UNIVERSITY OF OKLAHOMA
GRADUATE COLLEGE

COMPUTATIONAL ASPECTS OF THE ADJOINT METHOD

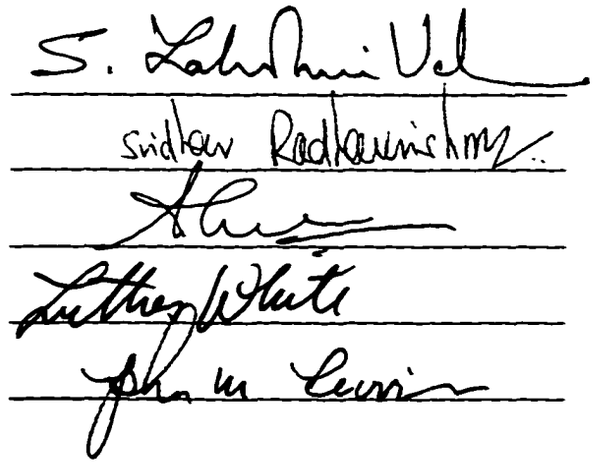
A Dissertation
SUBMITTED TO THE GRADUATE FACULTY
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy

By
RACHEL FIEDLER
Norman, Oklahoma
1997

COMPUTATIONAL ASPECTS OF THE ADJOINT METHOD

A Dissertation APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

By



Dedication

This dissertation is dedicated to Lisa and Greta.

Acknowledgements

I would like to thank Dr. Lakshmivarahan and Dr. John Lewis for their expert advice and patience. I thank Drs. Dhall, Radhakrishnan and White for their wise counsel. I also thank my husband for his expert help with LaTeX.

Contents

List Of Tables

List Of Figures

Abstract

Modern numerical weather prediction often employs an adjoint of the forecast model to optimally initialize the model with observations of the state of the atmosphere and thus to improve the accuracy of the forecast. This dissertation investigates the computational aspects used for the adjoint methods, and in particular the effectiveness of parallel algorithms. For demonstration purposes, we consider these algorithms applied to a simple, linear, shallow-water model, which is often used in real world problems.

A forecast model, with its adjoint, can also be used as a tool to optimally determine parameters in physical laws, without the intent of using the forecast beyond the assimilation period. This dissertation also investigates such a use of an adjoint, in an investigation of an atmospheric mixed layer.

Chapter 1

Introduction

Meteorologists have successfully demonstrated the use of deterministic models to describe the space-time evolution of different types of weather phenomena - the barotropic model, the primitive equation model, to mention a few. A distinguishing feature of the deterministic model is that the solution of the model equations (assuming existence and uniqueness) is solely dependent on initial and/or boundary conditions and the values of key parameters in the model. Since the initial and boundary conditions and the parameters together control the evolution of the solution of the model equations, these are often called "control" variables. In using the deterministic model to predict the evolution of a physical process, clearly, the quality of the prediction is critically dependent on the accuracy of the control parameters. Consequently, a basic question in this context is: how to go about determining values of control variables that would lead to a more accurate prediction?

A common approach to this problem calls for observing the states of the system being modeled over a period of time. The aim is then to use the collected data to determine or estimate the values of the control variables such that the solution of the dynamical system based on the estimates closely "resembles" the observed data. For example, either the least mean square error or the maximum likelihood estimate can be one of the many possible criteria for measuring closeness. Stated in other words, the problem is to absorb or assimilate the "information" contained in the observed data into the control variables. Consequently, this problem is often called the "data assimilation" problem (Daley 1991, Ghil & Malanotte-Rizzoli 1991).

Formally, let c be a control vector of size m and S denote the feasible region for the control vector. For any c in S, let $J(c)$ denote the weighted sum of the squared

difference between the observation and the solution of the model corresponding to the control vector c. Except in trivial cases the explicit form of J as a function of c is not known. It is to be emphasized that there are other possible choices for the J function. One may be interested only in the state of the model at a given time instant, say $t = \Delta$. Whatever be the nature and type of the J function, mathematically, the data assimilation problem can be stated as follows : find a c^* in S such that $J(c^*)$ is a minimum, that is, we are lead to an optimization problem under the dynamical constraints of the model equations. Since J is a "smooth" function, one method for finding c^* is to use one of the many variants of the classical gradient method. This is however, more easily said than done. The difficulty primarily stems from the fact that J is not known explicitly. A now popular method for finding the gradient of J is called the "adjoint" method (Ghil & Malanotte-Rizzoli 1991, Thacker & Long 1988). A summary of data assimilation using the adjoint method is shown in Fig. 1.1 and described below:

Step 1: Make a first estimate of the initial condition (I.C.).

Step 2: Run the forward model to generate a forecast over the assimilation interval.

Step 3: Compute the cost function, J, using the observations and the model output.

