AN EFFICIENT DATA ASSIMILIATION ALGORITHM WITH A GAUSSIAN COVARIANCE STRUCTURE

by

Ranjit M. Passi
Kent Goodrich
John C. Derber
Mark Limber

TR-2/94
Approved for public release; distribution is unlimited October 1993

The Center for Ocean & Atmospheric Modeling (COAM) is operated by The University of Southern Mississippi under sponsorship of the Department of the Navy, Office of the Chief of Naval Research. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

AN EFFICIENT DATA ASSIMILATION ALGORITHM WITH A GAUSSIAN COVARIANCE STRUCTURE

Ranjit M. Passi
Center for Ocean & Atmospheric Modeling
Stennis Space Center, MS 39529

Kent Goodrich
Department of Mathematics
University of Colorado, Boulder CO 80309

John C. Derber
National Weather Service/NOAA
World Weather Building, Washington D.C.20233

Mark Limber
Department of Mathematics
University of Colorado, Boulder CO 80309

ABSTRACT

To adequately describe the general circulation of the ocean requires a fine spatial resolution and extremely small time steps for model integration. This combination leads to large in-core computer memories and computer processing time for the model integration. In addition, the numerical models have to be coupled with data assimilation schemes to derive accurate initial conditions computing model forecasts. To make the nowcasting and forecasting practical, it is necessary that the data assimilation schemes be computationally efficient and parsimonious in computer in-core memories.

The work reported in this technical report was initiated at the Institute for Naval Oceanography (INO) to develop data assimilation procedures that are efficient and are widely applicable. Our initial data assimilation work was done with a scheme which was developed by Derber and Rosati (1989) for their global data assimilation application. The scheme is quite efficient; but when it was combined with a primitive equation, numerical model of the Gulf of Mexico, the nowcasting step took too much computer time for it to be useful in real-time applications. So, after an initial implementation and test of the scheme, our obvious thrust was to enhance and make it more efficient.

Most of the computation resources while using the Derber-Rosati scheme are required in approximating a matrix product $\Sigma_m g$, where Σ_m is the covariance matrix of the model output error having a Gaussian spatial covariance structure that stays fixed, and g is a varying vector. They approximate this product by repeated applications of a Laplacian operator.

A new algorithm is developed that, while maintaining the accuracy, approximates $\Sigma_m g$ far more efficiently than the Laplacian algorithm. It also admits a wider class of covariance structures with equal ease and efficiency. The new algorithm approximates $\Sigma_m g$ by applying a product of two operators that are polynomials in simple averaging operators. The algorithm is efficient and general with the only restriction that Σ_m be based on a covariance function that is a product of factors, which are even functions in a single dimension.

The purpose of this technical report is to provide details of the Derber-Rosati algorithm, which has since found a much wider acceptance in the community. It has provided insight into the approximation of $\Sigma_m g$ that led to the development of the new algorithm. Details of the new algorithms are also given; and using computer simulations, the approximation efficiencies of the two algorithms are compared with the actual matrix multiplication, $\Sigma_m g$.

1. INTRODUCTION

Nowcasting and forecasting are important activities in meteorology. Nowcasting combines the output of a numerical forecast model with observations to provide initial conditions, that are in dynamical balance, used by the model to calculate the forecast. Now that the numerical models of the general circulation of the ocean have matured, these two activities are playing an important role in oceanography to provide a realistic description of the various oceanic regions.

To adequately describe the general circulation of the ocean requires a fine spatial resolution and extremely small time steps in numerical model integration. This combination leads to large in-core computer memories and computer processing time for the model integration. In addition, the numerical models have to be combined with data assimilation schemes to derive accurate initial conditions which are used by the model to compute forecasts. To make the nowcasting and forecasting practical, it is necessary that numerical modeling and data assimilation schemes be computationally efficient and not require inordinately large computer in-core memories.

Data assimilation procedures that are efficient and are widely applicable, as reported in this technical report, were developed at the Institute for Naval Oceanography (INO). Initial data assimilation work was done using an algorithm reported by Derber and Rosati (1989), (hereafter referred to as DR) which was developed for global data assimilation application. This scheme is an optimum interpolation scheme but unique to the extent that it performs a global minimization to update the entire model domain simultaneously, as opposed to other optimum interpolation schemes that update at a single point at a time. The scheme is quite efficient; but when combined with a primitive equation, numerical model of the Gulf of Mexico, the nowcasting step took too much computer time for it to be useful in real-time applications. After an initial implementation and test of the of this scheme, our obvious thrust was to enhance and make it more efficient.

The DR algorithm combines the model output with the observations using a linear least squares method. Because of the large dimensions involved, the minimization in the least squares procedure is performed by the method of steepest descent, as it avoids inversion of the covariance matrix, $\Sigma_{\mathbf{m}}$, of the model output. Instead, the procedure requires several multiplications of the matrix $\Sigma_{\mathbf{m}}$ with a vector, $\mathbf{g}$. Because the dimensions of $\Sigma_{\mathbf{m}}$ are large, and in-core storage parsimony is a necessity, it is not stored in-core. Since, repeated computation of $\Sigma_{\mathbf{m}}$ would require large computational resources, DR developed an efficient algorithm to evaluate $\Sigma_{\mathbf{m}}\mathbf{g}$. This algorithm avoided evaluation of $\Sigma_{\mathbf{m}}$ and approximated the multiplication $\Sigma_{\mathbf{m}}\mathbf{g}$ by repeated applications of a Laplacian operator.

The purpose of this technical report is to provide details of the DR algorithm, which has since found a much wider acceptance in the community. This algorithm provided insight into the mechanism of approximating $\Sigma_{\mathbf{m}}\mathbf{g}$ via spectral domain and led to the development of a new algorithm. The new algorithm (named as the Product-Polynomial Operator (PPO) algorithm), while maintaining the accuracy, approximates $\Sigma_{\mathbf{m}}\mathbf{g}$ far more efficiently than the Laplacian algorithm; the approximation is affected by applying a product of two operators that

are polynomials in simple averaging operators. Using computer simulations, approximation efficiencies of the two algorithms are compared with the actual matrix multiplication, $\Sigma_m \mathbf{g}$.

In addition to the basic motivation to develop a more efficient data assimilation algorithm with a Gaussian covariance function, it has also been of interest to develop an algorithm which accepts other spatial covariance functions of the model output errors. The PPO algorithm is shown to admit, with equal ease and efficiency, a wider class of covariance functions that can be expressed as a product of factors that are even functions in a single dimension. However, the practical utility of this result is yet to be explored; presently, not many two-dimensional covariance structures, other than Gaussian, appear to satisfy the factorability condition.

Although, the application of this work is intended for atmospheric and oceanic applications, its contents are more mathematically oriented. In Section 2, we first develop the statistical framework needed to arrive at a common formulation of the optimum interpolation problem, leading to the least squares solution by the steepest descent, iterative minimization process. In particular, it indicates the step where the matrix multiplication $\Sigma_m \mathbf{g}$ has to be evaluated repeatedly. Next, in Section 3, we describe the Laplacian smoothing algorithm used in DR to approximate this matrix multiplication. This is followed, in Section 4, by the development and description of our newly-developed PPO algorithm for the same approximation, with a discussion on the determination of the operator-polynomial degrees, approximation accuracy, and implementation procedures. Finally, in Section 5, we present the results of numerical simulations that compare the approximation accuracy and the computation efficiency of the two algorithms.

2. STATISTICAL FORMULATION OF THE DATA ASSIMILATION PROBLEM

The purpose of data assimilation is to estimate the true state of the ocean, Θ, by combining the model output, $\mathbf{T}_m$, and observations $\mathbf{T}_o$. The combined estimate (after dynamic initialization, in some cases; see Daley, 1981) is then used as the initial condition for the model integration. The vectors Θ and $\mathbf{T}_m$ are N-vectors defined on the model grid, $\mathcal{G}_m$; the observation vector, $\mathbf{T}_o$ is an M-vector defined on the observation grid $\mathcal{G}_o$. Usually, $M << N$, and $\mathbf{T}_o$ alone cannot provide an adequate representation of Θ. Thus, assimilation is performed, resulting in an estimate on $\mathcal{G}_m$ to provide initial conditions for the numerical integration of the model equations.

We assume there exists a mapping such that $\mathbf{D}(\mathcal{G}_m) = \mathcal{G}_o$. Then Θ_o, the true state of the ocean at the observation grid, can be written as $\Theta_o = \mathbf{D}(\Theta)$. Often $\mathbf{D}$ is assumed to be a linear mapping so that:

$$\Theta_o = \mathbf{D}\Theta. \tag{2.1}$$

The motivation to perform such a data assimilation is the assumption that both the model values and the observational data are unbiased estimators of the true state of the ocean, and that the optimal combination of the two will lead to an estimator of Θ, which has a smaller error variance than either of the two. Under the unbiasedness assumption, we can write the following linear model:

$$\begin{pmatrix} \mathbf{T}_m \\ \mathbf{T}_0 \end{pmatrix} = \begin{pmatrix} \Theta \\ \mathbf{D}\Theta \end{pmatrix} + \begin{pmatrix} \mathbf{e}_m \\ \mathbf{e}_0 \end{pmatrix} \tag{2.2}$$

where $\mathbf{e}_m$ and $\mathbf{e}_o$ are vectors of zero mean random errors in the model output and observations, with Σ_m and Σ_o as their respective covariance matrices. It is reasonable to assume that the errors in $\mathbf{T}_o$ and $\mathbf{T}_m$ are statistically independent. Then, the least squares solution to the true state Θ is obtained by minimizing the quadratic functional:

$$\mathcal{Q} = (\mathbf{T}_m - \Theta)'\Sigma_m^{-1}(\mathbf{T}_m - \Theta) + (\mathbf{T}_o - \mathbf{D}\Theta)'\Sigma_o^{-1}(\mathbf{T}_o - \mathbf{D}\Theta) \tag{2.3}$$

where the prime indicates matrix transposition. Lorenc (1986) arrived at the minimization of this quadratic functional from a Bayesian approach. It is customary to write $\mathcal{Q}$ in terms of corrections to the model values:

$$\mathcal{Q} = \mathbf{T}_c'\Sigma_m^{-1}\mathbf{T}_c + (\mathbf{D}\mathbf{T}_c - \mathbf{T}_{co})'\Sigma_o^{-1}(\mathbf{D}\mathbf{T}_c - \mathbf{T}_{co}), \tag{2.4}$$

where $\mathbf{T}_c = \mathbf{T}_m - \Theta$ and $\mathbf{T}_{co} = \mathbf{D}\mathbf{T}_m - \mathbf{T}_o$.

Minimization of $\mathcal{Q}$ is the most common formulation of data assimilation. Ordinarily, this minimization should be quite simple but for the fact that the dimensions of the model grid, and hence that of the covariance matrix, $\Sigma_\mathbf{m}$, are extremely large; thus, the routinely-used procedures are rendered inadequate, and efficient minimization algorithms must be devised.

The data assimilation algorithm is determined by specifying the covariance matrices $\Sigma_\mathbf{m}$ and Σ_o, and by the method of minimizing $\mathcal{Q}$. With a given $\Sigma_\mathbf{m}$, DR gave an efficient, preconditioned conjugate gradient algorithm for minimizing $\mathcal{Q}$, which avoids computing Σ_m^{-1} (also, see Navon and Legler, 1987). It iteratively reduces the gradient vector, $\mathbf{g}$, to zero; i.e.,

$$\Sigma_m^{-1}\mathbf{T}_c + \mathbf{D}'\Sigma_o^{-1}(\mathbf{D}(\mathbf{T}_c) - \mathbf{T}_{co}) \rightarrow 0. \tag{2.5}$$

Preconditioning is supplied by the matrix Σ_m. The iteration steps of the conjugate gradient method are as follows. Let $\mathbf{T}_j$ be the iterative solution to $\mathbf{T}_c$ at iteration j. We start with initialization:

$$\mathbf{T}_1 = \mathbf{0} \tag{2.6a}$$

$$\mathbf{g}_1 = -\mathbf{D}^\mathrm{T}\Sigma_o^{-1}\mathbf{T}_0 \tag{2.6b}$$

$$\mathbf{h}_1 = \Sigma_m \mathbf{g}_1 \tag{2.6c}$$

$$\mathbf{d}_0 = \mathbf{e}_0 = \mathbf{0}.$$

Then, the iterations are carried out via equations, (2.7a)-(2.7h), given below:

$$\mathbf{d}_n = \mathbf{h}_n + \beta_{n-1}\mathbf{d}_{n-1} \tag{2.7a}$$

$$\mathbf{e}_n = -\mathbf{g}_n + \beta_{n-1}\mathbf{e}_{n-1} \tag{2.7b}$$

$$\mathbf{f}_n = \epsilon_n + \mathbf{D}^T \Sigma_o^{-1} \mathbf{D}\mathbf{d}_n \tag{2.7c}$$

$$\alpha_n = \frac{\langle \mathbf{g}_n . \mathbf{h}_n \rangle}{\langle \mathbf{d}_n . \mathbf{f}_n \rangle} \tag{2.7d}$$

$$\mathbf{g}_{n+1} = \mathbf{g}_n + \alpha_n \mathbf{f}_n \tag{2.7e}$$

$$\mathbf{T}_{n+1} = \mathbf{T}_n + \alpha_n \mathbf{d}_n \tag{2.7f}$$

$$\mathbf{h}_{n+1} = \Sigma_m \mathbf{g}_{n+1} \tag{2.7g}$$

$$\beta_{n+1} = \frac{\langle \mathbf{g}_{n+1} . \mathbf{h}_{n+1} \rangle}{\langle \mathbf{g}_n . \mathbf{h}_n \rangle} \tag{2.7h}$$

where n is the iteration index. These steps are repeated until convergence is achieved. Because of statistically independent observation errors, their covariance matrix, Σ_o, and its inverse, Σ_o^{-1}, are diagonal. Thus, expressions in eqs. (2.6b) and (2.7c) involving these matrices are easily computed. Also, the bilinear interpolation matrix $\mathbf{D}$ is ingeniously manipulated. Thus, the only computational concerns arise from eqs. (2.6) and (2.7), where computation of $\mathbf{h}_{n+1} = \Sigma_m \mathbf{g}_{n+1}$ is carried out.

For most implemented covariance structures, this operation is quite expensive, moreso for repeated applications of the operation. In fact, there is a two-fold problem at this step: One is the multiplication operation that we just mentioned; the second is a practical problem of actual storage of the large-dimensioned computed covariance matrix. Although the computed matrix can be stored on a file and read when needed, the input/output operations can result in excessive demand on computer resources.

The DR algorithm alleviates the above difficulties when the covariance structure of Σ_m is based on a Gaussian function

$$\epsilon(r) = a \exp(-r^2/b^2). \tag{2.8}$$

$\epsilon(r)$ represents the covariance between two model grid values, r is the distance between the two grid points, and b is the correlation length scale. The parameter a represents the error variance of the model values. We have developed a new algorithm, based on polynomials in simple averaging operators, that is more efficient than the DR algorithm and accepts other covariance structures. The two algorithms are described below.

