Sherman-Morrison-Woodbury-Formula-Based Algorithms for the Surface Smoothing Problem*

Shang-Hong Lai†

Siemens Corporate Research
Princeton, New Jersey 08540

and

B. C. Vemuri‡

Department of Computer & Information Sciences
University of Florida
Gainesville, Florida 32611

Submitted by Richard A. Brualdi

ABSTRACT

Surface smoothing applied to range-elevation data acquired using a variety of sources has been a very active area of research in computational vision over the past decade. Generalized splines have emerged as the single most popular approximation tool to this end. In this paper we present a new and fast algorithm for solving the surface smoothing problem using a membrane, a thin-plate, or a thin-plate–membrane spline for data containing discontinuities. Our approach involves imbedding the surface smoothing problem specified on an irregular domain (in the sense of discontinuities and boundaries) in a rectangular region using the capacitance-matrix method based on the Sherman-Morrison-Woodbury formula of matrix analysis. This formula is used in converting the problem of solving the original linear system resulting from a finite-element discretization of the variational formulation of the surface smoothing problem to solving a Lyapunov matrix equation or a cascade of two Lyapunov matrix equations. The reduced problem can then be solved very efficiently using the ADI method and the bi-conjugate-gradient technique. Our solution requires the generation

*This research was supported in part by the NSF grant ECS-9210648.
†E-mail: lai@scr.siemens.com.
‡E-mail: vemuri@cise.ufl.edu.


© 1997 Elsevier Science Inc. All rights reserved.
655 Avenue of the Americas, New York, NY 10010
0024-3795/97/$17.00
PII S0024-3795(97)00191-2
of a dense capacitance matrix, for which we propose a practical and efficient method. We demonstrate the efficiency of our algorithm via experiments on sparse-data surface smoothing, making performance comparisons with the conjugate-gradient and hierarchical-basis preconditioned conjugate-gradient algorithms. © 1997 Elsevier Science Inc.

1. INTRODUCTION

Surface smoothing is a very important problem in several areas of engineering. This problem manifests itself in two possible ways in the field of early vision, the application domain of interest in this paper. Early vision involves tasks wherein meaningful 3D shape information is extracted from raw 2D image data. Tasks such as extraction of 3D shape from shaded 2D images of objects, from textured images of objects in a scene, from a stereo pair of images of objects in a scene, etc., require that a dense surface shape be recovered either from a sparse, scattered set of noisy depth-elevation constraints or from a dense set of noisy surface orientation constraints. In some applications, it is required to obtain a smooth surface approximation to a dense set of noisy surface depth/elevation constraints. The key feature of these problems is that the data can contain surface depth as well as surface orientation (normal) discontinuities at some unknown set of locations in the domain. The goal in early vision is to recover the surface shape along with an explicit representation of the discontinuities from the noisy measurements. This problem, with the exclusion of discontinuities, is primarily the surface smoothing problem of approximation theory [13, 14, 10, 20]. In this paper, we will not address the automatic detection of depth/orientation discontinuities; however, their location is assumed to be known a priori, and the algorithms for surface smoothing will provide scope for explicitly incorporating the discontinuities at known locations into the solution.

Numerous solution methods exist for the surface smoothing problem in the approximation-theory literature [13, 14, 10, 20, 15, 8]. Most of these do not deal with data containing discontinuities in the surface or its orientation. Since discontinuities contain very useful information about the shape of an object, numerous techniques of surface smoothing have been developed over the past decade and a half to construct a smooth surface approximation with explicit representation of discontinuities. The most popular formulation of the surface smoothing problem has been the variational-principle formulation. In this formulation, surface smoothing is treated as an energy minimization problem involving terms corresponding to a smoothness constraint and a data constraint. Smoothness constraints are expressed in the form of a Tikhonov or
a controlled continuity stabilizer [18]. A thin-plate ($C^1$) or a membrane ($C^0$) surface is called a Tikhonov stabilizer, and a convex combination of the two is called a controlled continuity stabilizer. In this framework, all data constraints are treated as penalty terms. A physical interpretation may be given to the penalty term by attaching springs from the elevation/depth data points to a thin-plate or membrane surface freely floating over the domain of data specification. The stiffness in the spring is made inversely proportional to the uncertainty in the data. The goal is to minimize the strain energy in the constrained thin-plate, membrane, or thin-plate-membrane surface. When using the controlled continuity stabilizer, the solution surface will comprise thin-plate patches away from orientation discontinuities and membrane strips along loci of orientation discontinuities [19].

The solution surface is defined by a configuration of minimal energy $\mathcal{E}$ of the physical system composed of the thin-plate patches, the membrane strips, and the springs. To solve the variational principle an explicit expression must be obtained for the potential energy functional $\mathcal{E} : V \rightarrow \mathbb{R}$, where $V$ is the admissible space of the variational principle. The space $V$ consists of functions which represent possible plate-membrane configurations; it is a (Sobolev) space of bounded energy functions. The precise expression for the variational principle is given by

$$
\mathcal{E}(v) = \int_{\Omega} \rho(x, y) \left\{ \tau(x, y) \left( v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2 \right) + [1 - \tau(x, y)] \left( v_x^2 + v_y^2 \right) \right\} \, dx \, dy , \tag{1}
$$

where $v(x, y)$ is the admissible (thin-plate-membrane) deflection function and $v_x, v_y$ its partial derivatives, assumed to be small. The domain $\Omega$ is a region of interest over which the data are specified. $\rho(x, y)$ and $\tau(x, y)$ are called continuity control functions; they constitute an explicit representation of depth and orientation discontinuities, respectively. The functions range over the interval $[0, 1]$ and need not vary continuously. Setting the continuity control functions $\rho$ and $\tau$ requires prior knowledge of the location of the discontinuities. Such knowledge may be obtained from other sources of information such as registered multisensor data.

The above energy expression serves as a stabilizer in the overall variational principle for the surface smoothing problem. To the stabilizer, we add data constraints via what are known as penalty terms. For the surface smoothing from depth data, the following penalty term, which measures the discrepancy between the surface and data weighted by the uncertainty in the data, may be
used:

\[ e_d(u) = \frac{1}{2} \sum c_i [v(x_i, y_i) - d_i]^2, \tag{2} \]

where \( d_i \) are the depth data points specified in the domain \( \Omega \), and \( c_i \) are the uncertainties associated with the data. The total energy is

\[ E = E_d + \lambda E_s, \tag{3} \]

where \( \lambda \) is the regularization parameter that controls the amount of smoothing performed. The goal is to find a \( u \) that minimizes the total potential energy \( E(u) \) in Equation (3).

Two different approaches have been proposed for solving the above variational problem. One is the reproducing-kernel Hilbert-space approach [2, 10, 20], which gives an analytical expression for the smoothing spline solution. This expression is a linear combination of the fundamental solutions of the corresponding differential operator (e.g., the biharmonic operator \( \Delta^2 \) for the thin-plate spline) shifted to the data locations, and the basis functions in the null space of this differential operator. Thus, the problem is reduced to finding the coefficients for the shifted fundamental solutions and the basis functions, which can be obtained by solving a dense linear system. The other approach for minimizing the above energy functional is by use of a finite-element technique to directly discretize the variational principle [19, 17] or numerically solve the PDE corresponding to the necessary condition for a minimum of the variational principle [19]. Then, the energy functional in Equation (3) becomes a function of the nodal variables used in the discretization. The above functional minimization problem is reduced to minimizing this function with respect to the nodal variables. In the reproducing kernel Hilbert-space approach, the solution is obtained over all of \( \mathbb{R}^2 \), whereas in the finite-element (or finite-difference) method, the solution is usually computed only on a bounded domain. Additionally, we can easily incorporate discontinuities of any order into the solution when using the finite-element approach [19], whereas the ability to incorporate discontinuities for the reproducing-kernel Hilbert-space approach is very limited [8]. In this paper, we use the finite-element discretization approach to solve the above-discussed variational principle.