Step 4: Compute the gradient of the cost function with respect to the control variable, $\nabla J(c)$, by integrating the adjoint model (which is a combination of the forward model and cost functional) backwards using the Lagrangian multiplier method.

Step 5: Generate a new estimate of the initial condition (I.C.) for another forecast using a minimization procedure (*e.g.*, Steepest Descent or Conjugate Gradient Method). This is done iteratively toward the minimum of the cost function using the calculated cost function and its gradient.

Step 6: The process is repeated until some convergence criteria are met, *i.e.*, the cost functional, J, is near its local minimum. In practice, this is determined by an insignificant change in the gradient from one iteration to the next. If not, the process will be terminated when some maximum number of iterations have been achieved.

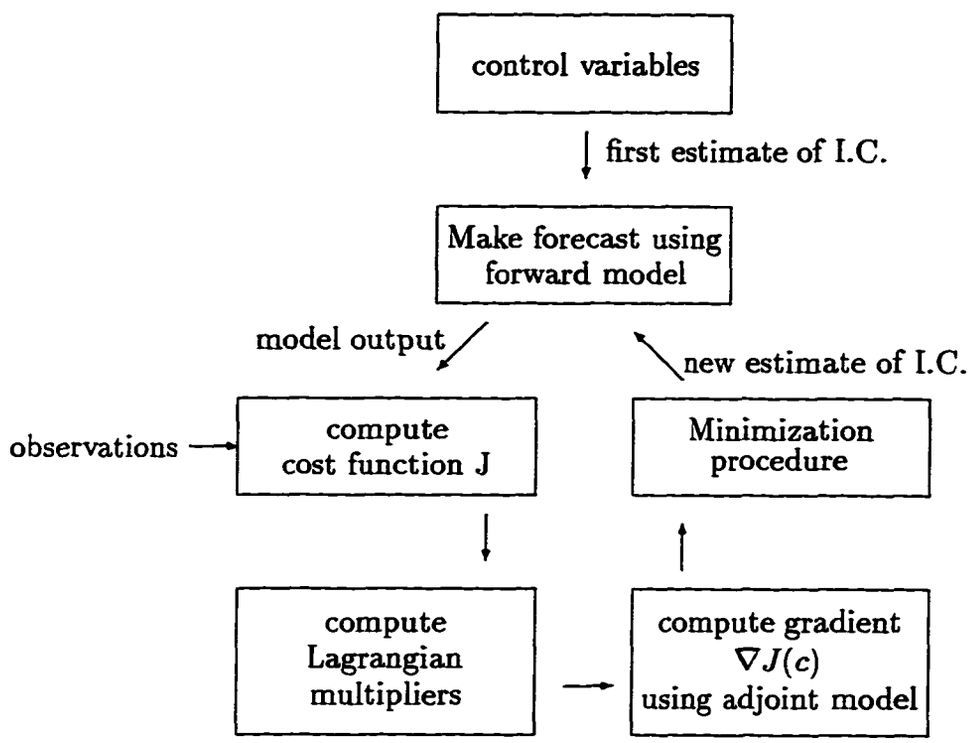


Figure 1.1: Flowchart of data assimilation using the adjoint method.

Considerable success has been reported in the literature in the use of adjoint method for finding c^* (Luenberger 1973, Sun & Flicker 1991, Wolfsberg 1987). The success of this combination is largely dependent on the properties of the J function. This approach can succeed only if J is unimodal in S. It turns out that the modality of J critically depends on the model dynamics. It is now known that the nonlinearity in the model dynamics induces multimodality in the J function (Li, Droegemeier & Lewis 1991, Chung 1996).

There are at least two factors affecting the rate of convergence of the iterates leading to the optimum value of the J function. First, is the number and distribution of the observations. There is a minimum number of observations required from an algorithmic viewpoint, but satisfaction of this requirement is insufficient to guarantee a solution. The distribution of these data (in space and time), in concert wtih the dynamics, dictates the existence of a solution. Second, is the shape of the J function. Judicious choice of scaling can remove eccentricities in the J-field. It is thus imperative to understand the role of these two factors affecting the iterates.

The effect of the number and distribution of observations on the quality of the iterates are often examined using controlled experiments which have come to be known as the "twin" experiments. In this, a point in the feasible region for the control vector is first chosen and then the model solution is calculated for this value of the control vector . Then observations (including known error) are generated from the model solution by adding noise with known characteristics. By computing the optimal estimate of the control vector for different sets of observations, we can develop a better understanding of the dependence of the optimal estimate on the number, distribution and accuracy of observations.