3. THE LAPLACIAN SMOOTHING ALGORITHM

A description of the DR algorithm is provided below for several reasons. From a historical perspective, there is interest as to where we are coming from, what are the salient features of the original algorithm, and what features are being modified or enhanced. Because of a wide acceptance of this algorithm, it is appropriate to provide a detailed description. This algorithm provided the motivation to look at the efficiency problem in the spectral domain. A heuristic formulation of this development is a necessary background for the PPO algorithm also.

The DR algorithm is based on three stipulations: (1) the observation errors are statistically independent; (2) the model errors are statistically independent of the observation errors, and (3) the model errors are assumed to have a Gaussian covariance structure. The first two components led to the minimization of the quadratic functional $\mathcal{Q}$ in (2.3). To avoid inversion of the large-dimensioned matrix $\mathbf{\Sigma_m}$ the iterative method of conjugate gradient is used, whose iterative steps are specified in eqs. (2.6) and (2.7).

Note that we still need to invert $\mathbf{\Sigma}_o$, the observation error covariance matrix; the problem of this inversion is easily resolved from the stipulation that the errors of observations are statistically independent. In addition, to efficiently perform the iterative steps in eqs. (2.6)-(2.7), one needs to define the bilinear interpolation matrix $\mathbf{D}$. DR formulated $\mathbf{D}$ in a simple but elegant way so that its transposition and multiplication in eq. (2.7c) are easily manipulated; they handle observations, one at a time, so that the matrix $\mathbf{D}$ does not have to be handled in its giant form.

Finally, they implemented a computationally efficient algorithm for performing the matrix multiplication $\mathbf{\Sigma_m g}$. This algorithm involved N applications of the operator $(1 + \bar{\nabla}^2/4\beta N)$ on the field, g, where $\bar{\nabla}^2$ is the Laplacian operator defined on the discrete model domain. The number N, referred to as the degree of the Laplacian operator, is determined in a heuristic manner. Described below is their algorithm, the method of determining the degree N, and the way they handle the boundary problems inherent in the application of the Laplacian operator.

3.1 Approximating $\mathbf{\Sigma_m g}$

The usual protocol is to estimate the covariance structure, $\mathbf{\Sigma_m}$, from long-term historical nowcasting experiments using real data. However, in the absence of such a historical record, one can use reasonable approximations to provide a substitute covariance function. The use of the Gaussian formulation is a well accepted alternative.

Gaussian covariance structures have been used quite extensively in various applications (Daley, 1991) and have been shown to be quite appropriate in regions where correlations are positive (Thompson, 1956; Thiebaux and Pedder, 1987). With respect to the Gulf of Mexico, Gaussian covariance structures have been used to derive realistic dynamic climatologies of the Gulf (Passi and Kantha, 1992). The algorithm described below to approximate the matrix multiplication, $\mathbf{\Sigma_m g}$, is predicated on the fact that $\mathbf{\Sigma_m}$ is based on a Gaussian covariance function.

The Gaussian covariance function, $\epsilon(r)$, from (2.8) is expressed in a two-dimensional plane as:

$$\epsilon(r) = \epsilon(x, y) = \alpha \exp[-\beta(x^2 + y^2)] \tag{3.1}$$

where the coefficients α and β describe the error and the scale functions being considered. Comparing with (2.8), $\alpha = a$ and $\beta = \frac{1}{b^2}$, where b is the correlation scale length.

For further development, it will be useful to write an expression for an element of the vector $\mathbf{h} = \mathbf{\Sigma_m g}$ that occurs in (2.6c) and (2.7g). Note that the vector $\mathbf{h}$ corresponds to the two-dimensional model grid. Thus, for a grid-point (i_0, j_0), we have:

$$h(i_0, j_0) = \sum_{i=1}^{I_m} \sum_{j=1}^{J_m} g(i, j) \alpha \epsilon^{-\beta\Delta[(i-i_0)^2 + (j-j_0)^2]}. \tag{3.2}$$

where Δ is the uniform grid length along the X and Y directions, and I_m, J_m are the maximum values of i, j. Thus, the array $\mathbf{h}$, resulting from the matrix multiplication, $\mathbf{\Sigma_m g}$, represents smoothing of the $\mathbf{g}$ array using the Gaussian function, $\epsilon(x, y)$. DR approximate such smoothing in the continuous domain by a Laplacian operator and carry over the results to the discrete case using some empirical adjustments. Their Laplacian approximation result can be derived as follows:

Proposition. Let $\tilde{g}(x, y)$ be a function having continuous second order derivatives $\partial^2\tilde{g}/\partial x^2$ and $\partial^2\tilde{g}/\partial y^2$, and let $\epsilon(x, y)$ be a Gaussian function as defined in (3.1), then for a large integer N,

$$\left(1 + \frac{\nabla^2}{N}\right)^N \tilde{g}(0, 0) \approx \frac{\beta}{\alpha\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{g}(x, y) \epsilon(x, y) dx dy.$$

A Heuristic Motivation: Here, we provide a heuristic motivation that guided the development of the DR algorithm. A rigorous and exact treatment of the required analysis is neither necessary nor the intent of this technical report. The main objective is to provide an insight into and direction toward the analysis of the PPO algorithm, which is both rigorous and exact.

Since $\tilde{g}(x, y)$ is continuous, it can be expressed in the spectral domain as:

$$\tilde{g}(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} a_{mn} \exp[-i(mx+ny)]. \tag{3.3}$$

The smoothed value, $\tilde{h}(x,y)$ at $x = y = 0$ of $\tilde{g}(x,y)$, using the Gaussian smoothing function (3.1), is given by:

$$\begin{aligned}\tilde{h}(0,0) &= \alpha \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp[-\beta(x^2+y^2)] \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}\exp[-i(mx+ny)]dxdy \\ &= \frac{\alpha\pi}{\beta}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}\exp\left[-\frac{n^2+m^2}{4\beta}\right].\end{aligned} \tag{3.4}$$

The objective now is to approximate $\tilde{h}(0,0)$ of eq. (3.4) by applying a Laplacian smoothing. From (3.4) one needs to define an operator based on Laplacian ∇^2 that will yield an amplitude proportional to $\exp\left[-\frac{n^2+m^2}{4\beta}\right]$ at wave numbers (m,n).

Note that

$$\begin{aligned}\nabla^2\tilde{g}(x,y) &= \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\tilde{g}(x,y) \\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}(-m^2-n^2)\exp[-i(mx+ny)]\end{aligned} \tag{3.5}$$

Since, for a given y, as $N \to \infty$, $\lim(1+y/N)^N = \exp(y)$, an examination of (3.5) suggests the application of the operator $\mathcal{L} = (1+\frac{\delta}{N}\nabla^2)$ to $\tilde{g}$; so that at $x = y = 0$, this application yields:

$$\mathcal{L}\tilde{g}(0,0) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}\left(1-\frac{\delta}{N}(m^2+n^2)\right). \tag{3.6}$$

Successive N applications of this operator on $\tilde{g}$ yield:

$$\mathcal{L}^N\tilde{g}(0,0) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a_{mn}\left(1-\frac{\delta}{N}(m^2+n^2)\right)^N. \tag{3.7}$$

For values of m and n which are small relative to N,

$$\left(1-\frac{\delta}{N}(m^2+n^2)\right)^N \simeq \exp[-\delta(m^2+n^2)]. \tag{3.8}$$

Now, we assume that the high frequency content of $\tilde{g}(x,y)$ is negligible, i.e., there exists an integer $N' << N$, so that $a_{mn} \simeq 0$ for $m,n \geq N'$. Thus, from (3.7), we can approximate

$$\begin{aligned}
\mathcal{L}^N \tilde{g}(0,0) &\simeq \sum_{m=-N'}^{N'} \sum_{n=-N'}^{N'} a_{mn} \left(1 - \frac{\delta}{N}(m^2+n^2)\right)^N \\
&\simeq \sum_{m=-N'}^{N'} \sum_{n=-N'}^{N'} a_{mn} \exp[-\delta(m^2+n^2)] \qquad \text{from (3.8)} \\
&\simeq \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} a_{mn} \exp[-\delta(m^2+n^2)] \qquad (3.9)
\end{aligned}$$

We compare (3.4) with (3.9), and set $\delta = \frac{1}{4\beta}$ to obtain:

$$\mathcal{L}^N \tilde{g}(0,0) = (\beta/\alpha\pi)\tilde{h}(0,0) = (\beta/\alpha\pi)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{g}(x,y)e(x,y)dxdy,$$

which demonstrates the proposition.

The results of the continuous case demonstrated above are easily generalized to the discrete model domain. Let $\tilde{g}(x,y)$ be the continuous analog of the discrete array $g(i,j)$, i.e., $\tilde{g}(i,j) = g(i,j)$. Thus, for $(i_0,j_0) = (0,0)$, we can approximate $h(i_0,j_0)$ of (3.2) as

$$\begin{aligned}
h(i_0,j_0) &\simeq \gamma \tilde{h}(0,0) \\
&\simeq \gamma \frac{\alpha}{\beta\pi} \mathcal{L}^N \tilde{g}(0,0) \\
&\simeq \gamma \frac{\alpha}{\beta\pi} \left(1 + \frac{\delta}{N}\bar{\nabla}^2\right)^N g(0,0) \\
&= \gamma \frac{\alpha}{\beta\pi} \bar{\mathcal{L}}^N g(0,0), \qquad (3.10)
\end{aligned}$$

where γ is an appropriate constant, $\bar{\nabla}^2$ approximates the Laplacian from $g(i,j)$ defined on the discrete grid, $\bar{\mathcal{L}}$ is the corresponding operator replacing $\mathcal{L}$, and N is then referred to as the degree of the Laplacian operator.

In spite of all these approximations made in the above derivation, the Laplacian smoothing algorithm works extremely well.

3.2 Determination of the Degree of the Laplacian Operator $\bar{\mathcal{L}}^N$

The degree N of the operator $\bar{\mathcal{L}}^N$ is determined heuristically. The criterion is to choose that value of N which stabilizes the computations and provides accurate computations, though

no absolute measures of the two criteria are easily applied. To achieve this, the following procedure is used:

(i) Select a value $\mathcal{N}$ for the degree of $\bar{\mathcal{L}}^{\mathcal{N}}$.

(ii) Let $i = I_0$ be the middle of the grid, and consider points (I_0, j).

(iii) Sequentially, consider a two dimensional field ψ, such that

$$\psi_J(i,j) = \begin{cases} 1 & \text{if } i = I_0, j = J \\ 0 & \text{otherwise,} \end{cases}$$

where $1 \leq J \leq J_{max}$. Note that only one field is non-zero. For each combination $(I_0, J), 1 \leq J \leq J_{max}$ smooth the field with Laplacian smoothing using the degree $\mathcal{N}$ selected in (i). Let $\tilde{\psi}(I_0, J)$ be the smoothed value at (I_0, J), so as to form an array $\tilde{\psi}_{\mathcal{N}}(I_0, J)$, $1 \leq J \leq J_{max}$. The smoothing is performed without topography.

(iv) Compare the smoothed array $\tilde{\psi}_{\mathcal{N}}$ with the smoothed array $\tilde{\psi}_{\mathcal{N}'}$ obtained using a smaller degree $\mathcal{N}'$. Stability is achieved when the array $\tilde{\psi}_{\mathcal{N}}$ varies smoothly over J, and accuracy is achieved when the two arrays do not differ 'significantly'. We stop if the two criteria of smoothness and accuracy are satisfied, otherwise $\mathcal{N}$ is incremented by a suitable integer and the above steps are repeated.

In addition, one must perform extended numerical experimentation to ensure that the selected value of $\mathcal{N}$ does not make the data assimilation process unstable. To avoid this, we add an increment of 5-10% of the selected number. For instance, in our experimentation with the Gulf of Mexico data assimilation work, we found that numerical stability was attained around $\mathcal{N} = 300$. To be safe, we added 20 to it and used $\mathcal{N} = 320$.

3.3 Treatment of the Boundary Values

First, we consider computation of the Laplacian $\bar{\nabla}^2\psi(i,j)$ at an internal point (i,j) of the field ψ. In this case, the point (i,j) is considered as a center of five point star in the ocean, and the Laplacian is approximated as:

$$\begin{aligned} \frac{\delta}{\mathcal{N}}\bar{\nabla}^2\psi &\simeq [w_N(\psi(i,j+1) - \psi(i,j)) - w_S(\psi(i,j) - \psi(i,j-1))] \\ &\quad + [w_E(\psi(i+1,j) - \psi(i,j)) - w_W(\psi(i,j) - \psi(i-1,j))] \\ &= -\psi(i,j)(w_N + w_S + w_E + w_W) \\ &\quad + w_N\psi(i,j+1) + w_S\psi(i,j-1) + w_E\psi(i+1,j) + w_W\psi(i-1,j) \quad (3.11) \end{aligned}$$

where w's are location dependent and distance related weights, given by

$$w_N = w_S = \frac{\delta}{N d_y^2(i,j)} \text{ and } w_E = w_W = \frac{\delta}{N d_x^2(i,j)} \tag{3.12}$$

where d_x and d_y are the horizontal grid spacings for the model. Conceptually, because of the latitude difference, it is possible that $w_N \neq w_S$, but $w_W = w_E$.

When the point is a boundary point, then the following approximation is implemented. Consider a one dimensional space where $\psi_0 = 0$ is the value assigned to the point next to the boundary and ψ_1 is the value at the boundary point. Then the smoothed value using the Gaussian function is given by

$$\psi_{0g} = \psi_1 \exp[-\beta r_0^2] \tag{3.13}$$

where the suffix g in ψ_{0g} indicates the Gaussian smoothing, and r_0 is the distance between the two points. This is the effect approximated after N applications of the Laplacian (corresponding to the discussion of Section 4. Thus we denote

$$\begin{aligned} \psi_{0N} \simeq \psi_{0g} &= \psi_1 \exp[-\beta r_0^2] \\ &= \psi_1 \exp\left[-\frac{N}{N}\beta r_0^2\right] \end{aligned} \tag{3.14}$$

For L Laplacian applications, where L is a large integer, we obtain from (3.8):

$$\sum_{m=0}^{N'}\sum_{n=0}^{N'} a_{mn}\left(1 - \frac{\delta}{N}(m^2+n^2)\right)^L \simeq \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} a_{mn}\exp[-\frac{L}{N}\delta(m^2+n^2)] \tag{3.15}$$

The right-hand side of (3.14) is the smoothing effect when β is replaced by $\frac{N}{L}$. Thus, for large integer L,

$$\psi_{0L} = \psi_1 \exp\left[-\frac{N}{L}\beta r_0^2\right] \tag{3.16}$$

However, the approximation is assumed for all $L = 1, \ldots, N$. Instead of different smoothing at each Laplacian application, an average smoothing factor

$$a_e = \frac{1}{N}\sum_{l=1}^{N-1}\frac{1}{l} \tag{3.17}$$

is applied. Thus, at each Laplacian step, (3.16) is replaced by

$$\psi_{0a} = \psi_1 \exp[-N a_e \beta r_0^2] \tag{3.18}$$

where the suffix a in ψ_{0a} indicates average smoothing application. Dropping the direction indices in (3.12), setting $r_0^2 = d^2$, and recalling that $\delta = \frac{1}{4\beta}$, we can write

$$N\beta r_0^2 = \frac{1}{4w}. \tag{3.19}$$

Substituting in (3.19), we obtain

$$\begin{aligned} \psi_{0a} &= \psi_1 \exp[-Na_e\beta r_0^2] \\ &= \psi_1 \frac{1}{\exp[\frac{a_e}{4w}]} \\ &\simeq \frac{\psi_1}{1 + \frac{a_e}{4w} + .5\left(\frac{a_e}{4w}\right)^2} \end{aligned} \tag{3.20}$$

Now, we can discuss the following boundary cases:

(i) The point (i,j) is the right most point, i.e., the point $(i+1,j)$ corresponds to land: in this case set $w = w_E$, $\psi_1 = \psi_{i,j}$ and $\psi_{0a} = \psi_{i+1,j}$.