To compute a numerical solution to the above minimization problem, we first discretize the functionals \( E_d \) and \( E_s \) using finite-element techniques [19]. For an irregular domain \( \Omega \), we can use the domain imbedding technique and discretize the problem on a rectangular domain. For ease of
exposition, the domain \( \Omega \) is assumed to be rectangular in the rest of this paper. For constant \( \rho(x, y) \) and \( \tau(x, y) \) the energy function is a quadratic form. The energy due to the data compatibility term in discrete form becomes

\[
E_d(x, d) = \frac{1}{2} (x - d)^T K_d (x - d),
\]

(4)

where \( x \in \mathbb{R}^{n^2 \times 1} \) is a vector form of the discretization of \( v(x, y) \), \( d \in \mathbb{R}^{n^2 \times 1} \) is the data vector, and \( K_d \in \mathbb{R}^{n^2 \times n^2} \) is a diagonal matrix (for uncorrelated noise in the data) containing the uncertainties \( \sigma_i \) associated with the data points. The smoothness energy in Equation (1) in discrete form is a quadratic in \( x \) when the continuity control functions \( \rho(x, y) \) and \( \tau(x, y) \) are assumed to be constants, i.e.,

\[
E_s(x) = \frac{1}{2} x^T K_s x,
\]

(5)

where \( K_s \in \mathbb{R}^{n^2 \times n^2} \) is a very large sparse block banded matrix containing the computational molecules [19] from the membrane or thin-plate smoothness constraint. The resulting energy function is a quadratic in \( x \),

\[
E(x) = \frac{1}{2} x^T K x - x^T b + c,
\]

(6)

with \( K = \lambda K_s + K_d \) and \( b = K_d d \). The matrix \( K \) is also sparse with block banded structure and is called the stiffness matrix. The minimum of Equation (6) is found by solving the large sparse linear system

\[
Kx = b.
\]

(7)

Another way of approaching the solution to the minimization of the energy \( \mathcal{E} \) is using the calculus of variations [19]. The necessary condition for the minimum of the energy functional is a partial differential equation called the Euler-Lagrange equation. Discretization of this PDE leads to a system of algebraic equations as given in Equation (7). This large \( n^2 \times n^2 \) sparse linear system can be solved using iterative techniques such as successive overrelaxation (SOR), multigrid methods [18], hierarchical conjugate gradient [17], etc. The convergence of these methods deteriorates in the presence of discontinuities. In this paper, we develop a fast algorithm to solve the above linear system which takes account of the data and discontinuity locations in a unified manner. Consequently, the performance of the resulting algorithm is less sensitive to the discontinuities.
Our approach involves imbedding the surface smoothing problem specified on an irregular domain (in the sense of discontinuities and boundaries) in a rectangular region using the capacitance-matrix method based on the Sherman-Morrison-Woodbury formula of matrix analysis. This formula is used to convert the problem of solving the original linear system resulting from a discretization of the variational formulation of the surface smoothing problem to solving a Lyapunov matrix equation or a cascade of two Lyapunov matrix equations. These equations have a right-hand-side matrix which is obtained via a modification of the right-hand-side vector in the original linear system. We use the alternating direction implicit (ADI) method to solve these Lyapunov matrix equations in a constant number of iterations for a given error tolerance with each iteration taking $O(N)$ time, where $N$ is the number of nodes in the discretization [7, 6]. The modified right-hand side is obtained by solving a linear system with a smaller dense and nonsymmetric capacitance matrix, using a bi-conjugate-gradient algorithm. For the generation of this dense capacitance matrix we propose a practical and efficient solution.

The rest of the paper is organized as follows. First we derive the equivalent Lyapunov system for the original linear system in Section 2. The numerical solution technique is then described in Section 3. We present some experimental results depicting the performance of our algorithm in Section 4 and draw conclusions in Section 5.

2. OUR APPROACH

In this section, we will formulate the problem of solving the algebraic system of equations resulting from the discrete version of the variational principle, discussed in the previous section, into solving a Lyapunov matrix equation or a cascade of two Lyapunov matrix equations with a modified right-hand side (RHS).

The surface smoothing problem can be posed as follows: Given a dense or a sparse set of depth data, find a surface which passes close to the data, is smooth, and preserves any prespecified discontinuities in the data.

The stabilizer discussed in the previous section imposes smoothness constraints on the surface smoothing problem leading to a $C^0$ surface (the membrane spline), a $C^1$ surface (the thin-plate spline), or a combination of the two (the thin-plate-membrane spline). As mentioned earlier, the numerical solution to the surface smoothing problem involves solving a linear system $Kx = b$. The matrix $K$ is large and sparse but not tightly banded. This is primarily due to the fact that a vector has been used to represent a
two-dimensional array, which leads to representing neighboring interactions in two dimensions by interactions between very distant elements in one dimension. The matrix $K$ is a sum of two matrices $K_s$ and $K_d$, with $K_s$ containing the membrane or thin-plate molecules [19] defined on the interior and boundary of $\Omega$. The matrix $K_d$ is a diagonal matrix containing nonzero weights.

By making use of the structure of the matrix $K$, we can split $K$ into two components $K_0$ and $UV^T$, with $K_0 \in \mathcal{M}_N (N = n^2)$ and $U, V \in \mathcal{M}_{N \times p}$, such that $K_0$ is close to $K$, i.e., the rank of $UV^T$ is small compared to the rank of $K$, and there exists a fast algorithm to solve a linear system with the matrix $K_0$. Then, the solution to the linear system $Kx = b$ can be obtained by solving a well-structured linear system $K_0x = b'$ with a modified RHS $b'$. The derivation is based on the Sherman-Morrison-Woodbury formula [4]

$$(K_0 + UV^T)^{-1} = K_0^{-1} - K_0^{-1}U(I + VTK_0^{-1}U)^{-1}V^TK_0^{-1}. \quad (8)$$

where $K_0$ is nonsingular, $I \in \mathcal{M}_p$ is the identity matrix, and the matrix $I + VTK_0^{-1}U$ is nonsingular. The matrix $I + VTK_0^{-1}U$ is called the capacitance matrix [5] and is denoted by $C$. Note that $C$ is nonsingular when both $U$ and $V$ are of full column rank. Consequently, we choose the matrices $U$ and $V$ so that the respective column vectors of these two matrices are linearly independent. Thus, the solution to the linear system $Kx = b$ can be obtained by using the Sherman-Morrison-Woodbury formula, which yields

$$x = K_0^{-1}(b - UC^{-1}V^TK_0^{-1}b). \quad (9)$$

Denote $b' = b - UC^{-1}V^TK_0^{-1}b$. Thus the original linear system $Kx = b$ is converted to an equivalent linear system $K_0x = b'$, which can be solved very efficiently with an appropriate choice of $K_0$.

The choice of $K_0$ depends on the data compatibility matrix $K_d$ and the smoothness constraint matrix $K_s$. We have different choices of $K_0$ for the cases of sparse (scattered) and dense data. They are given in the following subsections.

2.1. Dense-Data Surface Smoothing

We will first describe our choice of the matrix $K_0$ and the equivalent Lyapunov matrix equation for the dense-data surface smoothing with a membrane spline. Then we will present the matrix $K_0$ for the surface smoothing with a thin-plate-membrane spline, based on the construction
scheme for the membrane problem. The thin-plate spline is a special case of thin-plate-membrane spline and will not be discussed separately. In addition, the selection of the matrices U and V will be discussed for both cases.

