the IF system (3) is not consistently ordered, we can show [12] that it belongs to the closely related class of *generalized consistently ordered* systems [10]. Note that here we need to assume that the mesh is a manifold. In addition, the spectral radius of the *block Jacobi matrix*[2] $J(B_1)$ of B_1 can be computed analytically, as we show below. We are thus motivated to adopt Young's results to estimate the optimal ω for IF filtering.

Recall that $B_1 = I + \lambda U = (1+\lambda)I - \lambda C$. It follows that $J(B_1) = \lambda C/(1+\lambda)$ and since $\rho(C) = 1$, we have $\rho(J(B_1)) = \lambda/(1+\lambda)$. We propose to solve the IF system (3) using SOR, where the relaxation parameter ω^* chosen is given by the optimal ω_b for consistently ordered systems: $\omega_b = 2/(1+\sqrt{1-\rho^2(J(B_1))})$. It follows that

$$\omega^* = \frac{2+2\lambda}{1+\lambda+\sqrt{1+2\lambda}}. \tag{4}$$

Remarkably, our extensive tests show that ω^* is indeed very close to optimal consistently.

Experimental results: In Table 1, we report the performance of SOR and BCGS for six of the twenty mesh models — these are fairly representative of the general trend. We use a stopping criterion which stops the iterations when no apparent progress is being made [12]. This is fairly common for iterative solvers exhibiting a regular convergence behavior [1], as is the case for SOR and BCGS. We set the error tolerance $\epsilon = 3\times10^{-3}$. The SOR parameter $\omega^* = 1.412$ for $\lambda = 10$ and $\omega^* = 1.754$ for $\lambda = 100$; both are obtained from (4). As we can see, SOR achieves about $50 - 65\%$ gain in execution times over BCGS for typically smoothing tasks, which translate to a two to three-fold speed-up.

Mesh	**Horse**	**Bunny**	**Bone**	**Teeth**	**Igea**	**Isis**
Faces	30K	70K	137K	200K	250K	375K
BCGS	0.55	1.25	2.35	3.40	2.74	6.62
SOR	0.28	0.72	1.01	1.49	1.23	2.30
Gain%	49%	42%	57%	56%	55%	65%
BCGS	2.25	3.69	7.10	9.55	7.34	16.02
SOR	0.87	2.14	2.67	3.71	3.15	6.34
Gain%	61%	42%	62%	61%	57%	60%

Table 1. Execution time in seconds and gain for SOR. Top block: $\lambda = 10$. Bottom: $\lambda = 100$.

As pointed out by Desbrun et al. [2], $\lfloor\lambda\rfloor$ steps of Laplacian smoothing (explicit integration) produces about the same smoothness result as implicit fairing using λ as the time step. For small values of λ, e.g., $\lambda \le 15$, $\lfloor\lambda\rfloor$ steps of Laplacian smoothing generally takes less time than even SOR. This is because the per-iteration cost of explicit integration is slightly less than that of SOR. For larger λ's however, IF filtering using SOR starts to outperform explicit integration, and this is more so as λ increases.

3.2. Solving Butterworth systems via factorization

Factorization: Observe that for any positive integer N, the coefficient matrix B_N of the N-th order Butterworth system (2) can be factorized in the *complex domain*: $B_N = \prod_{j=1}^{N}(\lambda U - \gamma_j I)$, where $\gamma_1, \ldots, \gamma_N$ are the N distinct complex roots of -1. When $N = 2^m$, the factorization has a particularly simple form of expression,

$$B_{2^m}(\lambda, U) = \prod_{j=0}^{2^m-1}\left(I + e^{i\frac{(2j+1)\pi}{2^m}}\lambda U\right).$$

[2]Let A_D, A_L, and A_U be the diagonal, strictly lower and strictly upper part of a square matrix A, then $J(A) = -A_D^{-1}(A_L + A_U)$.

Since in most cases, it suffices to use Butterworth filters of these orders to achieve various degrees of mesh smoothing, we shall only consider this particular case in this paper.

In general, the solution of $B_N\mathbf{x} = \mathbf{b}$ can be reduced to the solution of N sparse linear systems in complex numbers. Each system, which we call a *first-order complex Butterworth system*, is more sparse than the original. It turns out that we may reduce the amount of work further by a half. To see this, consider the second-order system,

$$(I + \lambda^2 U^2)\mathbf{x} = \mathbf{b}, \tag{5}$$

which is equivalent to $(I + i\,\lambda U)\,(I - i\,\lambda U)\,\mathbf{x} = \mathbf{b}$, where $U, \mathbf{x}, \mathbf{b}$ are real. We show that it suffices to solve

$$(I + i\,\lambda U)\,\mathbf{y} = \mathbf{b}, \tag{6}$$

for $\mathbf{y} \in \mathbf{C}^n$. Let $\mathbf{y}'$ and $\mathbf{y}''$ be the real and imaginary parts of $\mathbf{y}$. Since both $\mathbf{x}$ and $\mathcal{U}$ are real, we have $(I - i\lambda U)\mathbf{x} = \mathbf{y} = \mathbf{y}' + i\mathbf{y}''$ if and only if $\mathbf{x} = \mathbf{y}'$ and $-\lambda U\mathbf{x} = \mathbf{y}''$. Note that the second condition is redundant, as it can be implied from $\mathbf{x} = \mathbf{y}'$ and (6). Therefore, $\mathbf{y}'$, the real part of $\mathbf{y}$ is exactly the solution to the second-order system (5).