(ii) The point (i,j) is the left most point, i.e., the point $(i-1,j)$ corresponds to land: in this case set $w = w_W$, $\psi_1 = \psi_{i,j}$ and $\psi_{0a} = \psi_{i-1,j}$.

(iii) The point (i,j) is the top most point, i.e., the point $(i,j+1)$ corresponds to land: in this case set $w = w_N$, $\psi_1 = \psi_{i,j}$ and $\psi_{0a} = \psi_{i,j+1}$.

(iv) The point (i,j) is the bottom point, i.e., the point $(i,j-1)$ corresponds to land: in this case set $w = w_S$, $\psi_1 = \psi_{i,j}$ and $\psi_{0a} = \psi_{i,j-1}$.

4. PRODUCT-POLYNOMIAL OPERATOR ALGORITHM

As we follow the derivation of the DR algorithm in Section 3, two possible improvements are suggested. The first is to replace the operator $\left(1+\frac{\nabla^2}{N}\right)^N$ in the Proposition by some other polynomial in ∇^2, $p(\nabla^2)$. The idea is to pick p so that $p(\nabla^2)$ is a better approximation to (3.4). If a more accurate approximation is achieved, then we believe it would be possible to pick the degree of p much lower than N, thus saving many computations when computing $p(\nabla^2)$. As we analyze this possibility, we notice the second improvement. Since the quantity we are trying to estimate (3.2) is a convolution product over a discrete space, it should be possible to give an analysis that exploits this fact directly without using continuous approximations as in Section 3. This proves to be true. We find that in this new domain analysis, the Laplacian operator is best replaced by a product of simple averaging operators. This leads to a good low-degree approximation to $h(m,n)$ of (3.2), and a simpler and more direct analysis. This, in turn, makes it much easier to specify the degree of p and handle the boundary conditions. For completeness we give a mathematical development of this analysis.

4.1 Preliminaries and Notation

In the subsequent development, we will express the matrix multiplication, $\Sigma_{\mathbf{m}}\mathbf{g}$, as a convolution, and exploit the fact that the Fourier transform of the convolution is a product of the transforms. Convolution between two, two-dimensional arrays, $g_1(s,t)$ and $g_2(m,n)$ is defined as:

$$(g_1 * g_2)(m,n) = \sum\sum g_1(s,t)g_2(m-s,n-t). \tag{4.1}$$

From Fourier theory, we know that

$$\widehat{g_1 * g_2} = \hat{g}_1\hat{g}_2. \tag{4.2}$$

Further, we shall define a function of several variables, $f(x_1,\ldots,x_n)$ as *separable* if it can be expressed as a product of factors that are even functions of a single variable, i.e., $f(x_1,\ldots,x_n) = \prod f_i(x_i)$ such that $f_i(-s) = f_i(s)$.

In our application

$$\epsilon(s,t) = \exp[(-s^2\delta_1^2 - t^2\delta_2^2)/b^2]. \tag{4.3}$$

Accordingly, this covariance function (4.3) is separable, since $\epsilon(s,t) = af_1(s)f_2(t)$ where $f_i(u) = \exp(-\delta_i u^2/b^2)$. We will require that the covariance functions be separable. A consequence of the functional separability of a covariance function is that its Fourier transform is also separable (due to factorability) and real (because of evenness).

4.2 Derivation of Canonical Forms

The development of the Product-Polynomial Operator is aided by deriving two canonical forms that are functionally similar. One canonical form expresses the two dimensional convolution, $h(m,n)$, in the Fourier domain, while the other represents the resultant of when $g(m,n)$ is operated on by the PPO. A comparison of the two expressions leads to the selection of the polynomials to specify the PPO.

We will consider convolution approximation in two dimensions with an 'elliptical' covariance function $\epsilon(s,t)$ given by (4.3); the Gaussian covariance function is a special case of (4.3) with $\delta_1 = \delta_2 = \delta$. The 'elliptical' covariance formulation provides the flexibility of different

model grid lengths along the two coordinates. The elements of the $\mathbf{h}$ vector corresponding to (3.2) are given by

$$h(m,n) = \sum\sum_{s,t} g(s,t)a \exp\left[-\frac{\delta_1^2(m-s)^2 + \delta_2^2(n-t)^2}{b^2}\right]. \tag{4.4}$$

We want to show that $h(m,n)$ in (4.4) can be approximated by successively applying, yet to be determined polynomial operators, $P_1(D_1)$ and $P_2(D_2)$ where $D_j, j = 1,2$ are simple averaging operators, given by

$$D_1 g(m,n) = [g(m+1,n) + g(m-1,n)]/2. \tag{4.5a}$$

$$D_2 g(m,n) = [g(m,n+1) + g(m,n-1)]/2. \tag{4.5b}$$

The implementation is affected by a comparison of the two canonical expressions given in the following theorems:

Theorem 1. Let $\hat{g}(\theta,\psi)$ be the two dimensional Fourier transform of $g(m,n)$. Then

$$h(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\psi) q_1(\theta) q_2(\psi) e^{2\pi i(m\theta + n\psi)} d\theta d\psi \tag{4.6}$$

where

$$q_k(\theta) = \sum_s \exp(-s^2\delta_k^2/b^2)\exp(-2\pi i s\theta),\ k = 1,2. \tag{4.7}$$

Theorem 2. Let $D_j, j = 1,2$ be the averaging operators from (4.5). Then for any polynomials P_1 and P_2.

$$P_2(D_2)P_1(D_1)g(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\psi) P_2(\cos 2\pi\psi) P_1(\cos 2\pi\theta) e^{2\pi i(m\theta + n\psi)} d\theta d\psi. \tag{4.8}$$

Note that the right-hand sides of (4.6) and (4.8) are similar functional forms. An examination of the two equations indicates that we could approximate $h(m,n)$ as

$$h(m,n) \simeq P_2(D_2)P_1(D_1)g(m,n) \tag{4.9}$$

provided we choose P_1 and P_2 such that $P_1(\cos 2\pi\theta) \simeq q_1(\theta)$, and $P_2(\cos 2\pi\psi) \simeq q_2(\psi)$. How good an approximation (4.7) is depends on how well we can approximate q_i by polynomials, P_i, over the domain of q_i.

We now prove the two results in (4.6) and (4.8). First, we note that $h(m,n)$ is a convolution of $g(m,n)$ with the two-dimensional array $\epsilon(k,l)$ defined in (4.3). Following the arguments of (3.2),

$$\begin{aligned} h(m,n) &= \sum\sum_{s,t} g(s,t)a \exp\left[-\frac{(x_m - x_s)^2 + (y_n - y_t)^2}{b^2}\right] \\ &= \sum\sum_{s,t} g(s,t)a \exp\left[-\frac{\delta_1^2(m-s)^2 + \delta_2^2(n-t)^2}{b^2}\right] \\ &= \sum\sum_{s,t} g(s,t)\epsilon(m-s,n-t) \\ &= g * \epsilon. \end{aligned} \tag{4.10}$$

The proof of Theorem 1 follows as an immediate consequence of the convolution (4.10).

Proof of Theorem 1. We start by computing the Fourier transform, $\hat{\epsilon}$:

$$\hat{\epsilon}(\theta,\psi) = \sum\sum \exp[(-s^2\delta_1^2 - t^2\delta_2^2)/b^2] \exp(-2\pi i s\theta - 2\pi i t\psi) \tag{4.11}$$

where θ and ψ are in the interval $[0,1]$. Because $\epsilon(s,t)$ is separable, we can write

$$\begin{aligned} \hat{\epsilon}(\theta,\psi) &= \left(\sum \exp(-s^2\delta_1^2)/b^2 \exp(-2\pi i s\theta)\right)\left(\sum \exp(-t^2\delta_2^2)/b^2 \exp(-2\pi i t\psi)\right) \\ &= q_1(\theta)q_2(\psi) \end{aligned} \tag{4.12}$$

where $q_k, k = 1,2$ are defined in (4.7). Next, since from (4.10), $h(m,n)$ is a convolution of g and ϵ, we can write $\hat{h}$ in the transform space as a product of their transforms:

$$\begin{aligned} \hat{h} &= \hat{g}\,\hat{\epsilon} \\ &= \hat{g}q_1q_2. \end{aligned} \tag{4.13}$$

Thus, by the Fourier Inversion Theorem

$$h(m,n) = \int_0^1\int_0^1 \hat{h}(\theta,\psi)\epsilon^{+2\pi i m\theta}\epsilon^{+2\pi i n\psi}d\theta d\psi.$$

Finally, Theorem 1 is proved when $\hat{h}$ in the above equation is replaced by the right-hand side of (4.13).

Proof of Theorem 2. With $\hat{g}$ as the Fourier transform of g, we can write

$$g(m,n) = \int_0^1\int_0^1 \hat{g}(\theta,\psi)\epsilon^{2\pi i(m\theta+n\psi)}d\theta d\psi. \tag{4.14}$$

Applying operator D_1 from (4.5a) to (4.14) we get

$$D_1 g(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\psi)\frac{(e^{2\pi i\theta} + e^{-2\pi i\theta})}{2} e^{2\pi i(m\theta+n\psi)} d\theta d\psi$$
$$= \int_0^1 \int_0^1 \hat{g}(\theta,\psi)\cos(2\pi\theta) e^{2\pi i(m\theta+n\psi)} d\theta d\psi.$$

We note that every time D_1 operates on g, we get an extra factor of $\cos(2\pi\theta)$ inside the integral. Thus, computing $D_1, D_1^2, \ldots$, it is easy to see that the application of a polynomial operator, $P_1(D_1)$, yields:

$$P_1(D_1)g(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\psi)P_1(\cos 2\pi\theta) e^{2\pi i(m\theta+n\psi)} d\theta d\psi.$$

A similar application of operator polynomial $P_2(D_2)$ yields:

$$P_2(D_2)g(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\psi)P_2(\cos 2\pi\psi) e^{2\pi i(m\theta+n\psi)} d\theta d\psi.$$

Thus, a successive application of two polynomial operators, P_1 and P_2, gives the desired expression (4.8), proving Theorem 2.

If we want to approximate $h(m,n)$ by $P_2(D_2)P_1(D_1)(g)(m,n)$, then the error will be bounded by

$$|(E - P_2(D_2^2)P_1(D_1))g(m,n)| \leq \int_0^1 \int_0^1 |\hat{g}(\theta,\psi)| d\theta d\psi \times \sup |q_1(\theta)q_2(\psi) - P_1(\cos 2\pi\theta)P_2(\cos 2\pi\psi)| \tag{4.15}$$

where q_i are given by (4.6). Thus, we see that our goal should be to estimate $q_i(\theta)$ by $P_i(\cos 2\pi\theta)$, $i = 1, 2$. This is discussed in Section 4.4 below.

4.3 Generalization to Separable Covariance Functions

This generalization is easily achieved by the following lemma and a restatement of Theorem 1. The results are stated in two dimensions, extension to higher dimensions is obvious.

Lemma 1: Let $\epsilon(x,y) = \epsilon_1(x)\epsilon_2(y)$ be a separable function, i.e., $\epsilon_i(s) = \epsilon_i(-s)$, then its Fourier transform is also separable such that

$$\widehat{\epsilon(\theta,\psi)} = \hat{\epsilon}_1(\theta)\hat{\epsilon}_2(\psi),$$

and $\hat{\epsilon}_i(\alpha) = \epsilon_i(-\alpha)$, $0 \leq \alpha \leq 2\pi$.

Theorem 3. Let $\hat{g}(\theta, \omega)$ *be the two dimensional Fourier transform of* $g(m,n)$. *Then* $h(m,n)$ *is given by (4.6); in this case*

$$q_k(\theta) = \sum_s \epsilon_k(s) \exp(-2\pi i s\theta),\ k = 1,2 \tag{4.16}$$

Lemma 1 is essentially proved in the beginning of the proof of Theorem 1 when $\exp[(-s^2\delta_1^2 - t^2\delta_2^2)/b^2]$ in (4.11) is replaced by $\epsilon(s,t)$. The separability of the Fourier transform is demonstrated in (4.12).

We again note that, although the applicability of the PPO algorithm to the wider class of separable covariance functions is quite elegant, its practical utility is yet to be explored. Presently, often employed covariance structures, other than Gaussian, do not appear to be amenable to such simple representation.

4.4 Polynomial-Operator Approximations

Using the evenness property of $\epsilon_k(s)$ in (4.16), we have

$$q_i(\theta) = \sum_{s \geq 1} \epsilon_i(s) 2\cos(2\pi s\theta) + \epsilon_i(0).$$

In our application, $\epsilon_i(s) = \exp(-s^2\delta_i^2/b^2)$, which is an even function in s. Now, consider picking a polynomial P_1 so that $P_1(\cos 2\pi\theta)$ is a good approximation to $q_1(\theta)$. To more easily compute this polynomial, we make the change of variables $t = \cos 2\pi\theta$, such that t ranges over [-1,1]. Thus, we need to compute $T_k = \cos(2\pi k\theta)$ in terms of t. This is easily achieved using the Chebychev polynomial recursion formula (Conte and deBoor, pg. 239, 1980). This result implies there exists a kth degree polynomial T_k such that $T_k(t) = T_k(\cos 2\pi\theta) = \cos(2\pi k\theta)$, and

$$T_{k+1} = 2tT_k(t) - T_{k-1}(t),\ k = 1,2,\ldots$$

where $T_0(t) = 1$, and $T_1(t) = t$. This is easily seen by induction using trigonometric identities:

$$\begin{aligned} T_{k+1}(t) = \cos(2\pi(k+1)\theta) &= \cos 2\pi\theta \cos 2k\pi\theta - \sin 2\pi\theta \sin 2k\pi\theta \\ &= 2\cos 2\pi\theta \cos 2k\pi\theta - (\cos 2\pi\theta \cos 2\pi k\theta + \sin 2\pi\theta \sin 2k\pi\theta) \\ &= 2\cos 2\pi\theta \cos 2k\pi\theta - \cos(2\pi(k-1)\theta) \\ &= 2tT_k(t) - T_{k-1}(t) \end{aligned}$$

We can compute $q_1(\theta)$ for any value of $t = \cos(2\pi\theta)$ using the recursion to compute $\cos(2\pi s\theta)$ in terms of t and adding up enough terms so that the remainder of the terms will be small. This is possible since $\sum_{s\geq 0} |\epsilon_1(s)| < \infty$.