As for the shape of the J function, since it is not known explicitly, we must be contented with the analysis of the properties of J around the local minima. This is often done by approximating J around the optimum c^* using a quadratic form such as

$$J(c) = \frac{1}{2}c^t Hc + p^t c + q, \tag{1.1}$$

where H is the Hessian matrix which is a symmetric matrix of the second derivatives of J with respect to c,

$$p = (p_1, p_2, p_3, p_4, ..., p_n)^t \tag{1.2}$$

is a vector and q is a constant. By analyzing the eigenvalues of H, we can draw inferences on the shape of J in the vicinity of c^* (Thacker 1989). For example, if one of the eigenvalues of H is very small, then the contours or the level curves of J for the valley around c^* are elongated ellipses. This would imply that the iterative process of locating the minimum would converge slowly and with difficulty.

In this dissertation, one aim is to apply the adjoint method to the mixed layer model (Ball 1960, Lilly 1968). This model is used to predict the return flow of the warm, humid air from the Gulf of Mexico into the coastal plains during the winter months. The mixed layer model has a small number of unknown variables/parameters. Athough small, the model is nonlinear and thus presents difficulties in dynamical optimization. With the experience gained, we wanted to analyze the computational aspects of data assimilation. This brought us to the shallow water equations (Pedlosky 1987), which has a larger number of variables in the discrete model. We are interested in solving some of the computational aspects of this model that are relevant to data assimilation. Since the solution of the forward model and the adjoint take a good fraction of the efforts, we turned our attention to parallel methods for solving this class of equations.

The shallow water model is discretized first using the Euler scheme. But due to its instability, we also examined the solution using a stable leapfrog scheme. Both types of discretization resulted in a block bi-diagonal system of equations for the forward model and the adjoint model. We are interested in examining the comparative analysis of four vector/parallel algorithms in solving this system of equations. These four vector/parallel algorithms belong to a class of direct parallel methods. The first algorithm is also known as the "divide and conquer" method (Lakshmivarahan & Dhall 1990). The second algorithm is a variation of the divide and conquer method (Lakshmivarahan & Dhall 1990, Conn & Podrazik 1994, Meyer & Podrazik 1987, Van der Vorst 1988, Meyer & Podrazik 1989). The third algorithm is known as the partition algorithm (Van der Vorst & Dekker 1989). The fourth algorithm is the cyclic reduction method (Lakshmivarahan & Dhall 1990). A comparison of these four vector/parallel algorithms is done on the CrayJ90 in scalar mode and using 1,2,4 and 8 vector processors.

This dissertation is organized as follows. A description of the data assimilation problem using the Lagrangian multiplier method for the mixed layer model is given

in Chapter 2. The mixed layer model equations are described in Section 2.2. Using a standard Euler discretization scheme, discrete version of the model equations are given in Section2.2. Section 2.3 contains a derivation of a linear version of the discrete dynamical equation that govern the evolution of perturbations in the initial state. This system is known as the linear tangent model in the contemporary meteorological literature. The structure of the objective function J is described in Section 2.4. Since the adjoint method leads to minimizing J under the dynamical constraints of the model, the associated Lagrangian L is derived in Section 2.5. Conditions for the optimality and an expression for the derivative, $\frac{\partial J}{\partial c}$, are also derived in Section 2.5. Section 2.6 is devoted to the analysis of the quality of estimates and the structure of the $J(c)$ function near the optimum point. It deals with the so called "twin experiment" which in a sense helps to check the quality of estimates. Analysis of the structure of the J function in the neighborhood of the minimum using the Hessian (Thacker 1989, Callies & Eppel 1992) of J at c^* is contained in Section 2.6. The actual data assimilation using data collected from the Gulf is contained in Section 2.7.

Discretization of the shallow water model is given in Chapter 3. Section 3.2 gives the continuous form of the shallow water equations. Section 3.3 gives the discretization using the Euler scheme. The forward model equations are given in Section 3.3.1. The structure of the objective function J is described in Section 3.3.2. The adjoint model equations are given in Section 3.3.3. Sections 3.4.1 through 3.4.3 describe the forward and adjoint models for the leapfrog discretization. A discussion of all related matrix properties that resulted from both the Euler and leapfrog schemes is presented in Section 3.5.

The shallow-water model provides motivation for solving the block bi-diagonal system. A comparative study of four vector/parallel algorithms used for solving this system is given in Chapter 4. Four algorithms are defined in Section 4.2, with benchmarks presented in Section 4.3. Section 4.4 discusses the efficiencies and effectiveness of these algorithms. Conclusions and suggestions for future work are offered in Chapter 5.

Chapter 2

Data Assimilation Using the Lagrangian Multiplier Method - The Mixed Layer Model

2.1 Introduction

During winter in the northern hemisphere, cold dry air invades the continental United States. Roughly 25-30 of these air masses manage to move over the warm waters of the Gulf of Mexico each year during the cool season from October to March. Through a convective process known as turbulent "mixing", a transfer of sensible heat and water vapor from the warm water to the cold air takes place. The dynamical equations describing this transfer of heat and vapor is called the mixed layer model. As the air mass moves over the water, the air next to the ocean gets warmer and more moist. Subsequently, this modified air tracks back to the Gulf's coastal plain. The interest in this return flow stems from the fact that the confluence of this warm humid air flow with the next cold front has the potential to produce severe weather. The mixed layer model has been successfully used to capture this facet of warming of the cold air and the return flow (Burk & Thompson 1992, Liu, Lewis & Schneider 1992).