2.1.1. Membrane Spline for the Dense-Data Case. The data compatibility matrix \( K_\text{d} \) for the dense-data case with uniform weighting is an identity matrix. Focusing on the surface-smoothing problem in a rectangular domain, the discretization of the data compatibility energy and the membrane smoothness energy leads to a stiffness matrix \( K \) that can be formed from the computational molecules given in [19]. Note that we have a data constraint molecule in every node all over the domain for the dense-data case. A membrane molecule exists between most neighboring nodes except those separated by the discontinuities. Therefore, we choose the matrix \( K_0 \) to contain both the membrane computational molecules and the data constraint molecules all over the domain, as shown in Figure 1. The difference between the matrices \( K \) and \( K_0 \) is only in the enforcing of the membrane molecules at the discontinuity locations in \( K_0 \). The matrix \( K_0 \) of this choice has a very regular structure all over the rectangular domain. Using this choice of \( K_0 \), we can transform the original linear system \( Kx = b \) to an equivalent Lyapunov matrix equation, which can be solved very efficiently using the ADI method.

With the above choice, the matrix \( K_0 \) can be written as follows:

\[
K_0 = \lambda K_{\text{mem}} + K_\text{d} = \lambda K_{\text{mem}} + I_n^2, \tag{10}
\]

where

\[
K_{\text{mem}} = \begin{bmatrix}
D_1 & -I_n & & & \\
-I_n & D_2 & \ddots & & \\
& \ddots & \ddots & \ddots & \\
& & D_{n-1} & -I_n & \\
& & & 0 & D_n
\end{bmatrix}, \tag{11}
\]

\[
D_1 = D_n = \begin{bmatrix}
2 & -1 & & & \\
-1 & 3 & -1 & & \\
& \ddots & \ddots & \ddots & \\
& & -1 & 3 & -1 \\
& & & -1 & 2
\end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{12}
\]
Fig. 1. (a) Data constraint molecule. Membrane computational molecules for (b) the interior of the domain $\Omega$, (c) the boundary of $\Omega$, and (d) the corner of $\Omega$.

$$D_2 = \cdots = D_{n-1} = \begin{bmatrix} 3 & -1 & & & \cdot & \cdot & \cdot & -1 \\ -1 & 4 & -1 & & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \ddots & \cdot & \cdot & \cdot & \cdot \\ & \cdot & \cdot & \cdot & -1 & 4 & -1 \\ & & & & -1 & & & 3 \end{bmatrix} \in \mathbb{R}^{n \times n}. \quad (13)$$

The matrices $I_{n^2}$ and $I_n$ are identity matrices of sizes $n^2 \times n^2$ and $n \times n$, respectively. From the splitting $K = K_0 + UV^T$ and the above choice of $K_0$, it is obvious that the matrix $UV^T$ encodes the inhibition of membrane molecules due to the discontinuities (if any) in the surface. Given the matrices $K$ and $K_0$, we can show that the choice of the matrices $U$ and $V$ such that $UV^T = K - K_0$ where both matrices $U$ and $V$ are of full column rank is not unique. Our selection method for the matrices $U (= [u_1, \ldots, u_p])$ and $V (= [v_1, \ldots, v_p])$ is as follows. Let each column vector $u_i$, $1 \leq i \leq p$, contain only two nonzero entries $+1$ and $-1$, located at the two neighboring sites separated by the $i$th discontinuity. Then select the column vector $v_i$ to be $-\lambda u_i$, $1 \leq i \leq p$. Thus the matrices $U$ and $V$ are of full column rank $p$ equal to the number of discontinuities. Our selection scheme has the advantage that the column vectors of $U$ and $V$ are very sparse (only two nonzero entries in each column), a structure that is very conducive to the design of an efficient numerical algorithm. We will discuss more on this in Section 3.

The matrix $K_{\text{mem}}$ is a block-tridiagonal matrix. It can be decomposed as the sum of two Kronecker products of matrices $A$ and the identity matrix $I_n$.,
i.e.,

\[ K_{\text{mem}} = A \otimes I_n + I_n \otimes A, \tag{14} \]

with

\[ A = \begin{bmatrix} 1 & -1 \\ -1 & 2 & -1 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ -1 & 2 & -1 \\ -1 & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}, \tag{15} \]

and \( \otimes \) the Kronecker product. Similarly, from the above choice of the matrix \( K_0 \), it can be written as

\[ K_0 = A_0 \otimes I_n + I_n \otimes A_0, \tag{16} \]

where

\[ A_0 = \lambda A + \frac{1}{2} I_n. \tag{17} \]

By using this special structure of the matrix \( K_0 \), we can rewrite any linear system of the form \( K_0 z = f \) as the following Lyapunov matrix equation:

\[ A_0 Z + Z A_0 = F, \tag{18} \]

where \( Z \) and \( F \) are the original \( (n \times n) \) matrices corresponding to the concatenated \( n^2 \times 1 \) vectors \( z \) and \( f \) respectively. Notice that the matrix \( A_0 \) in Equation (18) is a symmetric positive definite tridiagonal matrix. We can use the ADI method to solve this Lyapunov matrix equation in \( O(N) \) operations [7, 6]. We will describe the ADI method in the next section.

As shown at the beginning of this section, the solution to the linear system \( Kx = b \) can be obtained by solving an equivalent linear system \( K_0 x = b' \) with a modified RHS. Thus, instead of solving the original large linear system \( Kx = b \) in Equation (7), our approach is to solve the corresponding Lyapunov matrix equation with a modified RHS matrix \( B' \) (a matrix form of the vector \( b' \)). From Equation (9), the modified RHS \( b' \) can be written as

\[ b' = b - Uy \tag{19} \]
with

\[ \mathbf{Cy} = \mathbf{V}^T \mathbf{K}_0^{-1} \mathbf{b}, \quad (20) \]

where \( \mathbf{y} \in \mathbb{R}^{p\times 1} \). Thus, the modified RHS \( \mathbf{b}' \) can be obtained by solving the auxiliary capacitance-matrix linear system in Equation (20) for \( \mathbf{y} \) and then computing \( \mathbf{b} = \mathbf{Uy} \). The numerical solution of the capacitance-matrix linear system will be discussed in Section 3.

2.1.2. Thin-Plate-Membrane Spline for the Dense-Data Case. The construction of the matrix \( \mathbf{K}_0 \) for the thin-plate–membrane spline problem is based on the above construction for the membrane spline. Analogous to the matrix \( \mathbf{K}_{\text{mem}} \), which is a well-structured matrix and close to the smoothness constraint matrix \( \mathbf{K}_s \) constructed for the membrane problem, a matrix \( \mathbf{K}_{tp} \) is used for the smoothness constraint matrix discretized from the thin-plate smoothness energy. The differentiation of the membrane smoothness energy function corresponds to a Laplacian operator \( \Delta \), while the corresponding operator for the thin-plate smoothness energy is a biharmonic operator \( \Delta^2 \). This matrix \( \mathbf{K}_{tp} \) can be written as the sum of three Kronecker products as follows:

\[ \mathbf{K}_{tp} = \mathbf{A} \mathbf{A} \otimes \mathbf{I}_n + 2 \mathbf{A} \otimes \mathbf{A} + \mathbf{I}_n \otimes \mathbf{A} \mathbf{A}, \quad (21) \]

where the matrix \( \mathbf{A} \) is given in Equation (15) and

\[ \begin{aligned} \bar{\mathbf{A}} = & \begin{bmatrix} 0 & 0 \\ -1 & 2 & -1 \\ \ddots & \ddots & \ddots \\ -1 & 2 & -1 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}. \end{aligned} \quad (22) \]