Now we want to construct P_1 as an approximation to $q_1(\theta)$ considered as a function of t. The idea here is to compute P_1 as an interpolating polynomial on[-1,1] in t. The question is at which points do we interpolate? We use the expanded Chebychev points (Conte and deBoor, pg. 244, 1980). These points are:

$$x_i = \left[a + b + (a-b)\cos\left(\frac{2i+1}{2n+2}\pi\right) \Big/ \cos\left(\frac{\pi}{2n+2}\right)\right] \Big/ 2, \quad i = 0, \ldots, n$$

where $a = -1$, $b = 1$. This choice of points gives nearly an optimal polynomial P_1 such that $\max_{-1\leq t\leq 1} |P_1(t) - q_1(t)|$ is nearly minimized (Conte and deBoor, pg. 243, 1980). At worst, the error is only about four times as great using these points as it would be if we actually found the optimal polynomial. There are algorithms for computing these optimal polynomials. See the second Algorithm of Remes, and the Differential Correction Algorithm (Ralston and Rabinowitz, pgs. 315-320, 1978). We use the expanded Chebychev points instead because of their simplicity and because we have not decided what degree polynomial to use. It will be a lesser computational effort to determine the degree using the expanded Chebychev points than if we were to use these more complicated algorithms to determine the actual best approximation.

Thus, we let $P_1(t)$ be the interpolating polynomial to $q_1(t) = q_1(\theta)$, at the expanded Chebychev points. The function $q_1(\theta) = q_1(t)$ is computed to ϵ-tolerance by using the Chebychev recursion formula, and

$$q_1(\theta) \approx \tilde{q}_1(\theta) = \epsilon_1(0) + \sum_{s=1}^{s_0} \epsilon_1(s) 2\cos(2\pi s\theta), \tag{4.17}$$

where we pick s_0 such that $2\sum_{s=s_0+1}^{\infty} |\epsilon_1(s)| \leq Tol_1$, where Tol_1 will be determined below.

4.5 Determination of the Degree of P_1

To determine the degree of P_1 requires some computational effort. We first determine an error tolerance Tol_2 (given below). We pick n and find the polynomial of degree n that interpolates $\tilde{q}_1(\theta)$ at the expanded Chebychev points, $x_i, i = 0, \ldots, n$. We approximate $||\tilde{q}_1 - P_1||$ by $\max_{1\leq i\leq 100} ||\tilde{q}_1 - P_1||$ where t_i are one hundred equally-spaced points on [-1,1]. If $||\tilde{q}_1 - P_1|| \leq Tol_2$, we stop. Otherwise, we increase the degree. We stop if the tolerance is satisfied or if the degree exceeds some predetermined degree. In our case, we chose this degree

to about 10. If the degree becomes too large, the computations become numerically unstable. One should also stop if $||\tilde{q}_1 - P_1||$ starts to increase. If the error goal, $||\tilde{q}_1 - P_1|| \leq Tol_2$, is not satisfied, then an error message is given, and the degree that minimized $||\tilde{q}_1 - P_1||$ is used. Assuming the error goal is achieved, we have $||q_1 - P_1|| \leq ||q_1 - \tilde{q}_1|| + ||\tilde{q}_1 - P_1|| \leq Tol_1 + Tol_2 = Tol_3$. A similar analysis holds for q_2 and P_2 with $||q_2 - P_2|| \leq Tol_3$.

4.6 Determination of Error Tolerances

To determine the error tolerances above, we need to keep in mind the goal of making

$$|h(m,n) - P_2(D_2)P_1(D_1))g(m,n)| \leq \epsilon, \forall (m,n) \in \mathcal{Z}^2$$

where ϵ is given. Note ϵ should not be chosen much smaller than the expected error in one's measurements. To determine Tol_1, Tol_2, we need an error estimate.

$$\begin{aligned} |h(m,n) - P_2(D_2)P_1(D_1))g(m,n)| &\leq \int_0^1 \int_0^1 (|\hat{g}(\theta,\omega)|d\theta d\omega)||q_1 q_2 - P_1 P_2|| \\ &\leq (M_1 Tol_3 + M_2 Tol_3)\mathcal{I} \end{aligned}$$

where $\mathcal{I} = \int_0^1 \int_0^1 |\hat{g}(\theta,\omega)|d\theta d\omega$, and $M_i = ||q_i||$ (sup-norm). We have also approximated $||P_2|| \approx M_2$. We can estimate $||q_i|| \leq |\epsilon_i(0)| + \sum_{s\geq 1} 2|\epsilon_i(s)|$, $i = 1,2$. Earlier, we were able to show that $||\hat{q}||_1 \leq ||g||_1$. Now, if $||g||_1$ is large, the above error estimate is too conservative to be of any use. Instead of an arbitrary g, we choose the special case of $g = \delta_{s,t}$, which is one at (s,t) and zero at all other points. Then $||g||_1 = 1$, and we need to make $Tol_3 \leq \frac{\epsilon}{M_1+M_2}$, and pick $Tol_1 = Tol_2 = \frac{Tol_3}{2}$. This is justified since

$$g(m,n) = \sum_s \sum_t g(s,t)\delta_{s,t}(m,n), \text{ i.e., } g = \sum_s \sum_t g(s,t)\delta_{s,t},$$

and $h(m,n) = \sum_s \sum_t g(s,t)E(\delta_{s,t})$. So, if we approximate $E(\delta_{s,t})$, we have a good approximation to $h(m,n)$. We have found that the above strategy works well. Other error estimates are too pessimistic and, hence, require much more computational effort.

4.7 What to Do at the Boundary

In practice, data are missing off some finite set. We set this missing data to zero. The algorithm is done in one dimension at a time. Each row can be done independently. This means that many of these computations can be done in parallel. The nonzero values in the data propagate into the missing data, one unit on each end of a fixed data line, each time

we apply D_i. Thus, we need to add temporary memory of size $2n_i$, where n_i = degree of polynomial P_i, $i = 1, 2$. We only need to start the computations at a point once there is a nonzero value on either side of the point.

The calculations for the PPO algorithm are local in the sense that each time we apply the averaging operator, we just use information directly to the left and right of the point where the computation is being performed. So for each power of the operator, we reach one unit further to the left and right. So it is only after n_d steps that we notice the effect of data n_d units away. The convolution operator itself averages over all the data and so it is global in extent. However, if we did a truncated sum for the convolution, i.e., stop adding after the Gaussian terms are small, then this operator would also be local in nature. The DR algorithm is also local in the same way, but it uses information in the five point stencil about the point (See Section 3.3), each time spreading outward one unit simultaneously in X and Y directions. After N applications of $\tilde{\mathcal{L}}$, the DR has absorbed information from and $N \times N$ square around the point.

The issue at the boundary is that both algorithms assume the missing data is zero. One could extrapolate or interpolate or extend the data in a periodic fashion, but in all cases we are filling in missing data (with PPO the data is set to zero). Then we apply the algorithm. However, we don't compute the operator very far outside our data set since we don't care about these values except to make our algorithm function. So, we only compute our n_d units off each row. Calculations at the boundary are suspect for both algorithms, the only difference being that with the PPO algorithm we treat the missing data as zero and apply the algorithm, whereas the DR algorithm tries to approximate the effect of applying operator in (3.20). This is because the degree of the Laplacian operator is so high that computing off the data set would introduce serious errors. In the PPO algorithm the degree is low, so it is no problem.

4.8 Computational Effort

When we interpolate q_i at the expanded Chebychev points $x_0, \ldots, x_n$, we obtain a polynomial P_i in Newton form (Conte and deBoor, pgs. 40-44, 1980),

$$P_i(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)\ldots(x - x_{n-1}).$$

Replacing x by D_i, we need to compute

$$P_i(D_i) = a_0 + a_1(D_i - x_0) + a_2(D_i - x_0)(D_i - x_1) + \ldots + a_n(D_i - x_0)\ldots(D_i - x_{n-1}).$$

We evaluate this in Newton nested form, just as we do for a polynomial. Thus, a typical step in this evaluation is $k_1 I + k_2(D - k_3 I)$ where I is the identity operator. So, one application of such an operator at a point in the grid requires three multiplications and two additions. To evaluate D requires one more addition and a division by 2 (a shift register, so we will not count this operation). Thus, one step in the nested evaluation is 6 arithmetic operations - 3 multiplications and 3 additions. Then we do this at each point of the grid. We do this operation n_i times (degree of P_i). Let N be the total number of points on the grid. We can either add $2n_i$ points to the data points or ignore the calculations at the boundary, if the degree is small compared to the length of the line. So, we have $6n_i N$ operations so far. After we have computed $P_1(D_1)$, we compute $P_2(D_2)$. So, we have a total of $6N(n_1 + n_2)$ arithmetic operations (O(N)). Thus, if the degrees of n_1 and n_2 are fairly small, we have an efficient algorithm. If n_1 and n_2 are large, then there are too many computations, and the numerical stability is a problem. The key question is how large are these degrees? It is, thus, important not to make unreasonable error requirements.

Additional Remarks: This technique fits into the general philosophy found in Parks and Burns (1987), and references therein. One estimates, in an optimal way, the response function q_i by selecting coefficients of a filter to best estimate q_i in the infinity (supnorm) norm. Our algorithm exploits this idea by using a polynomial in the simple averaging operators (filters).

5. EFFICIENCY COMPARISON OF THE THREE ALGORITHMS

In order to give some measure of accuracy we implemented two pieces of code: the first calculates $h(m,n)$ directly from equation (3.2) and the second approximates $h(m,n)$ with the PPO algorithm.

To be more explicit, the first routine, which we shall call Matrix-Mult, calculated

$$h(m,n) \approx \sum_s \sum_t g(s,t) \exp\left[(-(m-s)^2\delta_1^2 - (n-t)^2\delta_2^2)/b^2\right] \qquad (5.1)$$

where the sums are only taken over "close" pairs (s,t), close enough that the exponential term is still significant. The numbers δ_i in (5.1) above may be interpreted in two different manners: with equal grid lengths in the two orthogonal directions, they represent different correlation length scales, b/δ_i; or, with the same correlation length scale, b, the two numbers represent different grid lengths, δ_i. Obviously, the combination of the two situations is controlled by specifying δ_i/b.

In the first test case, the function $g(s,t)$ was a delta function, i.e., it is zero everywhere except at the center of the grid. The grid boundary was rectangular (Figs. 1-3). In the

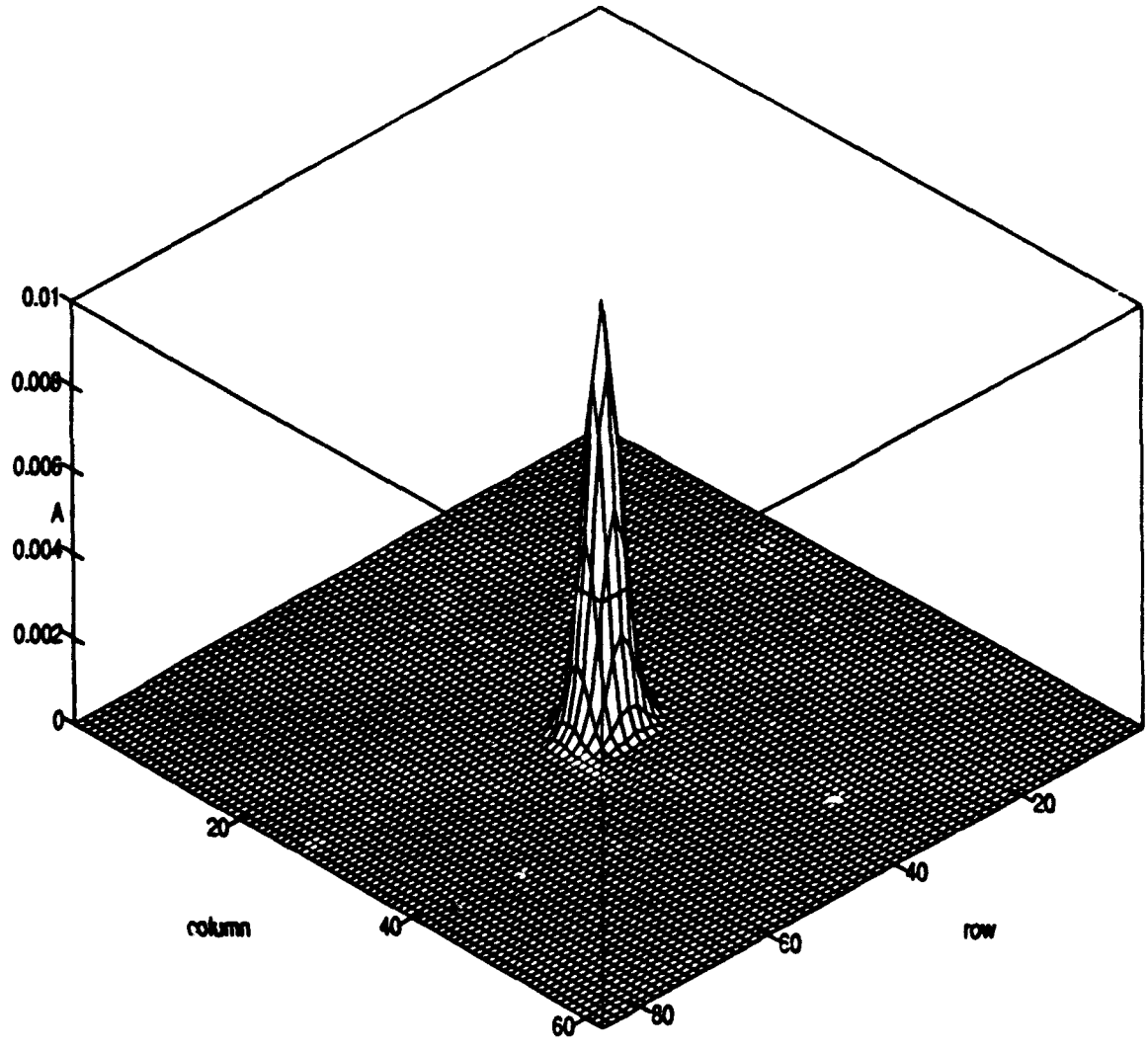

Figure 1. Test Case – Gaussian smoothing of delta function defined on a rectangular domain: Matrix-Mult Algorithm. Results are scaled to 0.01.