Numerical solution of Butterworth systems: As shown above, to solve a $2N$-th order real Butterworth system, we need to solve a series of N first-order complex Butterworth systems of the form

$$(I + \lambda e^{i\theta} U)\,\mathbf{x} = \mathbf{b},$$

where $\lambda = 1/\hat{\sigma}$ with $\hat{\sigma}$ being the cut-off frequency. Since a complex multiplication is about three times as costly as a real multiplication, we shall minimize the number of complex numbers in the coefficient matrix by applying a preconditioner $e^{-i\theta}I/\lambda$, and solve the following system instead,

$$\tilde{B}\mathbf{x} = (e^{-i\theta}I/\lambda + U)\,\mathbf{x} = e^{-i\theta}\mathbf{b}/\lambda. \tag{7}$$

Complex BCG and BCGS (BCG and BCGS in complex numbers), or CBCG and CBCGS for short, can be implemented in the same way as for the real case. But the irregular convergence patterns suffered by BCG appear to worsen consistently for the complex systems, resulting in extremely slow convergence. Thus we report results for BCGS only.

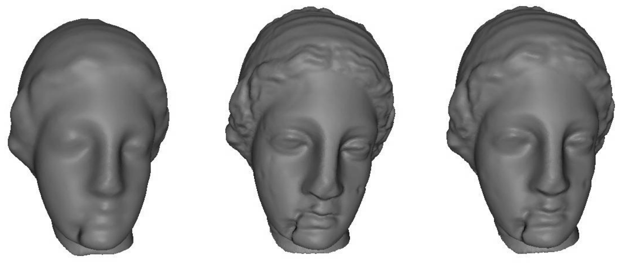
As for complex SOR, or CSOR for short, we could follow the same approach we have taken for the real case. The block Jacobi matrix for (7) is $J(\tilde{B}) = \lambda C/(\lambda + e^{-i\theta})$, and $\rho(J(\tilde{B})) = \lambda/(\lambda + e^{-i\theta})$. We could get an estimate $\tilde{\omega}^*$, of the optimal ω using (4) again. However, the convergence results obtained are not satisfactory. In fact, the relaxation parameters obtained this way are often not close to optimal.

Experimental results (second-order filters): We implement the second-order Butterworth filter with cut-off frequency $\hat{\sigma} \approx 0.033$, or equivalently, $\lambda = 30$. The performance of three methods are compared: direct solution of the second-order filter using (real) BCGS without factorization, which we call BCGS2, and CBCGS and CSOR applied to the first-order complex system (7).

In Table 2, we report the execution times we obtain. The error tolerance for both CBCGS and CSOR is $\epsilon = 3 \times 10^{-3}$, as before. But we have chosen to use a smaller error tolerance, $\epsilon = 1 \times 10^{-3}$, for BCGS2. This is because its convergence appears to be significantly slower than that of the other two methods. If we had chosen the same tolerance for all three methods, then we would not have been able to obtain similar mesh smoothness quality to make a fair comparison. Even with the current choice of ϵ's, BCGS2 still cannot achieve the same level of smoothness as CBCGS for some models, as illustrated in Figure 1.

Mesh	**Horse**	**Bunny**	**Bone**	**Teeth**	**Igea**	**Isis**
Faces	30K	70K	137K	200K	250K	375K
BCGS2	25.5	31.8	82.4	68.0	170.6	98.7
CBCGS	2.3	6.5	19.0	17.4	27.2	32.6
Gain%	91%	80%	77%	74%	84%	67%
CSOR	4.8	8.9	15.6	26.5	36.1	52.4

Table 2. Execution time in seconds and % gain for factorization (cut-off frequency $\hat{\sigma} = 0.033$).



(a) CBCGS. (b) BCGS2. (c) BCGS2.

Figure 1. Results of second-order Butterworth filtering with different error tolerances. (a) and (b): $\epsilon = 3 \times 10^{-3}$. (c): $\epsilon = 1 \times 10^{-3}$.

The computational advantage achieved through factorization is quite evident. We can generally observe about four to five-fold speed-up of CBCGS over BCGS2 for typical smoothing tasks, and as the order of the Butterworth filter increases, a higher percentage of gain is obtained. It is also worth noting that the iteration count of CBCGS is often about the same as the iteration count of BCGS for the first-order IF system (3), which is highly desirable.

Higher-order Butterworth filters: Higher-order Butterworth filters (order $2N \geq 4$) can be implemented using factorization and CBCGS at a higher computational cost. In Figure 2, we show the results of using first- and fourth-order filters on the bunny. Observe that the accuracy of the fourth-order filter, in approximating the ideal low-pass filter, translates to less shape distortion, as shown around the neck, the body, and especially the ear of the bunny. This would be impossible to achieve with lower-order filters, for which visible shape distortion often appears.