Note that \( \bar{\mathbf{A}} \) is the same as \( \mathbf{A} \) except for the first and the last rows. Since \( \mathbf{K}^2_{\text{mem}} = \mathbf{A}^2 \otimes \mathbf{I}_n + 2 \mathbf{A} \otimes \mathbf{A} + \mathbf{I}_n \otimes \mathbf{A}^2 \), we can relate \( \mathbf{K}_{tp} \) to \( \mathbf{K}^2_{\text{mem}} \) as follows:

\[ \mathbf{K}_{tp} = \mathbf{K}^2_{\text{mem}} + \mathbf{U}_b \mathbf{V}_b^T, \quad (23) \]

where \( \mathbf{U}_b = -\mathbf{V}_b \) and each column of the matrices \( \mathbf{U}_b \) and \( \mathbf{V}_b \) contains only two nonzero entries 1 and -1, corresponding to the neighboring locations between the first two rows, the first two columns, the last two rows, or the
last two columns. This is due to the fact that the matrix $K_{m\text{em}}^2$ contains not only the thin-plate smoothness constraints all over the discretized rectangular domain but also the membrane smoothness constraints between the first two rows, the first two columns, the last two rows, and the last two columns. The matrix $U_b V_b^T$ is used to disable these imposed membrane smoothness constraints along the boundary in the matrix $K_{m\text{em}}^2$.

The smoothness matrix for the thin-plate-membrane spline is a convex combination of the membrane and thin-plate smoothness matrices. Let's define the matrix $K_{tpm}$ as a convex combination of $K_{m\text{em}}$ and $K_{tp}$, i.e.,

$$K_{tpm} = \sigma K_{tp} + (1 - \sigma)K_{m\text{em}},$$

(24)

where $0 \leq \sigma \leq 1$. In order to apply a fast solver to the linear system with the matrix $K_0$, we choose $K_0$ as follows:

$$K_0 = \lambda \sigma K_{m\text{em}}^2 + \lambda(1 - \sigma)K_{m\text{em}} + I$$

$$= (aK_{m\text{em}} + I)(bK_{m\text{em}} + I),$$

(25)

where

$$a, b = \frac{\lambda(1 - \sigma) + \sqrt{\lambda^2(1 - \sigma)^2 - 4\lambda\sigma}}{2}$$

$$\frac{\lambda(1 - \sigma) - \sqrt{\lambda^2(1 - \sigma)^2 - 4\lambda\sigma}}{2}.$$  

(26)

When $\lambda \geq 4\sigma/(1 - \sigma)^2$, $a$ and $b$ are always real and positive. Then $K_0$ can be factored into two real matrices $aK_{m\text{em}} + I$ and $bK_{m\text{em}} + I$, which are of the same form as $K_0$ for the membrane dense-data problem. Thus, the solution to a linear system with the matrix $K_0$ can be obtained by solving two Lyapunov matrix equations, which can be solved using the ADI method in $O(N)$ time. However, when the above condition is violated, $a$ and $b$ as well as the factoring matrices in Equation (25) become complex. It is more complicated to use the ADI method to solve the complex Lyapunov matrix equations and analyze the computational complexity. Therefore, in this paper, we limit our discussion to the case when the aforementioned condition is satisfied for the dense-data surface-smoothing problem.

Using the above choice for the matrix $K_0$, we can see that the matrices $U$ and $V$ encode the inhibition of membrane and thin-plate computational
molecules due to the presence of depth and orientation discontinuities (if any) and the imposed membrane smoothness along the boundary in $K_{\text{mem}}^2$. Consequently, we can decompose the matrix $UV^T$ as $\lambda \sigma U_b V_b^T + \lambda \sigma U_{\text{tp}} V_{\text{tp}}^T + \lambda (1 - \sigma) U_{\text{mem}} V_{\text{mem}}^T$, where $U_b$ and $V_b$ correspond to the inhibition of the membrane molecules due to the imposed membrane smoothness along the boundary in $K_{\text{mem}}^2$, where $U_{\text{mem}}$ and $V_{\text{mem}}$ correspond to the inhibition of the membrane molecules due to depth discontinuities, and where $U_{\text{tp}}$ and $V_{\text{tp}}$ correspond to the inhibition of the thin-plate molecules due to depth and orientation discontinuities. The structures of the matrices $U_b$, $V_b$, $U_{\text{mem}}$, and $V_{\text{mem}}$ have already been discussed. To give a picture of the structures of the matrices $U_{\text{tp}}$ and $V_{\text{tp}}$, we consider the case that the orientation discontinuity locations are the same as the depth discontinuity locations. Then we have

$$U_{\text{tp}} V_{\text{tp}}^T = (K_{\text{mem}} + U_{\text{mem}} V_{\text{mem}}^T)^2 - K_{\text{mem}}^2$$

$$= (K_{\text{mem}} U_{\text{mem}}) V_{\text{mem}}^T + U_{\text{mem}} (K_{\text{mem}} V_{\text{mem}})^T$$

$$+ U_{\text{mem}} (V_{\text{mem}}^T U_{\text{mem}}) V_{\text{mem}}^T.$$

(27)

Note that the $p \times p$ matrix $V_{\text{mem}}^T U_{\text{mem}}$ in the bracket of the third term in Equation (27) is very sparse, since each column of $U$ and $V$ contains only two nonzero entries, +1 and −1, mapped to the discontinuity locations. Each diagonal entry is the inner product of the vector $u_i$ and the vector $v_i (=-u_i)$ and thus has a value −2. The $(i,j)$th off-diagonal entry is nonzero when there is overlap between the nonzero locations corresponding to the $i$th discontinuity and those corresponding to the $j$th discontinuity. Let’s denote the matrix $(V_{\text{mem}}^T U_{\text{mem}}) V_{\text{mem}}^T$, which contains very sparse row vectors, by $\tilde{V}_{\text{mem}}^T$; then we have

$$UV^T = \lambda \{ \sigma U_b V_b^T + \sigma U_{\text{tp}} V_{\text{tp}}^T + (1 - \sigma) U_{\text{mem}} V_{\text{mem}}^T \}$$

(28)

$$= \lambda \sigma \{ U_b V_b^T + (K_{\text{mem}} U_{\text{mem}}) V_{\text{mem}}^T + U_{\text{mem}} (K_{\text{mem}} V_{\text{mem}})^T + U_{\text{mem}} \tilde{V}_{\text{mem}}^T \}$$

$$+ \lambda (1 - \sigma) U_{\text{mem}} V_{\text{mem}}^T.$$

(29)

To assemble the matrices $U$ and $V$ based on Equation (29) with the restriction that $U$ and $V$ are of full column rank, we have the following choice
for $\mathbf{U}$ and $\mathbf{V}$:

$$
\mathbf{U} = \begin{bmatrix}
\lambda \sigma \mathbf{U}_b & \lambda \sigma \mathbf{K}_{\text{mem}} \mathbf{U}_{\text{mem}} & \mathbf{U}_{\text{mem}}
\end{bmatrix},
$$

(30)

$$
\mathbf{V} = \begin{bmatrix}
\mathbf{V}_b & \mathbf{V}_{\text{mem}} & \lambda \sigma (\mathbf{K}_{\text{mem}} \mathbf{V}_{\text{mem}} + \mathbf{\bar{V}}_{\text{mem}}) + \lambda (1 - \sigma) \mathbf{V}_{\text{mem}}
\end{bmatrix}.
$$

(31)

The column vectors of the matrices $\mathbf{U}$ and $\mathbf{V}$ in Equations (30) and (31) are very sparse, since the matrices $\mathbf{U}_b$, $\mathbf{V}_b$, $\mathbf{U}_{\text{mem}}$, $\mathbf{V}_{\text{mem}}$, and $\mathbf{\bar{V}}_{\text{mem}}$ contain very sparse column vectors and the matrix $\mathbf{K}_{\text{mem}}$ contains at most five nonzero entries in each row and each column. We will make use of this sparsity property of $\mathbf{U}$ and $\mathbf{V}$ to design very efficient numerical solution in Section 3.