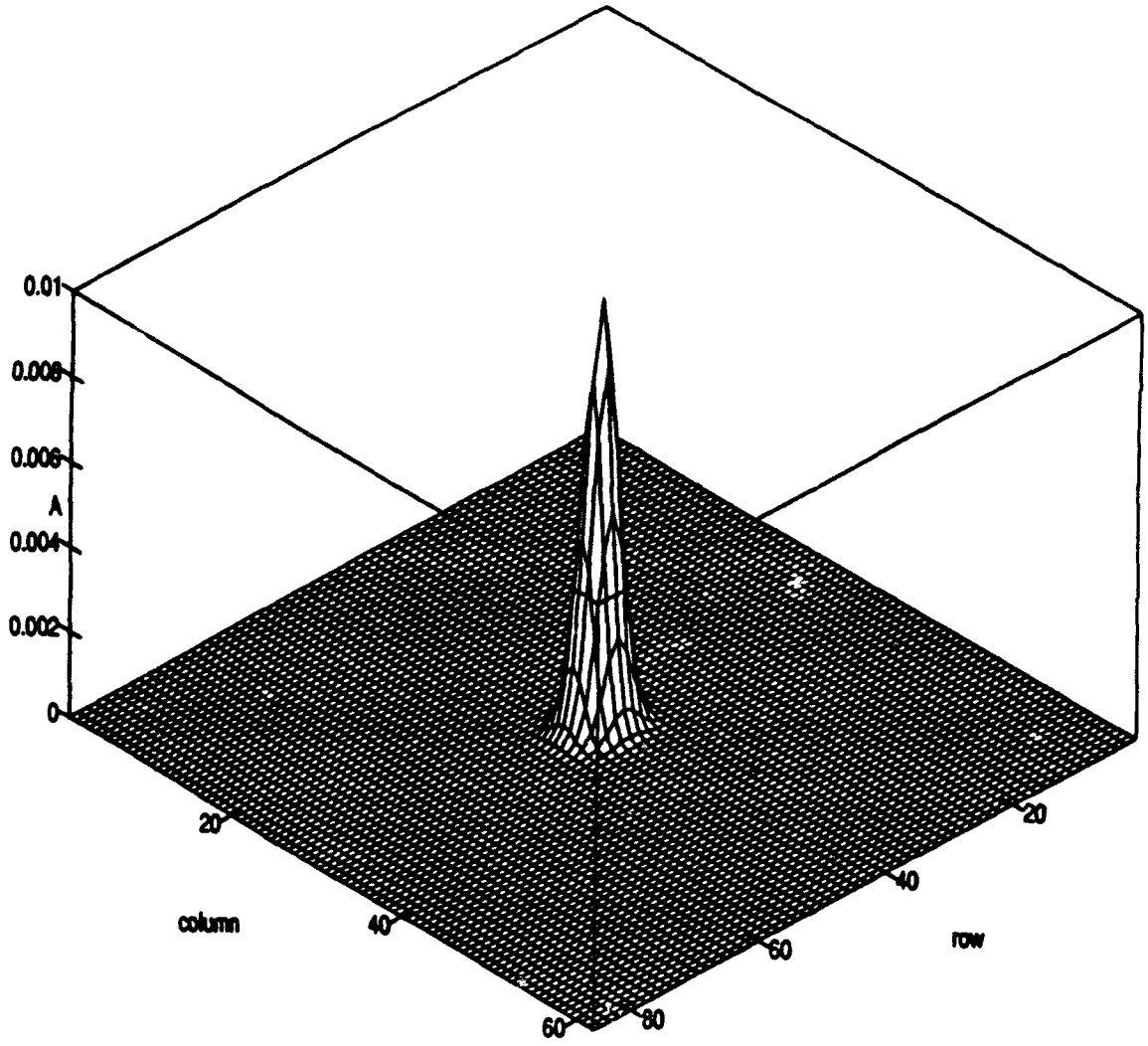

Figure 2. Test Case – Gaussian smoothing of delta function defined on a rectangular domain: Product Polynomial Algorithm. Results are scaled to 0.01.

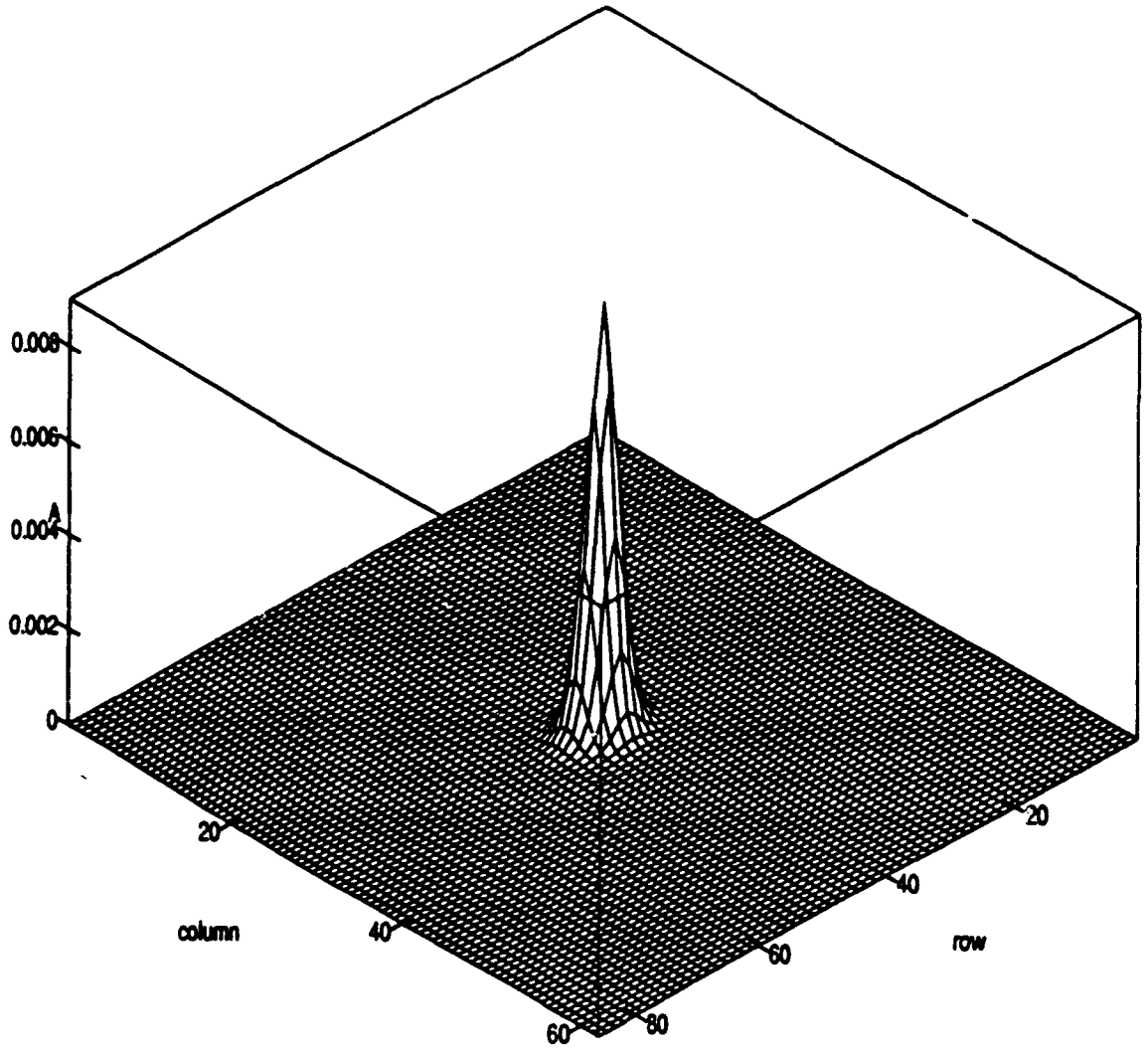

Figure 3. Test Case – Gaussian smoothing of constant function defined on rectangular domain: Derber-Rosati Algorithm. Results are scaled to 0.01.

second test case, we specified $g(s,t) = 1$ for all grid points, (s,t), within irregular boundaries depicting the Gulf of Mexico topography (Figs. 4-6); this grid was employed in data assimilation simulations using a numerical ocean model, which is the third test case. Thus, the first two cases provide the performance of the three algorithms in computing the matrix multiplication, while the third test case provides the comparison in actual data assimilation situation. For all simulations, the scale factor b and the grid dimensions δ_i in the simulations were taken so that $b = 50$ and $\delta_1/b = 0.4052$ and $\delta_2/b = 0.4448$.

While implementing the PPO algorithm, the number of terms, s_0, retained in $\tilde{q}_1$ and $\tilde{q}_2$ in (4.17) are calculated as functions of the grid spacing, a preset tolerance ($Tol = 0.001$) and the scale parameter b as specified for Matrix-Mult. Explicitly, we set the s_0 to be

$$s_0 = trunc[\sqrt{-\ln(Tol)}/(avg(dx_i)/b)] + 1 \tag{5.2}$$

where dx_i is the grid spacing in the direction x_i and $avg(dx_i)$ is the average grid spacing in that direction and 'trunc' is the truncation operator. For both $\tilde{q}_i$, (5.2) gave $s_0 = 7$. This led to the Chebychev polynomials P_i of degree 8, that uniformly approximated the $\tilde{q}_i$'s with error approximately 0.000186.

The coefficients of the polynomials P_i and the interpolation points remain fixed throughout any single run. To save the time recomputing we used a FORTRAN SAVE statement to save the interpolation points and coefficients, so they need only be calculated once. We set up temporary arrays to hold the enlarged grids to implement the boundary scheme. Off the data set the arrays are set to zero, and we use the off-grid points only as they become nonzero.

Then we compute the application of each operator $P_i(D_i)$ on the grid in order. That is, we first calculate $P_1(D_1)(g)$, then $P_2(D_2)[P_1(D_1)(g)]$. We use the Newton form of the polynomial with the precomputed and SAVE-ed interpolation points and coefficients.

All code was written in portable FORTRAN and tested on the CRAY YMP. The simulation results were scaled by 0.01, and are given in Figs. 1-3. Figure 3 provides the results from the original DR algorithm (with $n = 320$ repeated applications of the Laplacian) to compare with Matrix-Mult and the PPO algorithms in Figs 1 and 2 respectively. No significant performance difference is discernible in this case.

Perhaps more illuminating is the comparison provided by the second set of simulations. In this case, approximation due to the PP algorithm shows a better fidelity to the actual computation than the DR algorithm; we notice some performance deterioration of the DR scheme along the boundary. See Figs. 4, 5, and 6. The simulation results were scaled by 0.1.

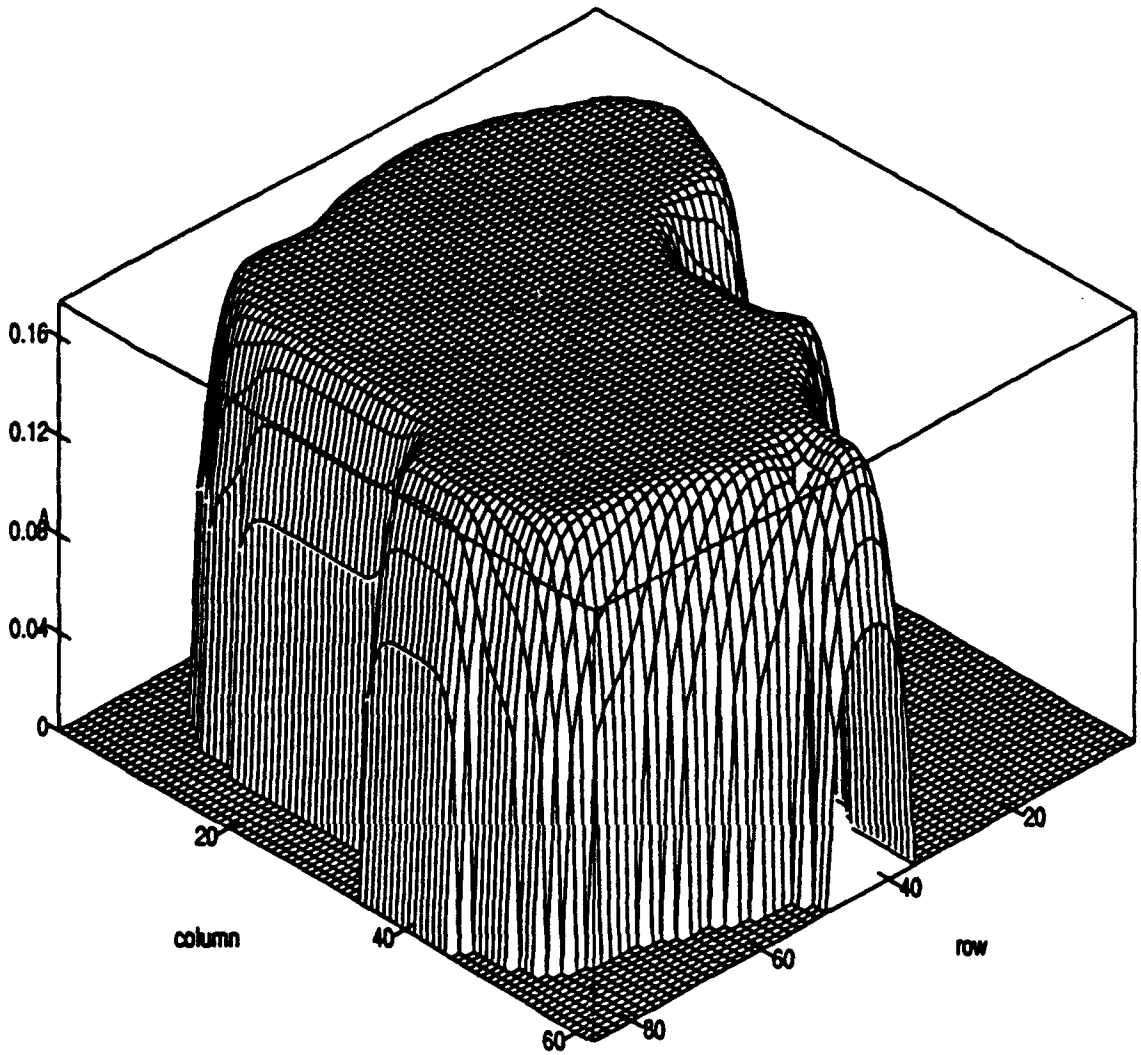

Figure 4. Test Case – Gaussian smoothing of constant function defined on an irregular domain provided by the Gulf of Mexico topography: Matrix-Mult Algorithm. Results are scaled to 0.1.