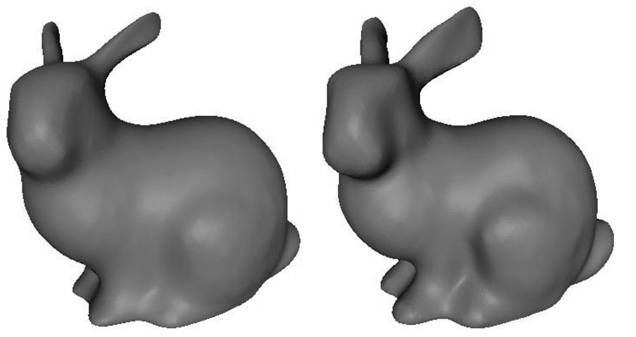


Figure 2. Butterworth filtering with $\hat{\sigma} = 0.02$. Left: first-order. Right: fourth-order.

However, as far as mesh fairing is concerned, filters with a very sharp transition about the cut-off frequency, as is the case for Butterworth filters with order $2N \geq 8$, are not recommended because of possible emergence of the ringing phenomenon [7], which is illustrated in Figure 3.

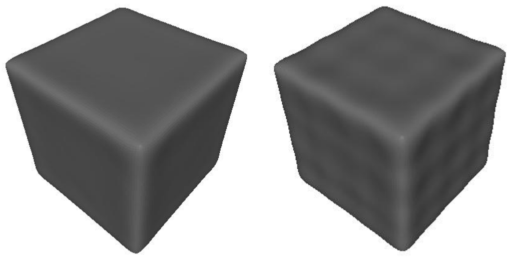


Figure 3. The ringing phenomenon. Left: second-order. Right: sixteenth-order

4. Conclusion and future work

The IF system for implicit fairing appears to be highly structured, as they belong to the class of generalized consistently ordered systems [12]. In using SOR to solve the IF systems, we exploit these structures to obtain significant speed-up over the use of conjugate gradient type solvers. Implicit fairing using SOR also outperforms Laplacian smoothing when sufficient level of smoothing is requested. For Butterworth filters, we show that by factorizing the linear system involved in the complex domain, the computational work required can be greatly reduced.

As we have seen, a great deal of effort is needed to attain the promised benefits of Butterworth filtering and implicit fairing [2]. We would like to view our work presented in this paper as a first attempt to tackle these important numerical problems. An immediate improvement over our current treatment of the problem is to utilize an objective measure of fairness quality as the means of evaluation, rather than using some fixed error tolerance determined by visual examination. We would also like to look into applying and/or extending the techniques developed in this paper to other Laplacian operators, such as the scale-dependent Laplacian and the mean curvature flow operator [2], as well as to other filters, such as band-pass and high-pass filters [7].

Another interesting variation of the IIR filtering approach is to incorporate weights or constraints, thus making the diffusion equation *anisotropic*. Finally, we plan to investigate the complex SOR problem further, and it would also be nice to have some theoretical justification for the use of ω^* in (4) for real SOR.

Acknowledgment: We appreciate discussions with Gabriel Taubin and Robert Russell on related topics and helpful comments from the anonymous reviewers. This research was in part supported by NSERC-262118. The mesh display software used is QVIS, developed by Michael Garland.

References

[1] R. Barrett, et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods,* SIAM, 1994.

[2] M. Desbrun, M. Meyer, P. Schröder, and A. Barr, "Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow," *SIGGRAPH 99,* pp. 317-324, 1999.

[3] D. Field, "Laplacian Smoothing and Delaunay Triangulations," *Communications in Applied Numerical Methods,* Vol. 4, pp. 709-712, 1988.

[4] L. Kobbelt, S. Campagna, J. Vorsatz, and H-P. Seidel, "Interactive Multi-Resolution Modeling on Arbitrary Meshes," *SIGGRAPH 98,* pp. 105-115, 1998.

[5] H. Moreton and C. H. S'equin, "Functional Optimization for Fair Surface Design," *SIGGRAPH 92,* pp. 167-176, 1992.

[6] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing,* Pretice Hall, 1975.

[7] W. K. Pratt, *Digital Image Processing,* 2nd Ed., Wiley, 1991.

[8] G. Taubin, "A Signal Processing Approach to Fair Surface Design," *SIGGRAPH 95,* pp. 351-358, 1995.

[9] G. Taubin, "Optimal Surface Smoothing as Filter Design," IBM Research Report, 1996.

[10] R. S. Varga, *Matrix Iterative Analysis,* Second Edition, Springer, 2000.

[11] D. M. Young, *Iterative Solution of Large Linear Systems,* Academic Press, 1971.

[12] H. Zhang and E. Fiume, "Butterworth Filtering and Implicit Fairing of Irregular Meshes," *Extended version* of current paper at http://www.cs.sfu.ca/˜haoz/papers/butimp.pdf.