2.2. Sparse-Data Surface Smoothing

In this subsection, the matrix $\mathbf{K}_0$ for the membrane-spline sparse-data surface-smoothing problem will be constructed first. Then, the construction will be extended to the cases of thin-plate and thin-plate-membrane splines.

2.2.1. Membrane Spline for the Sparse-Data Case. For the sparse-data surface smoothing using a membrane spline over an embedded rectangular domain $\Omega$, we choose the matrix $\mathbf{K}_0$ to be the sum of the membrane smoothness matrix $\mathbf{K}_{\text{mem}}$ and a rank-one matrix $\mathbf{u}_0 \mathbf{u}_0^T$, where $\mathbf{u}_0$ is the eigenvector in the null space of $\mathbf{K}_{\text{mem}}$. This choice is inspired by the rank augmenting method for singular matrices in the capacitance-matrix method described in [11] and yields a nonsingular matrix with nice properties, which transforms the problem of solving the linear system $\mathbf{Kx} = \mathbf{b}$ into solving a Lyapunov matrix equation.

To obtain the solution of the linear system $\mathbf{Kx} = \mathbf{b}$ by solving a Lyapunov matrix equation, we first split the matrix $\mathbf{K}_s$ into two components.

$$
\mathbf{K}_s = \mathbf{K}_{\text{mem}} + \mathbf{U}_s \mathbf{V}_s^T,
$$

(32)

where

$$
\mathbf{U}_s = \begin{bmatrix}
\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_k
\end{bmatrix},
$$

$$
\mathbf{V}_s = \begin{bmatrix}
\mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_k
\end{bmatrix},
$$

and $\mathbf{K}_{\text{mem}}$ is given in Equation (11). Each of the column vectors $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^{n^2 \times 1}$, $1 \leq i \leq k$, contains only one or two nonzero entries mapped to the
SURFACE SMOOTHING PROBLEM 217

boundary of the discretized \( \Omega \) or the discontinuity locations. As discussed in Section 2.1.1, the matrix \( \mathbf{K}_{\text{mem}} \) can be decomposed as the sum of two Kronecker products of matrices \( \mathbf{A} \) and \( \mathbf{I} \).

Note that \( \mathbf{K} = \lambda \mathbf{K}_{\text{mem}} + \mathbf{K}_d \). For the sparse-data case, the diagonal matrix \( \mathbf{K}_d \) contains sparsely distributed nonzero entries corresponding to data locations. Since the matrix \( \mathbf{K}_{\text{mem}} \) is singular with rank-one deficiency, we can choose the nonsingular matrix \( \mathbf{K}_0 \) to be the sum of the matrices \( \lambda \mathbf{K}_{\text{mem}} \) and \( \lambda \mathbf{u}_0 \mathbf{u}_0^T \), where \( \mathbf{u}_0 = (1/n)[1 \ 1 \ \cdots \ 1]^T \in \mathbb{R}^{N \times 1} \) is the eigenvector in \( \text{null}(\mathbf{K}) \).

Thus every nonzero entry of the sparse diagonal matrix \( \mathbf{K}_d \) is decomposed into the outer product of \( \mathbf{u}'_i \) and \( \mathbf{v}'_i \) and included in \( \mathbf{U} \) and \( \mathbf{V} \) as follows:

\[
\mathbf{U} = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_k & \mathbf{u}'_1 & \cdots & \mathbf{u}'_k \end{bmatrix},
\]

\[
\mathbf{V} = \begin{bmatrix} -\lambda \mathbf{u}_0 & \lambda \mathbf{v}_1 & \cdots & \lambda \mathbf{v}_k & \mathbf{v}'_1 & \cdots & \mathbf{v}'_k \end{bmatrix}.
\]

With the above choice of \( \mathbf{U} \) and \( \mathbf{V} \), all columns of \( \mathbf{U} \) and \( \mathbf{V} \) except the first columns are still very sparse, containing only one or two nonzero entries. Our numerical algorithm will take advantage of this sparsity.

The matrix \( \mathbf{K}_0 \) is nonsingular due to the rank-augmenting matrix \( \lambda \mathbf{u}_0 \mathbf{u}_0^T \); thus the Sherman-Morrison-Woodbury formula can be applied here to obtain the solution for the linear system with the matrix \( \mathbf{K} \) by solving the linear system having \( \mathbf{K}_0 \) with a modified RHS. Since the column spaces of the two matrices \( \lambda \mathbf{K}_{\text{mem}} \) and \( \lambda \mathbf{u}_0 \mathbf{u}_0^T \) in \( \mathbf{K}_0 \) are complementary, the solution to a linear system of the form \( \mathbf{K}_0 \mathbf{z} = \mathbf{f} \) can be decoupled into \( \mathbf{z}_0 + \mathbf{z}_1 \), where \( \mathbf{z}_0 \) and \( \mathbf{z}_1 \) are the minimum-norm solutions to the linear systems with the matrices \( \lambda \mathbf{u}_0 \mathbf{u}_0^T \) and \( \lambda \mathbf{K}_{\text{mem}} \), respectively, and with the same RHS \( \mathbf{f} \). Thus the solution \( \mathbf{z}_0 \) can be easily computed from \( (\mathbf{f}^T \mathbf{u}_0 / \lambda) \mathbf{u}_0 \). With the special structure of the matrix \( \mathbf{K}_{\text{mem}} \), the solution \( \mathbf{z}_1 \) can be obtained by solving the equivalent Lyapunov matrix equation \( \mathbf{A} \mathbf{Z}_1 + \mathbf{Z}_1 \mathbf{A} = \mathbf{F}_1 \), where \( \mathbf{Z}_1 \) is the matrix form of the solution \( \mathbf{z}_1 \) and \( \mathbf{F}_1 \) is the matrix form of the vector \( \mathbf{f} \) after the projection onto the column space of the matrix \( \lambda \mathbf{K}_{\text{mem}} \).

2.2.2. Thin-Plate Spline for the Sparse-Data Case. The extension of the construction of \( \mathbf{K}_0 \) from the membrane spline to the thin-plate spline for sparse-data surface smoothing is similar to that for the dense-data case. As discussed in Section 2.1.2, the matrix \( \mathbf{K}_{\text{tp}} \) for the thin-plate smoothness constraint all over the domain can be written as \( \lambda \mathbf{K}_{\text{mem}}^2 + \mathbf{U}_0 \mathbf{V}_h^T \). The treatment for the sparse matrix \( \mathbf{K}_d \) is the same as that described in Section 2.2.1, i.e., each sparse nonzero entry of the matrix \( \mathbf{K}_d \) situated along its diagonal corresponding to the data location is decomposed into the outer product of two vectors, with each containing only one nonzero entry, and is incorporated into \( \mathbf{UV}^T \). Similarly, the matrix \( \mathbf{K}_0 \) is chosen as \( \lambda (\mathbf{K}_{\text{mem}}^2 + \mathbf{u}_0 \mathbf{u}_0^T) \). Thus, the
solution to a linear system with the matrix $K_0$ consists of two terms. The first term is simply the scaled projection of the RHS onto the vector $u_0$. The second term is a minimum-norm solution to a linear system with the matrix $\lambda K_{mem}^2$, which can be obtained by solving two cascaded Lyapunov matrix equations. This is because the matrix $\lambda K_{mem}^2$ can be decomposed into a product of two matrices $\lambda K_{mem}$ and $K_{mem}$, each of which can be written as the scaled sum of two Kronecker products of $A$ and $I$ as given in Equation (14). Thus, the solution to a linear system with the matrix $K_0$ can be obtained by solving two Lyapunov matrix equations. The construction of the matrices $U$ and $V$ in this case is similar to that described in Section 2.1.2.