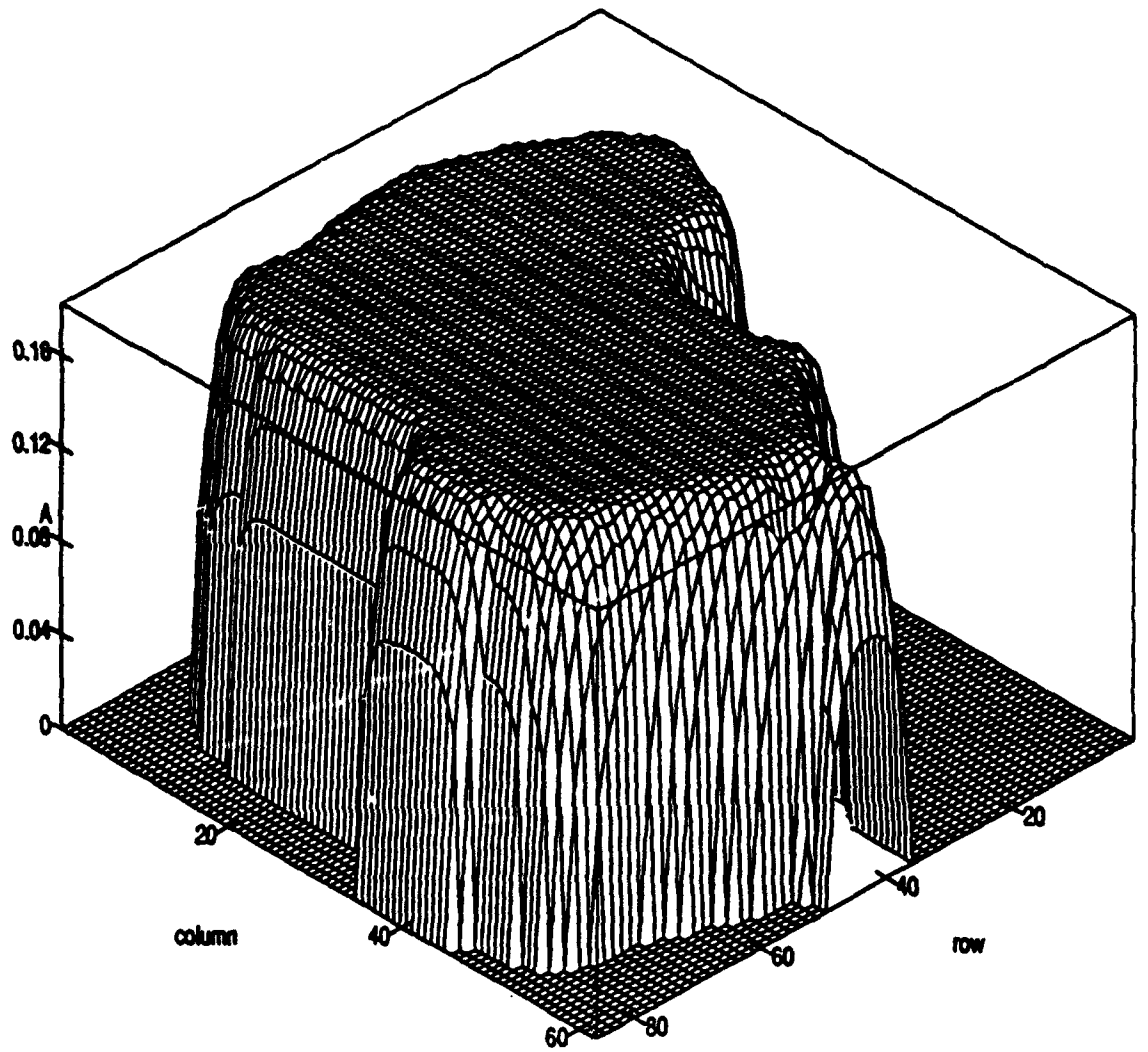

Figure 5. Test Case – Gaussian smoothing of constant function defined on an irregular domain provided by the Gulf of Mexico topography: Product Polynomial Algorithm. Results are scaled to 0.1.

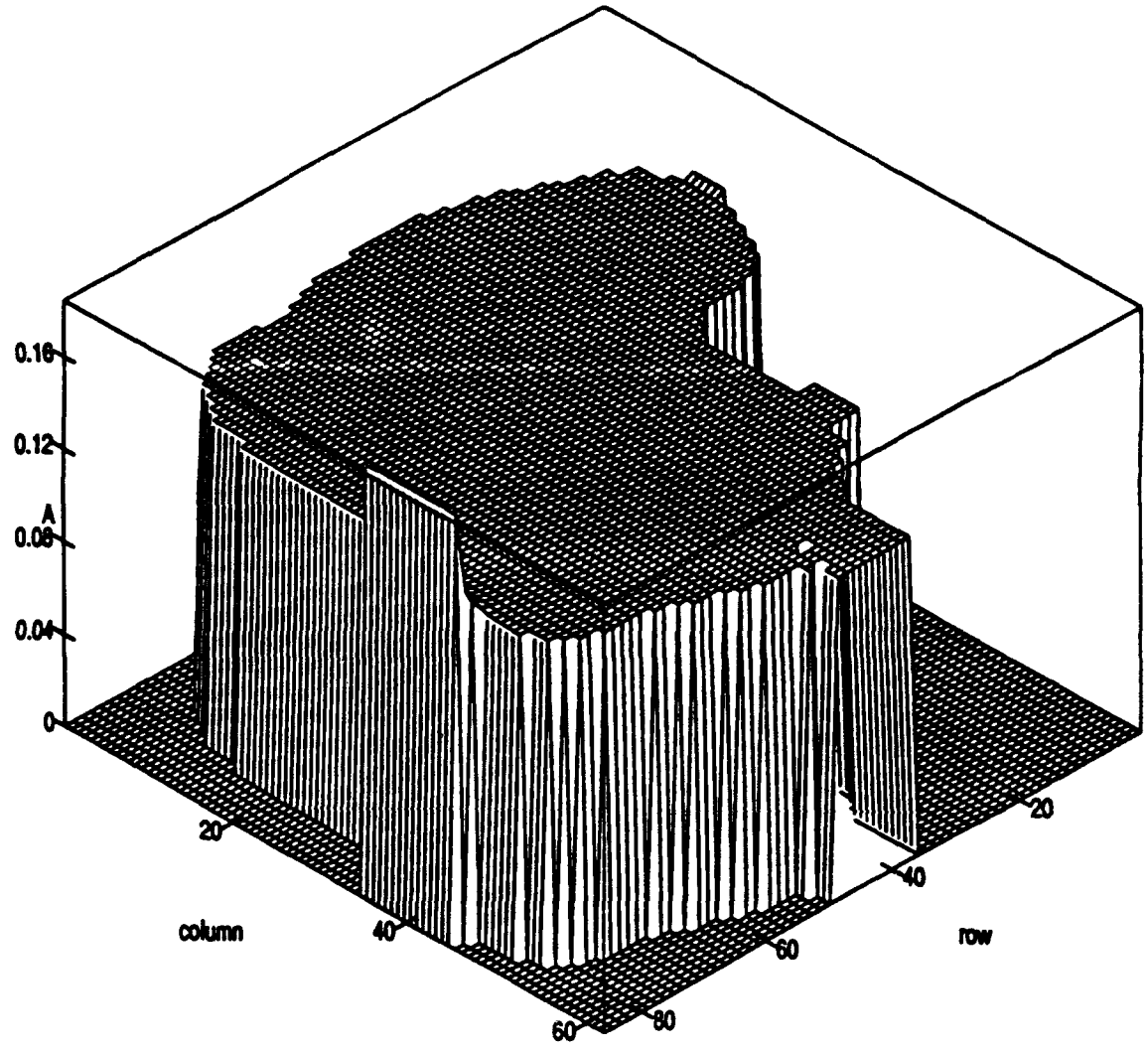

Figure 6. Test Case – Gaussian smoothing of constant function defined on an irregular domain provided by the Gulf of Mexico topography: Derber-Rosati Algorithm. Results are scaled to 0.1.

In addition to giving very accurate results, the PPO algorithm is very fast. On the CRAY YMP, for a single matrix multiplication with "real" data, the PPO algorithm takes approximately 0.09 seconds, the DR algorithm 3.1 seconds, and the MATRIX-Mult takes 15.4 seconds.

In a full data assimilation scenario, the computations were performed on CRAY YMP only. A primitive equation, numerical model of the Gulf of Mexico, integrated for forty days without any assimilation, followed by a ten-day integration with assimilation of the vertical temperature profile data. The DR scheme used $\mathcal{N} = 320$ repeated applications of Laplacians, as shown in eq. (3.10). The PPO algorithm used a sixth degree polynomial operators, $P(D_1$ and $P(D_2)$. In addition to assimilation by the two algorithms, the data were assimilated using actual matrix multiplication in (10g), so that the results of the two algorithms could be compared against a reference.

The results are shown Figs. 7a, b, and c for the computed sea-surface height. The three plots show remarkable similarity. The sea-surface height map derived from the PPO algorithm (Fig. 7b) matches exactly with the map obtained from the direct matrix multiplication (Fig. 7c), indicating differences of the order of 10^{-4} m. The sea surface height map from the DR algorithm (Fig. 7a) differs in the third decimal place. However, we see quite a significant difference from the sea surface temperature maps (Figs. 8a,b,c). The PPO algorithm (Fig. 8b) once again matches exactly, up to the accuracy of the map, with the reference case (Fig. 8c). But a noticeable deviation between the maps from the DR algorithm (Fig. 8a) and the reference is evident. At around (88W, 22N), the DR algorithm produces a high of 31.3°C as compared to a high of 30.3°C in the reference map.

For a comparison on computation times in the full data assimilation experiments, the DR scheme took 6.87 s per day of assimilation using a single CRAY YMP processor versus .68 s for the PPO algorithm. The matrix multiplication algorithm took an inordinately large time of 25 CPU s.

6. CONCLUDING REMARKS

We have formulated the problem of data assimilation in terms of a statistical linear model, where the true state of the ocean/atmosphere is estimated by minimizing a quadratic functional. Matrix multiplication, $\Sigma_{\mathbf{m}}\mathbf{g}$, is identified to be the major computational bottleneck, where E is a covariance matrix based on some covariance formulation and g is a multi-dimensional vector defined on a specified uniform rectangular grid.

With this background, we provided the details of the DR algorithm. This algorithm has found a wide acceptance in the user community as it incorporates several useful features and

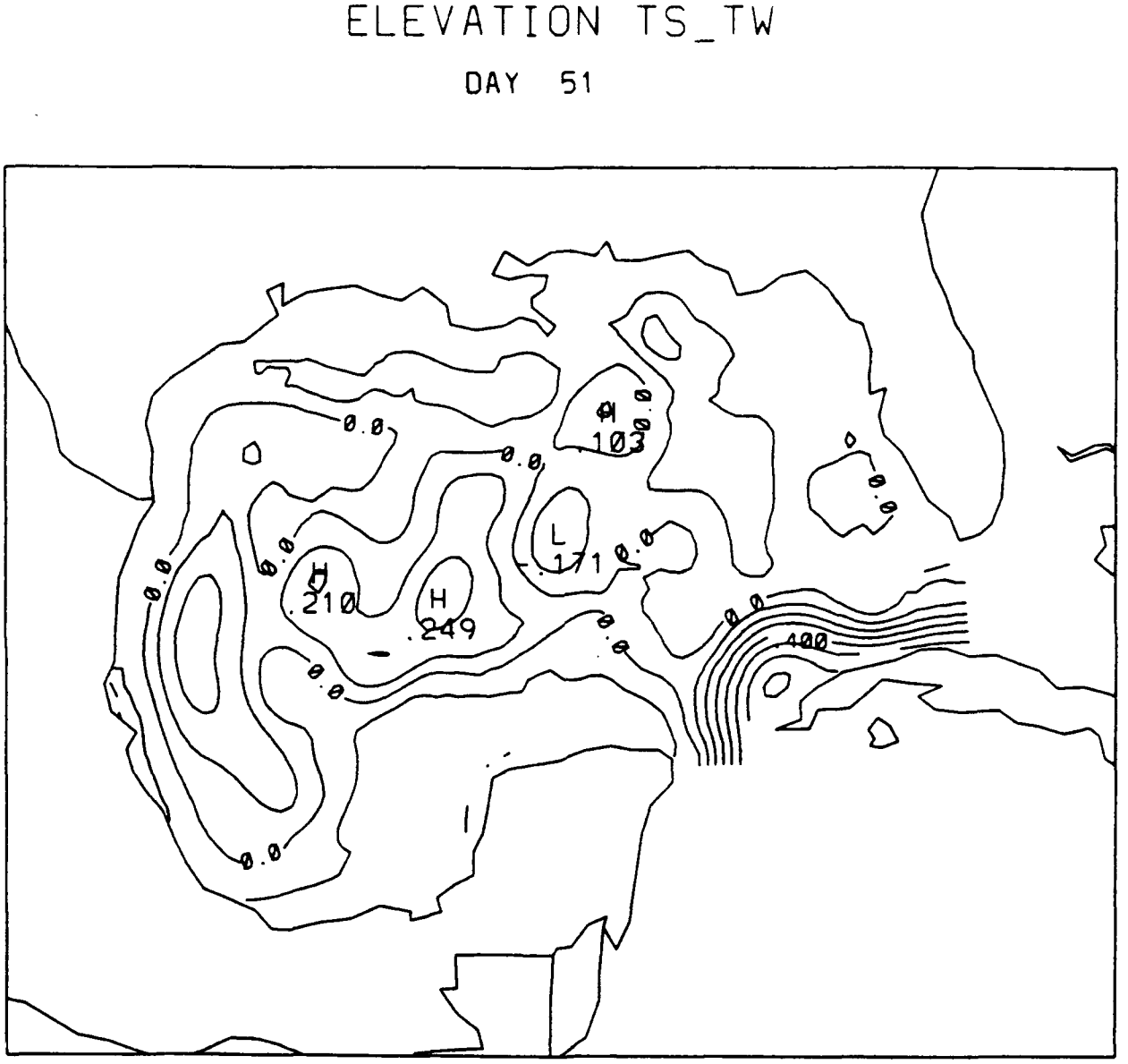

Figure 7a. Test Case – Data Assimilation simulations using vertical temperature profiles: Simulated Sea Surface Height Using Derber-Rosati algorithm.