2.2.3. Thin-Plate-Membrane Spline for the Sparse-Data Case. The thin-plate-membrane stabilizer is a convex combination of the membrane and thin-plate stabilizers; thus a matrix $K_{tpm}$ is defined as $K_{tpm} := \sigma K_{tp} + (1 - \sigma)K_{mem}$. Using the same treatment for the matrix $K_d$ for sparse-data case and that for the matrix $K_{tp}$ discussed above, we choose the matrix $K_0$ as follows:

$$K_0 = \lambda \sigma K_{mem}^2 + \lambda(1 - \sigma)K_{mem} + u_0u_0^T.$$  \hfill (33)

The solution to a linear system with the matrix $K_0$ is similar to that described in Section 2.2.2. The matrix $\lambda \sigma K_{mem}^2 + \lambda(1 - \sigma)K_{mem}$ in Equation (33) can be factored into the product of two matrices $\lambda K_{mem}$ and $\sigma K_{mem} + (1 - \sigma)I$, both of which can be written in the form of the RHS of Equation (14) with the matrix $A$ replaced by

$$\lambda A \quad \text{and} \quad \sigma A + \frac{1 - \sigma}{2}I$$

respectively. Thus the solution to a linear system with the matrix $K_0$ can be obtained by solving a cascade of two Lyapunov matrix equations. The construction of the matrices $U$ and $V$ is similar to that in the previous sections, and the column vectors in both matrices are still very sparse.

3. NUMERICAL SOLUTION

Our numerical algorithm can be described as follows:

1. Solve $K_0\tilde{x} = b$ for $\tilde{x}$ using the ADI method.
2. Form the capacitance matrix $C := I + V^TK_0^{-1}U$. 


3. Solve $Cy = V^T\hat{\mathbf{x}}$ using bi-conjugate-gradient (BCG) algorithm or Gaussian elimination (when the size of the matrix $C$ is small), and compute the modified RHS $b' = b - U\mathbf{y}$.

4. Solve $K_0\mathbf{x} = b'$ for $\mathbf{x}$ using the ADI method.

In steps 1 and 4, we need to solve linear systems with the well-structured matrix $K_0$. With our construction of $K_0$, the solution to the original linear system is equivalent to the solution of one or two cascaded Lyapunov matrix equations with a modified RHS. The ADI method is used to solve these Lyapunov matrix equations. The formation of the capacitance matrix $C$ is accomplished by using the sparsity of the column vectors in $U$ and $V$ and a table lookup. The linear system with the capacitance matrix in step 3 is dense and nonsymmetric and needs to be solved very efficiently. Our solution to this linear system is obtained by using the BCG algorithm [3] if the size of the capacitance matrix is large, or Gaussian elimination [4] if it is small.

In the following, the details of the ADI method for the Lyapunov equation, the formation of matrix $C$, and the solution to the linear system $Cy = V^T\hat{\mathbf{x}}$ will be given.

3.1. ADI Method for the Lyapunov Equation (Stages 1 and 4 of the Algorithm)

The ADI method for solving a Lyapunov matrix equation is described in [9]. For any Lyapunov matrix equation of the form $AX + XA = B$, the ADI method involves the following steps in each iteration $j = 1, 2, \ldots, J$:

$$\begin{align*}
(A + p_j I)X_{j-1/2} &= B - X_{j-1}(A - p_j I), \\
X_j(A + p_j I) &= B - (A - p_j I)X_{j-1/2}
\end{align*}$$

The nice properties of the matrix $A$ help to advance the ADI iterations very efficiently in each step of the technique. Since $A$ is SPD and tridiagonal, we can use Cholesky decomposition for the tridiagonal SPD matrix $A + p_j I$ to compute the updated solution. This solution update requires only $O(N)$ time per iteration.

We now examine the convergence of the ADI method for the problems in stages 1 and 4 of our algorithm. Let the initial $X_0 = 0$ (zero matrix); let $\Delta X_j = X_j - X^*$, where $X^*$ is the true solution of the Lyapunov matrix equation; and let $\lambda_0, \lambda_1, \ldots, \lambda_{n-1}$ and $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1}$ be the eigenvalues and the eigenvectors of $A$ respectively. For our particular matrix $A$, as given
in Equation (15) or (17), we can write it in a similarity transformation, i.e.,

\[ A = QDQ^T. \]  

(36)

where

\[ Q = \begin{bmatrix} q_0 & q_1 & \cdots & q_{n-1} \end{bmatrix}, \]

\[ D = \text{diag}[\lambda_0, \lambda_1, \ldots, \lambda_{n-1}], \]

\[ q_0 = \sqrt{\frac{1}{n}} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \]

\[ q_k = \sqrt{\frac{2}{n}} \begin{bmatrix} \cos\left(\frac{k\pi}{n} - \frac{k\pi}{2n}\right) \\ \cos\left(\frac{2k\pi}{n} - \frac{k\pi}{2n}\right) \\ \vdots \\ \cos\left(\frac{nk\pi}{n} - \frac{k\pi}{2n}\right) \end{bmatrix} \text{ for } k = 1, \ldots, n-1, \]

where \( \lambda_k = 2 - 2\cos(k\pi/n), k = 0, \ldots, n-1, \) for the matrix \( A \) in Equation (15), and \( \lambda_k = \frac{1}{2} + \lambda[2 - 2\cos(k\pi/n)], k = 0, \ldots, n-1, \) for the matrix \( A_0 \) in Equation 17. Notice that \( Q \) is an orthogonal matrix, i.e., \( QQ^T = I_n \).

By taking the tensor product of the eigenvectors, \( q_k \otimes q_l \), \( k, l = 1, \ldots, n \), we can generate an orthogonal basis for \( \mathbb{R}^{n\times n} \) and express the true solution \( X^* \) in this basis as follows:

\[ X^* = \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} a_{kl} (q_k \otimes q_l). \]

(37)

Subtracting each of the equations (34) and (35) from the Lyapunov equation \( AX + XA = B \), and expressing \( X \) in the above tensor-product basis, we have
the following result for the error after $t$ iterations:

$$\Delta X_t = - \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} \left[ \prod_{j=1}^{t} \frac{\lambda_k - p_j}{\lambda_k + p_j} \right] \frac{\lambda_l - p_j}{\lambda_l + p_j} a_{k,l}(q_k \otimes q_l). \quad (38)$$

The error $\Delta X_t$ is bounded by

$$|\Delta X_t| \leq \mu_t |X^*|, \quad (39)$$

where

$$\mu_t = \max_{k,l, \lambda_k, \lambda_l} \left\{ \prod_{j=1}^{t} \left( \frac{\lambda_k - p_j}{\lambda_k + p_j} \right) \left( \frac{\lambda_l - p_j}{\lambda_l + p_j} \right) \right\}. \quad (40)$$

Obviously, the error bound depends on the choice of the ADI parameters $p_j$. Choosing the optimal set of $p_j$'s leads to the ADI minimax parameter estimation problem. Given the bounds for the eigenvalues of the SPD matrix $A$ as, $\forall k$, $0 < a \leq \lambda_k \leq b$, we have

$$\mu_t \leq \min_{p_1, \ldots, p_t} \max_{0 < a \leq \lambda \leq b} \prod_{j=1}^{t} \left( \frac{\lambda - p_j}{\lambda + p_j} \right)^2. \quad (40)$$

The RHS is the minimax ADI parameter problem. The solution to this minimax problem gives the optimum set of ADI parameters. Computing the optimal choice of the ADI parameters is described in [9].

The minimax analysis also provides the number of iterations needed in ADI to achieve a specified error tolerance $\epsilon$ in $X$ such that $\|\Delta X_j\| \leq \epsilon \|X^*\|$. This number of iterations is given by

$$J = \left\lceil \frac{\ln(4/\epsilon) \ln(4/k^{'})}{\pi^2} \right\rceil, \quad (41)$$

where

$$k^{' \, \prime} = \frac{1}{m + \sqrt{m^2 - 1}}, \quad m = \frac{1}{2} \left( \kappa_2(A) + \frac{1}{\kappa_2(A)} \right).$$
\([z]\) denotes the smallest integer larger than \(z\), and \(\kappa_2\) is the ratio of the largest and smallest eigenvalues, i.e., the 2-condition number of \(A\).

The Lyapunov matrix equations that need to be solved as discussed in Section 2 are from the linear systems with the matrices either of the form \(aK_{mem} + bI\) or of the form \(aK_{mem}\), where \(a\) and \(b\) are positive constants. The former represents a discrete parabolic operator, while the latter is a representation of a discrete elliptic operator. In the discrete parabolic operator case, the corresponding condition number of \(A\) is bounded by a constant. From Equation (41), we can see that the number of iterations needed is bounded by a constant number for a given error tolerance. For the latter, when Frobenius-norm [4] error reduction is used instead of the absolute termwise error reduction in Equation (39), we can show that the number of iterations needed for a given error tolerance is bounded by a constant number under the mild assumption that the spectrum of the RHS is upper bounded by a function of \(\|\omega\|_2\) (\(\omega\) is the frequency-domain vector-valued variable) of degree less than 1 (see [7, 6] for details). Hence, the number of iterations needed in ADI for a given error tolerance is independent of the size of the problem for both cases. Thus, the computational time for the ADI method to solve the Lyapunov matrix equation is \(O(N)\).

3.2. Formation of the Matrix \(C\) (Stage 2)

Each entry \((i, j)\) in \(C\) can be written as

\[
C_{(i, j)} = v_i^T K_0^{-1} u_j + \delta_{ij}. \tag{42}
\]

Recall that each of the vectors \(v_i\) and \(u_j\) contains very sparse nonzero elements. This means that \(v_i^T K_0^{-1} u_j\) is the sampling at the nonzero locations of \(v_i\) from the impulse response of the linear system \(K_0^{-1}\) or the Lyapunov system with the impulses situated at the nonzero locations of \(u_j\). Unfortunately, this Lyapunov linear system is not shift-invariant, and this makes the problem of efficiently forming the matrix \(C\) more difficult.

For ease of exposition, let's consider vectors \(v_i\) and \(u_j\) containing single nonzero entries, located at \((p_i, q_i)\) and \((p_j, q_j)\), respectively. We can find an analytic form of the solution \(X\) at \((p, q)\) to the Lyapunov system (one Lyapunov matrix equation or a cascade of two Lyapunov matrix equations) with the input matrix \(B\) containing only one nonzero entry, i.e. \(B(k, l) = \delta_{k-p, l-q}\), by using the singular-value decomposition as in Equation (36) for the Lyapunov matrix equations involved in the Lyapunov system. This leads
to

\[ X(p_i, q_i) = f_n(p_i - p_j, q_i - q_j) + f_n(p_i - p_j, q_i + q_j - 1) \]

\[ + f_n(p_i + p_j - 1, q_i - q_j) + f_n(p_i + p_j - 1, q_i + q_j - 1), \]

(43)

where the function \( f_n \) is of the following form:

\[ f_n(p, q) = \frac{1}{n^2} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} \frac{\cos(k \pi/n) \cos(l \pi/n)}{\lambda_{kl}}. \]

(44)

The values in \( \lambda_{kl} \) depend on the type of data constraint (sparse or dense) and the smoothness constraint used. For the sparse-data surface smoothing using a membrane spline, we have \( \lambda_{kl} = \lambda \) when \( k = l = 0 \) and

\[ \lambda_{kl} = \lambda \left( 4 - 2 \cos \left( \frac{k \pi}{n} \right) - 2 \cos \left( \frac{l \pi}{n} \right) \right) = \lambda \left( 4 \sin^2 \frac{k \pi}{2n} + 4 \sin^2 \frac{l \pi}{2n} \right) \]

elsewhere. For the thin-plate spline case, we have \( \lambda_{kl} = \lambda \) when \( k = l = 0 \), and otherwise

\[ \lambda_{kl} = \lambda \left( 4 - 2 \cos \left( \frac{k \pi}{n} \right) - 2 \cos \left( \frac{l \pi}{n} \right) \right)^2 = \lambda \left( 4 \sin^2 \frac{k \pi}{2n} + 4 \sin^2 \frac{l \pi}{2n} \right)^2. \]

Note that the calculation of the function \( f_n(p, q) \) using the above formula may lead to numerical instabilities for very large \( n \). For sparse data, some of the terms in the summation involve subtraction of two close numbers in the denominator when \( k \) and \( l \) are very small and \( n \) is very large. This problem can be avoided by replacing the factor \( 4 - 2 \cos(k \pi/n) - 2 \cos(l \pi/n) \) with \( 4 \sin^2(k \pi/2n) + 4 \sin^2(l \pi/2n) \). Numerical instability due to division by a very small number (for very large problem sizes) can also be an issue in the above formula, and this needs to be investigated further. In our experiments on sparse-data surface smoothing using membrane splines and thin-plate-membrane splines, we did not observe any bad effects on the accuracy of the solution from numerical instabilities for problem sizes of \( 64 \times 64 \) and \( 128 \times 128 \), which are common sizes for surface smoothing problems in computer vision.
The function \( f_n(p, q) \), for \( 0 \leq p, q \leq n \), should be precomputed and stored as an \((n + 1) \times (n + 1)\) lookup table. Then, each entry in the matrix \( C \) can be efficiently calculated from the above equation and the lookup table. Once the lookup table is available, the formation of the capacitance matrix takes only \( O(N) \) operations.

### 3.3. Solution to \( Cy = V^T \tilde{x} \) (Stage 3)

The linear system \( Cy = V^T \tilde{x} \) is nonsymmetric, and \( C \) is a dense \( p \times p \) matrix. For the dense-data case, the size of \( C \) is determined by the number of discontinuities. For the sparse-data case it depends on the number of boundary points, the discontinuity locations, and the number of data points. A reasonable assumption is that the size is approximately \( O(n) \times O(n) \). Consequently, it is important to have a fast algorithm to compute the solution \( y \) to this linear system. When the size of the capacitance matrix is small, we use Gaussian elimination to solve the linear system. Otherwise, we use the bi-conjugate-gradient (BCG) method.

As in CG, the operations and storage requirements per step in BCG are constant. The convergence behavior is similar to that of the CG method, except that the BCG algorithm is susceptible to breakdowns and numerical instabilities. To be more specific, division by 0 (or a number very close to 0) may occur inside the BCG iterations. These breakdowns may be avoided by using the lookahead Lanczos algorithm or the quasiminimal residual (QMR) algorithm [3].