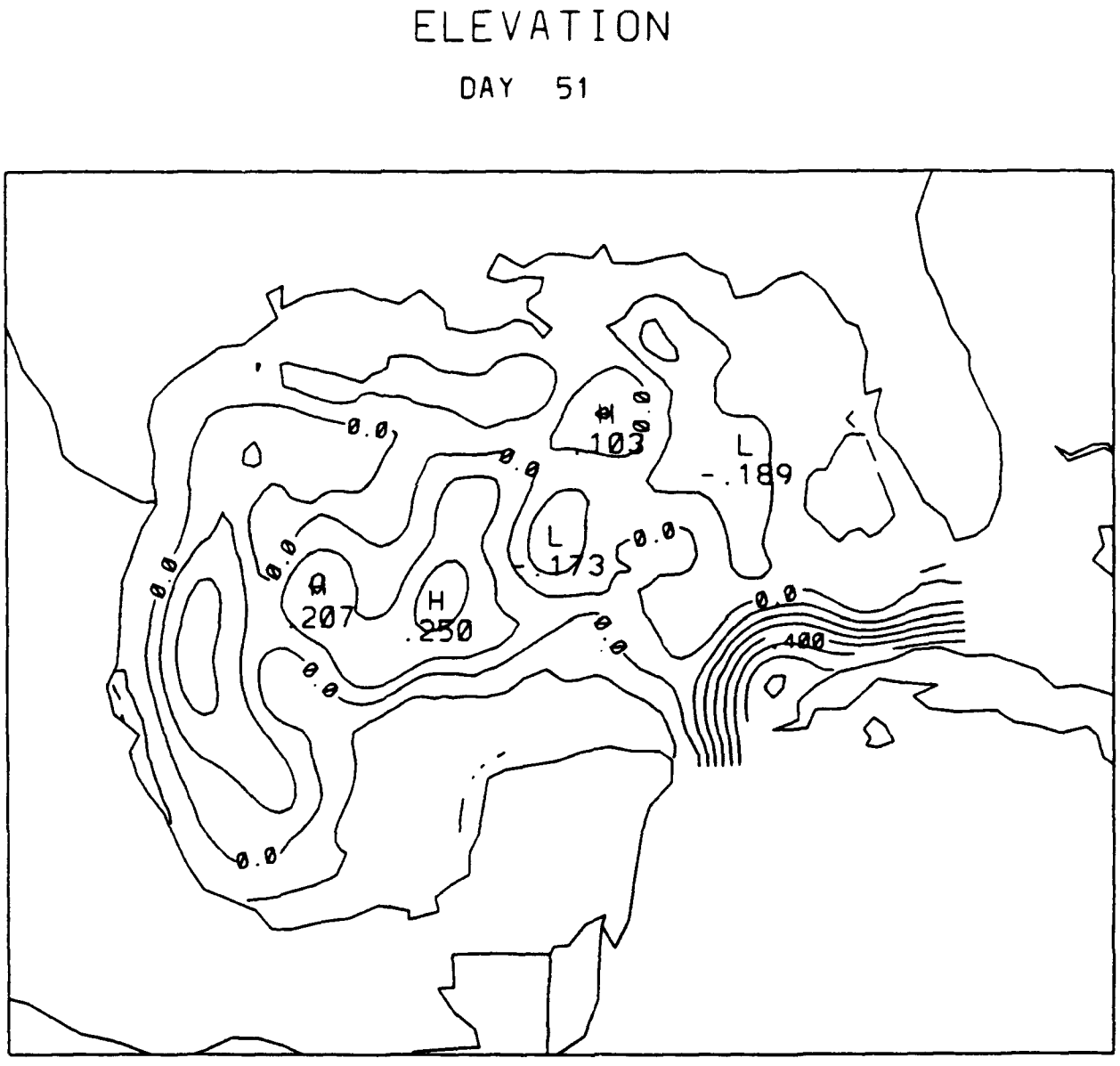

Figure 7b. Test Case – Data Assimilation simulations using vertical temperature profiles: Simulated Sea Surface Height Using Product-Polynomial algorithm.

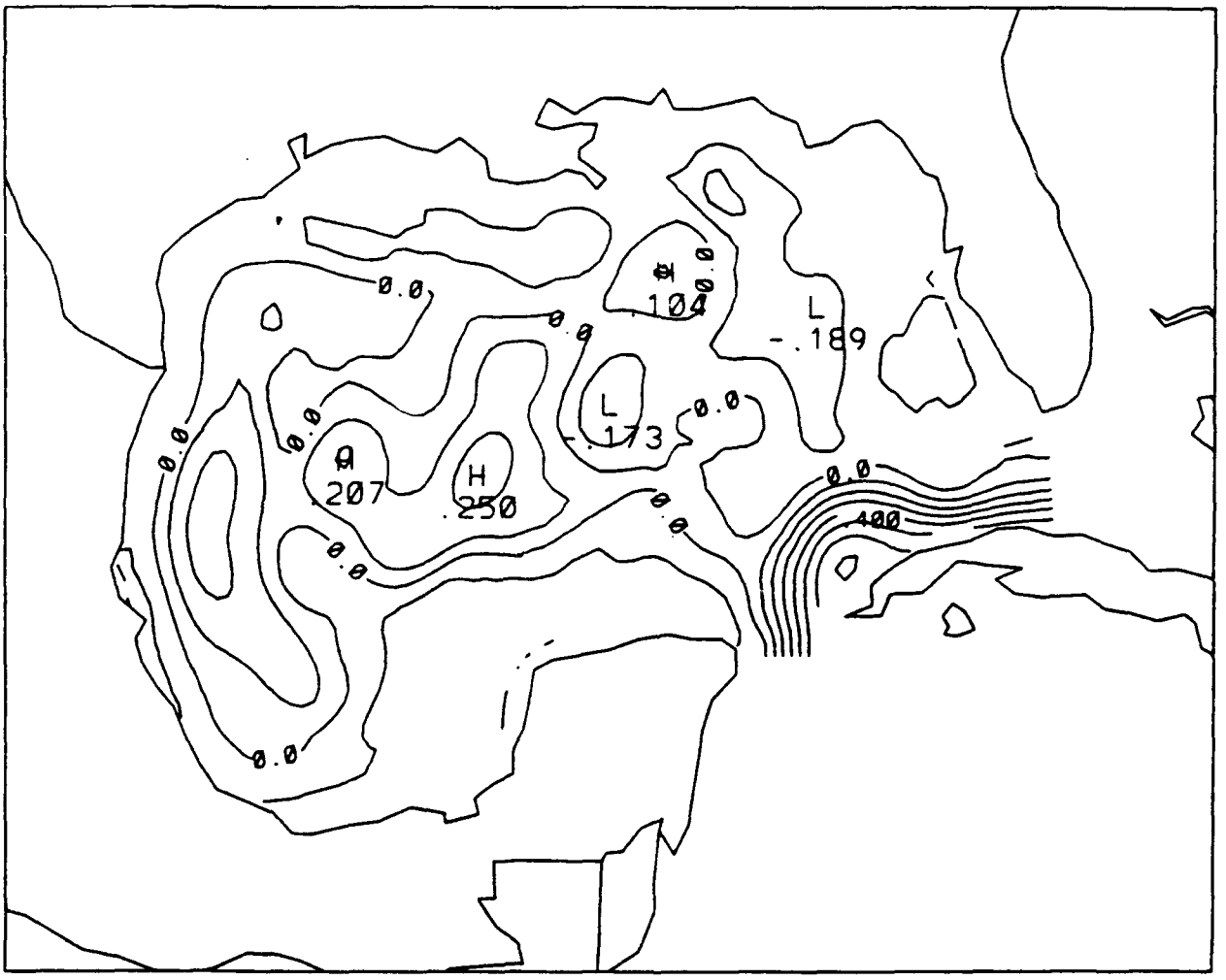

Figure 7c. Test Case – Data Assimilation simulations using vertical temperature profiles: Simulated Sea Surface Height Using Matrix-Mult algorithm.

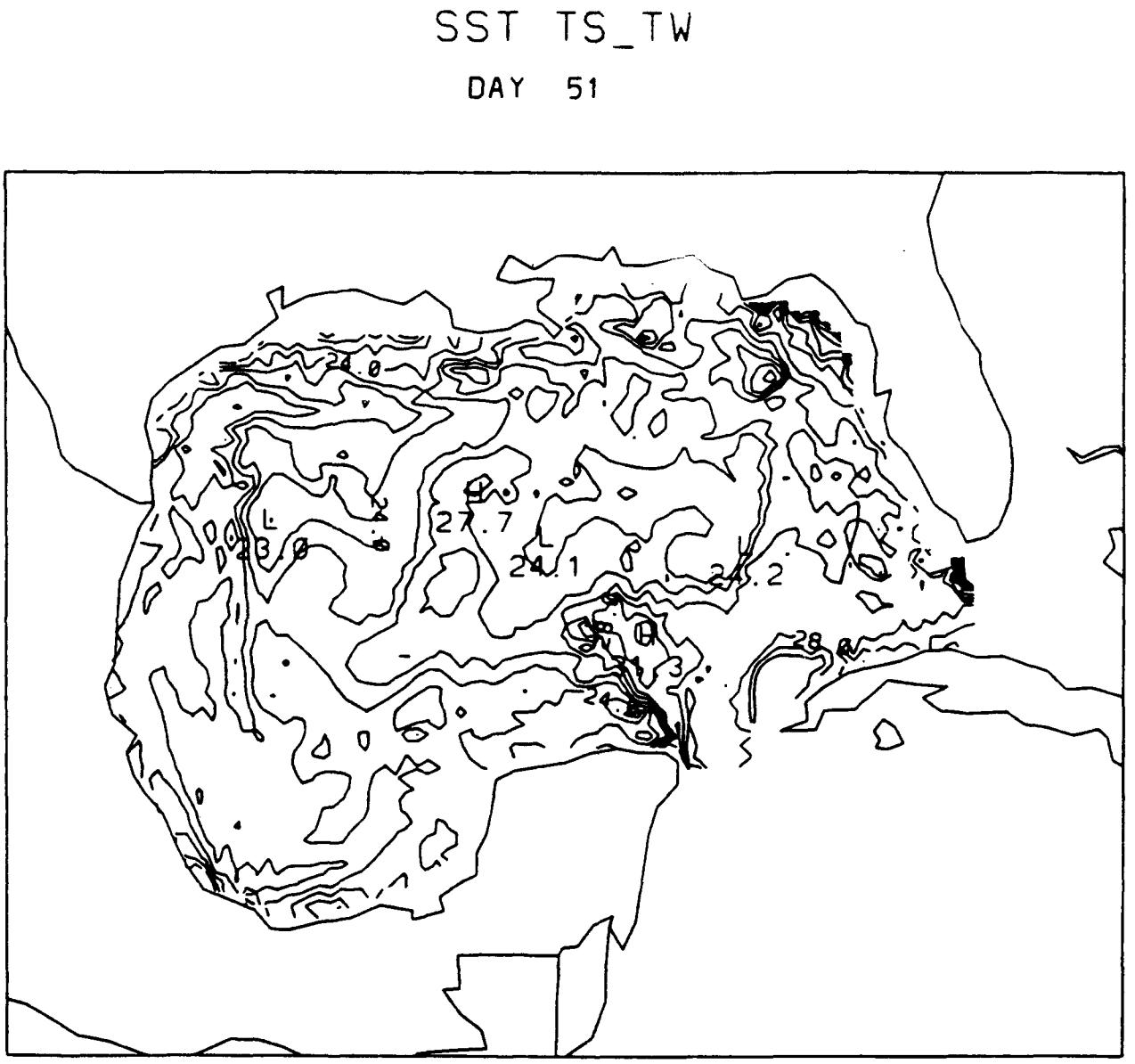

Figure 8a. Test Case – Data Assimilation simulations using vertical temperature profiles: Simulated Sea Surface Temperature Using Derber-Rosati algorithm.

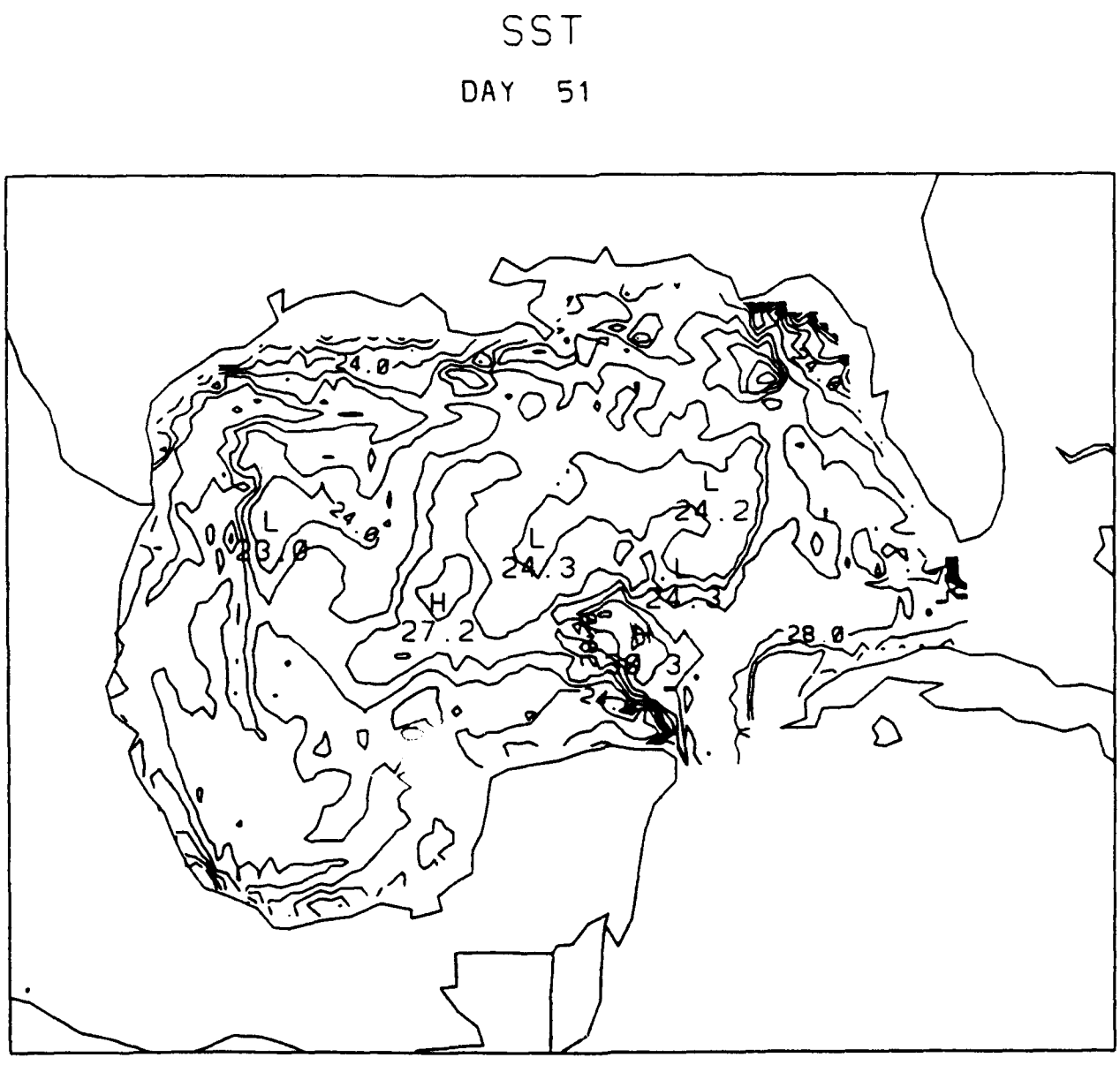

Figure 8b. Test Case – Data Assimilation simulations using vertical temperature profiles: Simulated Sea Surface Temperature Using Product-Polynomial algorithm.