In the BCG algorithm, two matrix-vector multiplications are required in each iteration. Since the matrix \( C \) is dense, the multiplication takes \( O(p^2) \) operations. When the matrix \( C \) is \( O(n) \times O(n) \), the matrix-vector multiplication will take \( O(N) \) \((N = n^2)\) operations, which is expensive for just one iteration. The rate of convergence for the BCG algorithm depends on the condition number of the matrix \( C \). In the worse case, it could take \( p \) iterations, which would lead to an \( O(p^3) \) algorithm. However, the rate of convergence for a CG-type algorithm usually can be improved with appropriate preconditioning. Proskurowski and Vassilevski [12] applied several preconditioning techniques for the capacitance matrix in the domain decomposition (DD) to the capacitance matrix in domain imbedding (DI) for second-order elliptic problems, in view of the observation that DD and DI are usually complementary, i.e., the capacitance matrix in DI is the inverse of the capacitance matrix in DD. Since several preconditioners have been well developed for the capacitance matrix in DD, Proskurowski and Vassilevski [12] regarded them as the inverses of the preconditioners in DI. Among them, Dryja's preconditioner [1] is based on the square root of the finite-difference analog of \(-d^2/ds^2\) along the boundary of the domain. Dryja proved
that this preconditioner is spectrally equivalent to the capacitance matrix for the second-order elliptic problem, which means the iterative algorithm can converge in a constant number of iterations with this preconditioning. Smith and Widlund [16] used the hierarchical basis preconditioner [21] for the capacitance matrix in second-order elliptic problems. Unfortunately, these preconditioners cannot be directly applied to the capacitance matrix arising from the surface smoothing problem presented in this paper. The extension of these preconditioning techniques to the capacitance matrix used in this paper will be the focus of our future research.

4. EXPERIMENTAL RESULTS

In this section, we present examples of the algorithm performance on sparse-data surface smoothing problems. We demonstrate in experiments the computational efficiency of our algorithm compared to the conjugate gradient (CG) algorithm and the hierarchical-basis preconditioned conjugate-gradient (HBCG) algorithm [17, 21]. These algorithms were tested on 64 × 64 and 128 × 128 discretization meshes. The input data set is very sparse and contains only 15 data points randomly distributed in the plane, as shown in Figure 2. The discontinuity locations are specified along the line between coordinates (1, 32) and (30, 32) for the 64 × 64 mesh and between coordinates (1, 64) and (60, 64) for the 128 × 128 mesh.

In our algorithm, we first compute the modified RHS by solving the capacitance matrix linear system using the BCG algorithm. The modified RHS is shown in Figure 2(b). This depicts the changes that are made to the original data before applying the ADI method. Note the changes along the discontinuities, and at the data locations. The ADI iterations are then employed to obtain the interpolated surface. From Equation (41), it is evident that just 16 iterations can guarantee the error tolerance \( \varepsilon \) to be less than \( 10^{-6} \). We depict the surfaces after 1 and 16 ADI iterations in Figure 2(c) and (d).

From Figure 2(c), we can see that just one ADI iteration can produce the global shape of the exact surface. This is because we advance the ADI iterations with the parameters \( p_j \) starting with a small value and proceeding toward large values. The smaller parameters are used to recover the low-frequency detail of the surface, and the large values refine the shape with high-frequency components. Consequently, it is not surprising that the global shape was recovered in one iteration.

The experiments on sparse-data surface smoothing using membrane splines were performed on 64 × 64 and 128 × 128 problems. We measured
the CPU time required on a Sun SPARC-20 workstation for the CG, HBCG, and our capacitance-matrix method, respectively, to obtain a solution with relative 2-norm error below $10^{-4}$. The required CPU time for these three methods is listed in Table 1. Our capacitance-matrix method outperforms the other two methods in terms of efficiency, even though we did not use any preconditioning for the capacitance matrix in the BCG algorithm. For the $64 \times 64$ membrane smoothing problem, it takes 26 BCG iterations to obtain a solution with relative error less than $10^{-5}$ for the linear system with the $46 \times 46$ capacitance matrix. For the $128 \times 128$ problem, it takes 34 BCG iterations to converge within the error tolerance $10^{-6}$ for the capacitance matrix of size $76 \times 76$. Our capacitance-matrix method can be speeded up even more by appropriate preconditioning of the capacitance matrix.
The next experiment is on thin-plate-membrane surface smoothing with given discontinuity for the sparse data set in Figure 2(a). The weighting coefficient $\sigma$ for the combination of thin-plate and membrane smoothness is here set to 0.9. For solving the linear system with the matrix $K_0$, we solve a cascade of two Lyapunov matrix equations by using the ADI method for each one. After computing the modified RHS, we depict the solution obtained while solving the second Lyapunov matrix equation after 1 and 16 iterations in Figure 3. The phenomenon of global-to-local shape recovery is also evident in these thin-plate-membrane examples.

We compared our capacitance-matrix method with the CG and HBCG algorithms in terms of the CPU time required on a Sun SPARC-20 workstation to obtain a solution with a relative 2-norm error of $10^{-3}$. Table 2 depicts the CPU-time results for the $64 \times 64$ and $128 \times 128$ problems. As is evident from the table, our algorithm outperforms CG and HCG, even though we did not use any preconditioning technique in the BCG part of our algorithm for

<table>
<thead>
<tr>
<th>Size ($n \times n$)</th>
<th>CG</th>
<th>HBCG</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$64 \times 64$</td>
<td>2.37</td>
<td>1.08</td>
<td>0.30</td>
</tr>
<tr>
<td>$128 \times 128$</td>
<td>20.27</td>
<td>6.55</td>
<td>1.60</td>
</tr>
</tbody>
</table>

Fig. 3. Example of thin-plate-membrane spline surface smoothing: (a) solution after 1 iteration of ADI method, (b) after 16 iterations.
TABLE 2
CPU TIME REQUIRED FOR SPARSE-DATA SURFACE SMOOTHING PROBLEMS USING
THIN-PLATE–MEMBRANE SPLINE

<table>
<thead>
<tr>
<th>Size (n × n)</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG</td>
</tr>
<tr>
<td>64 × 64</td>
<td>32.33</td>
</tr>
<tr>
<td>128 × 128</td>
<td>297.29</td>
</tr>
</tbody>
</table>

solving the capacitance-matrix linear system, which occupied roughly 85% of the total CPU time in this experiment. For the 64 × 64 problem, our capacitance-matrix method involves 26 ADI iterations for stages 1 and 4, respectively, and 63 BCG iterations in stage 3 for solving the linear system with a capacitance matrix of size 332 × 332. For solving the 128 × 128 problem, it takes 30 ADI iterations for stages 1 and 4 of our algorithm, respectively, and 110 BCG iterations in stage 3 for a capacitance matrix of size 648 × 648. The stiffness matrix \( K \) for the sparse-data thin-plate–membrane spline smoothing problem is very ill-conditioned, and hence all these algorithms take more time to converge than for the membrane spline smoothing problem, where \( K \) is far better conditioned.

5. CONCLUSION

In this paper, we have presented a new and fast algorithm for solving the sparse- and dense-data surface smoothing problems using a membrane, thin-plate, or thin-plate–membrane spline. The problems are formulated as solutions to a Lyapunov matrix equation or a cascade of two Lyapunov matrix equations with an appropriate RHS by using the Sherman-Morrison-Woodbury formula of matrix analysis. The RHS is obtained by solving the capacitance-matrix linear system using the BCG algorithm. To solve the Lyapunov matrix equations, we use the ADI algorithm and obtain convergence (for a specified tolerance) in a constant number of iterations. Unlike other existing methods, the performance of our algorithm is not sensitive to prespecified discontinuities in the underlying surface, since the effect of discontinuities is accounted for in the modification of the RHS. The efficiency of our algorithm is demonstrated through experiments on sparse-data surface smoothing, in comparisons with the CG and HBCG algorithms.

REFERENCES


Received 15 May 1995; final manuscript accepted 30 October 1996