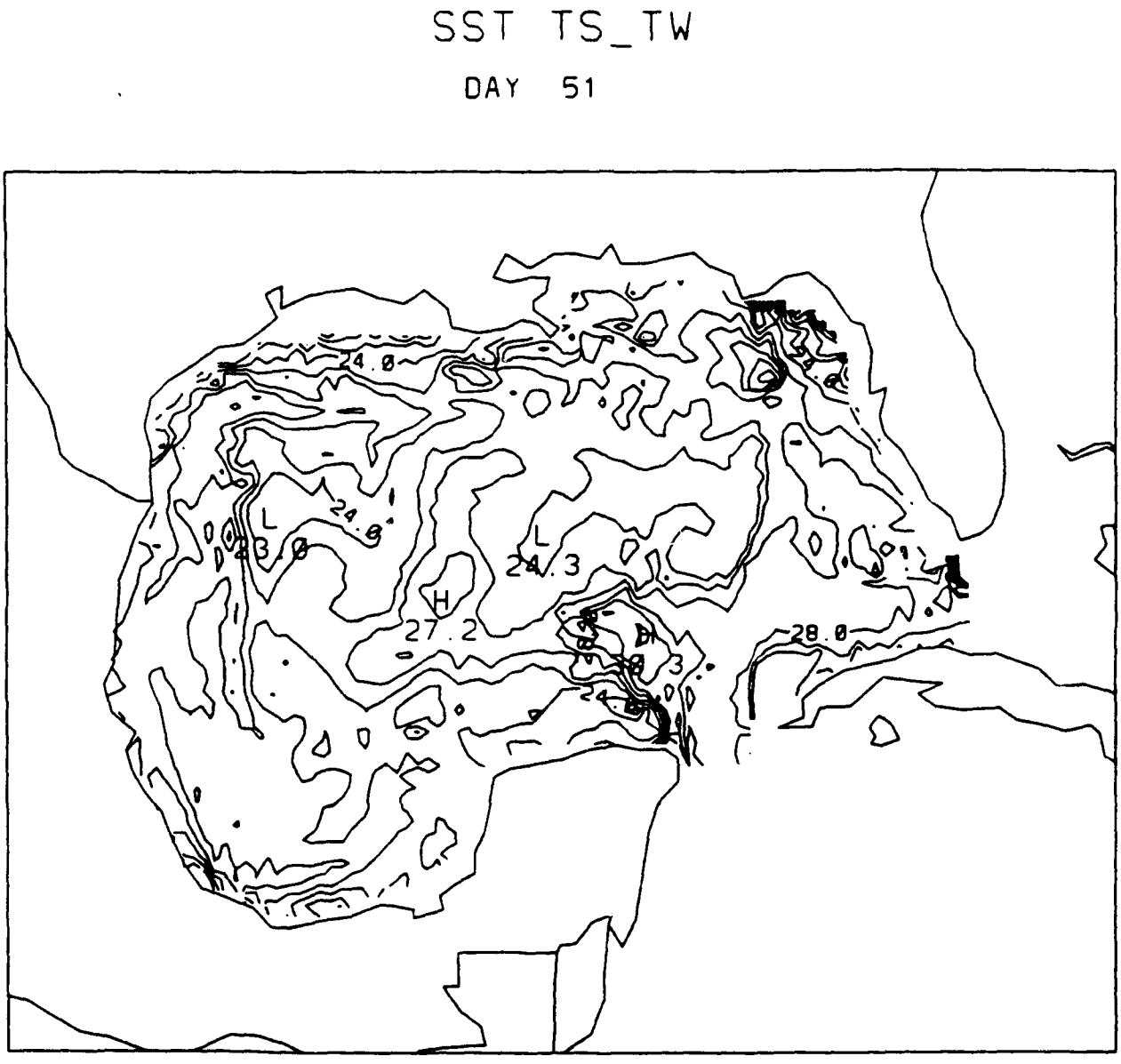

Figure 8c. Test Case – Data Assimilation simulations using vertical temperature profiles: Simulated Sea Surface Temperature Using Matrix-Mult algorithm.

provides a significant computational efficiency over the actual matrix multiplication, using a Laplacian smoother provided the covariance structure was Gaussian. However, the algorithm needed to be made more efficient for it to be useful in practical applications.

Retaining the major features of the DR algorithm, we have derived a new algorithm that introduces several generalizing contributions:

- Identify that $\Sigma_m \mathbf{g}$ is a convolution and, thus, express it in the spectral domain (4.6).
- Identify simple averaging operators (4.5), and show that, with a separable covariance structure, a PPO in the averaging operators applied to $\mathbf{g}$ approximates $\Sigma_m \mathbf{g}$ more accurately and efficiently.

The overall effect is that we have come up with an elegant algorithm that is fast and efficient and computes convolutions supporting a much wider class of covariance structures encountered in the physical sciences.

Although the derivation is shown in two dimensions, the extension to higher dimensions is obvious. Our Fourier transform-pair in one dimension is the integers and the circle group. In higher dimensions, we take the corresponding product of these groups as our Fourier pair.

The algorithm is efficient if one of the functions remains fixed and the convolution is computed many times. This is because we must construct a polynomial fit to a Fourier transform of the fixed factor. If we are able to find a fit within our error tolerance with a low degree polynomial, then we are able to efficiently and accurately estimate the convolution with our polynomial fit applied to a simple averaging operator.

ACKNOWLEDGMENTS

The research described in this paper was initiated at the Institute for Naval Oceanography (INO) and funded by the Navy Ocean Modeling Program. We would like to thank the reviewer for providing constructive comments on the earlier draft which led to a considerable clarification and improvement. Finally, thanks are due to Ms. Susan Sprouse for her technical editing of the report.

REFERENCES

Conte, S.D., and C. deBoor, 1980: *Elementary Numerical Analysis.* McGraw-Hill Inc., New York, 432 pp.

Daley, R., 1991: *Atmospheric Data Analysis*. New York, Cambridge University Press, 457 pp.

Derber, J., and A. Rosati, 1989: A global data assimilation system. *J. Phys. Ocean.*, 1333-1347.

Folland, G.B., 1984: *Real Analysis: Modern Techniques and their Applications*. John Wiley & Sons. New York, 350 pp.

Goodrich, R.K., R.M. Passi and M. Limber, 1992: A large scale Gaussian smoothing algorithm. *Proc. 24th Symposium on the Interface: Computing Science and Statistics*, 24, 380-84.

Lorenc, A., 1986: Analysis methods for numerical weather prediction. *Quart. J. Roy. Meteor. Soc.*, 112, 1177-94.

Navon, I.M., and D.M. Legler, 1987: Conjugate-gradient methods for large-scale minimization in meteorology. *Mon. Wea. Rev.*, 115, 1479-1502.

Parks, T.N., and C.S. Burns, 1987: *Digital Filter Design*. Wiley Interscience, New York, 342 pp.

Passi, R.M., and L.H. Kantha, 1992: A dynamic climatology of the Gulf of Mexico. An internal COAM Report.

Ralston, A., and P. Rabinowitz, 1978: *A First Course in Numerical Analysis*. McGraw-Hill Inc., New Yc '., 556 pp.

Royden, H. L., 1968. *Real Analysis*, 2nd Ed.. MacMillan Publishing Co., Inc., New York, 347 pp.

Thiebaux, H.J., and M.A. Pedder, 1987: *Spatial Objective Analysis with Applications in Atmospheric Science*, Academic Press, New York, 299 pp.

Thompson, P.D., 1956: Optimum smoothing of two-dimensional fields. *Tellus*, 8, 384-393.

DISTRIBUTION LIST

Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000
Thomas Curtin
Emanuel Fiadeiro
Thomas Kinder
Robert Peloquin (3)

Eric Hartwig
Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, D. C. 20375-5000

Naval Research Laboratory
Building 1103, Room 248
Stennis Space Center, MS 39529
George Heburn
Harley Hurlburt
Edward Johnson
John Kindle

John Hovermale (2)
Naval Research Laboratory
Monterey, CA 93943-5006

Don Durham
Naval Oceanography Command
Building 1020
Stennis Space Center, MS 39529

Fleet Numerical Oceanography Center
Monterey, CA 93943
Mike Clancy
Jim Cummings

Naval Oceanographic Office
Stennis Space Center, MS 39529
Michael Carron
Charles Horton

Robert Haney
Naval Postgraduate School
Monterey, CA 93943-5100

A. D. Kirwin
Department of Oceanography
Old Dominion University
Norfolk, VA 23529-0276

George Mellor
Princeton University
P. O. Box CN710, Sayre Hall
Princeton, NJ 08544

Tal Ezer
Princeton University
P. O. Box 308
Princeton, NJ 08540

Woods Hole Oceanographic Institution
Woods Hole, MA 02543
Ken Brink
William Schmitz

Harvard University
29 Oxford Street, Room 100D
Cambridge, MA 02138
Allan R. Robinson
Hernan Arango

Paola Rizzoli
Massachusetts Institute of Technology
Department of Earth, Atmospheric & Planetary Science
Cambridge, MA 02139

National Meteorological Center
5200 Auth Road
Camp Springs, MD 20746
Ron McPherson
James Purser

Minerals Management Service
381 Elden Street MS 4310
Herndon, VA 22070-4817
Bob Labelle
Ronald Lai

Oregon State University
Department of Oceanography
Corvallis, OR 97331-2849
John Allen
Andrew Bennett

UCAR Library
P. O. Box 3000
Boulder, CO 80307

National Center for Atmospheric
Research
P. O. Box 3000
Boulder, CO 80303
Richard Anthes
William Holland
Ying-Hwa Kuo
Warren Washington

Donna Blake
Division of Atmospheric Science
Programs
National Science Foundation
1800 G Street, NW
Washington, D. C. 20550

Rutgers University
P. O. Box 231
New Brunswick, NJ 08903-0231
Scott Glenn
Dale Haidvogel

Murray Brown
Minerals Management Service
1201 Elmwood Park Boulevard
New Orleans, LA 70123-2394

University of Rhode Island
Department of Oceanography
Kingston, RI 02881
Peter Cornillion
Randy Watts

NOAA, National Severe Storms Laboratory
1313 Halley Circle
Norman, OK 73069
John Lewis
Raul Lopez

NOAA, Environmental Research Laboratory
NOAA/ERL/FSL R/E/FS1
325 Broadway
Boulder, CO 80303
Thomas W. Schlatter
Stanley G. Benjamin

Library (3 copies)
NRL (Code 7125)
Stennis Space Center, MS 39529

John Leese
Department of Meteorology
University of Maryland
College Park, MD 30742

William Schramm
NOAA Center for Ocean Analysis and
Prediction
2560 Garden Road
Monterey, CA 93940

Peter Ranelli
SPAWARS 165
5 Crystal Park, Room 301
Arlington, VA 22202

Lie-Yauw Oey
Dept. of Civil Engineering
Stevens Institute of Technology
Hoboken, NJ 07030

Center for Air Sea Technology
Mississippi State University
Building 1103, Room 233
Stennis Space Center, MS 39529
Harsh Anand
Jim Corbin
Lanny Yeske

NOAA, National Ocean Services
Suite 611
1825 Connecticut Avenue, NW
Washington, D. C. 20235
Dave Evans
Robert Cheney

Louise Perkins
The University of Southern Mississippi
Computer Science Department
Building 1103
Stennis Space Center, MS 39529

David Goodrich
Office of Global Programs
NOAA
1100 Wayne Avenue, Suite 1225
Silver Springs, MD 20910

Herbert C. Eppert
Director, Ocean Sciences Directorate
NRL (Code 7300)
Stennis Space Center, MS 39529

Joseph McCaffrey, Jr.
Head, Ocean Sensing and Prediction Division
NRL (Code 7320)
Stennis Space Center, MS 39529

Albert W. Green, Jr.
NRL (Code 7330)
Building 1105
Stennis Space Center, MS 39529

Edward H. Barker
NRL West (Code 440)
Monterey, CA 93943-5006

William B. Moseley
Technical Director
NRL (Code 7320)
Stennis Space Center, MS 39529

Administrative Grants Officer
Office of Naval Research
Resident Representative N66020
Administrative Contracting Officer
Georgia Institute of Technology
206 O'Keefe Building
Atlanta, GA 30332-0490

Director, Naval Research Laboratory
Attn: Code 2627
Washington, D. C. 20375

Defense Technical Information Center (2)
Building 5, Cameron Station
Alexandria, VA 22304-6145

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. Agency Use Only *(Leave blank).*

2. Report Date. October 1993

3. Report Type and Dates Covered. Technical Report

4. Title and Subtitle.

AN EFFICIENT DATA ASSIMILATION ALGORITHM WITH A GAUSSIAN COVARIANCE STRUCTURE

5. Funding Numbers.

Program Element No.

Project No.

Task No.

Accession No.

6. Author(s).

Ranjit M. Passi
Kent Goodrich
John C. Derber
Mark Limber

7. Performing Organization Name(s) and Address(es).

Center for Ocean & Atmospheric Modeling
The University of Southern Mississippi
Building 1103, Room 249
Stennis Space Center, MS 39529-5005

8. Performing Organization Report Number.

TR-2/94

9. Sponsoring/Monitoring Agency Name(s) and Address(es).

Office of Naval Research
Code 1513: RKL
Ballston Tower One
800 North Quincy Street
Arlington, VA 22217-5660

10. Sponsoring/Monitoring Agency Report Number.

11. Supplementary Notes.

ONR Research Grant No. N00014-92-J-4112

12a. Distribution/Availability Statement.

Approved for public release; distribution is unlimited.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

Details are provided of our attempts to develop computationally efficient and widely acceptable data assimilation algorithms. Our initial data assimilation work was done with a scheme which was developed by Derber and Rosati (1989) for their global data assimilation application. The scheme is quite efficient: but when it was combined with a primitive equation, numerical model of the Gulf of Mexico, the nowcasting step took too much computer time for it to be useful in real-time applications. So, after an initial implementation and test of the scheme, our obvious thrust was to enhance and make it more efficient.

Most of the computation resources while using the Derber-Rosati scheme are required in approximating a matrix product $\Sigma_m g$, where Σ_m is the covariance matrix of the model output error having a Gaussian spatial covariance structure that stays fixed, and g is a varying vector. They approximate this product by repeated applications of a Laplacian operator.

A new algorithm is developed that, while maintaining the accuracy, approximates $\Sigma_m g$ far more efficiently than the Laplacian algorithm. It also admits a wider class of covariance structures with equal ease and efficiency. The new algorithm approximates $\Sigma_m g$ by applying a product of two operators that are polynomials in simple averaging operators. The algorithm is efficient and general with the only restriction that Σ_m be based on a covariance function that is a product of factors, which are even functions in a single dimension.

14. Subject Terms.

(U) Data assimilation, (U) Computationally efficient, (U) Covariance function, (U) Gaussian, (U) Laplacian, (U) Separable functions, (U) Operator polynomials

15. Number of Pages. 45

16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
Unclassified	Unclassified	Unclassified	SAR