Recently, a team of scientists from the National Severe Storms Laboratory (Lewis, Hayden, Merrill & Schneider 1989, Crisp & Lewis 1992, Lewis & Crisp 1992) spent several months observing and collecting data related to these return flows. Our aim is to use these data in conjunction with the adjoint method to determine the best value for the control parameters. Using this estimate, we then predict the thermodynamic structure of the air involved in the return flow.

2.2 The Mixed Layer Model

The mixed layer model consists of three coupled non-linear ordinary differential equations in time t :

$$\frac{d\theta(t)}{dt} = \frac{1}{h(t)} C_T V (1 + K) [T(t) - \theta(t)] \tag{2.1}$$

$$\frac{dh(t)}{dt} = K C_T V \frac{[T(t) - \theta(t)]}{\sigma(t)} + W \tag{2.2}$$

$$\frac{d\sigma(t)}{dt} = -\frac{d\theta(t)}{dt} + \gamma \frac{dh(t)}{dt} \tag{2.3}$$

where

$\theta(t)$ - is the potential temperature (C),

$h(t)$ - is the height of the mixed-layer (m),

$\sigma(t)$ - denotes the jump in at the top of the layer (C),

$T(t)$ - is the sea surface temperature (C),

γ - refers to the gradient of θ above the mixed layer (Cm^{-1}),

C_T - is the transport (turbulent mixing) coefficient,

V - is the surface wind velocity (ms^{-1}),

K - denotes the entrainment coefficient,

W - denotes the vertical velocity affecting the height of the mixed layer (ms^{-1}).

The value of $C_T = 0.001$, $V = 10ms^{-1}$, $\gamma = 0.004C\ m^{-1}$ and $T(t)$ is a given function of time, specified in Section 2.7. Refer to Figure 2.1 for a schematic diagram corresponding to this model.

In Figure 2.1, the temperature profile is partitioned into three layers : (1) a so-called superadiabatic layer ($\frac{\partial\theta}{\partial z} < 0$) on the bottom, (2) an adiabatic layer ($\frac{\partial\theta}{\partial z} = 0$) in the middle, and (3) a subadiabatic ($\frac{\partial\theta}{\partial z} > 0$) layer on top.

At the top of the adiabatic layer, a discontinuity or jump, σ, in θ is depicted. When the low level air comes into contact with the water as it streams from continent

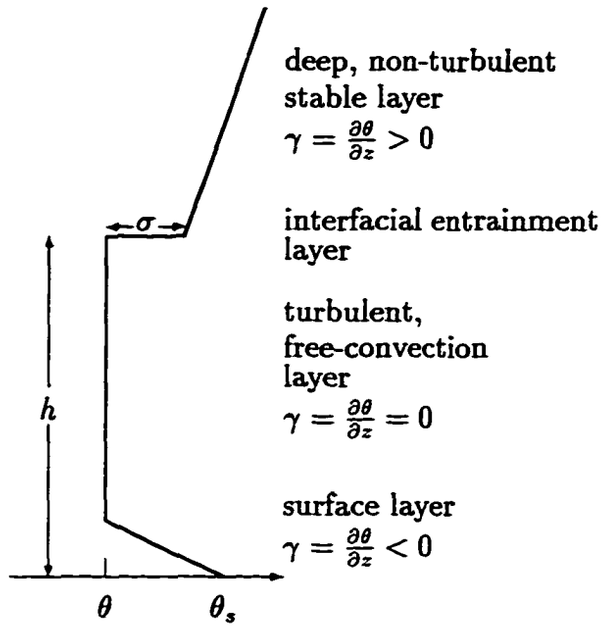


Figure 2.1: A schematic diagram for the mixed layer model.

to ocean, these lowest layers are much colder ($10C$) than the underlying water. A strong vertical gradient of potential temperature develops next to the ocean surface, typically 10-50 meters thick. (This is the surface layer in the diagram). In this environment of a warm lower boundary, heat is transferred to the air as the result of turbulent motion. (In much the same way as water is warmed in a pan that is heated over the flames or burner of a stove.) The air parcels or eddies in the turbulent layer "eat away" at the stable layer above ($\frac{\partial \theta}{\partial z} > 0$) and there is a transition zone at the top of the "mixing layer" where the air from above is entrained down into it. If there is a mean vertical component of wind velocity W, then this acts to either vertically suppress ($W < 0$) or vertically expand ($W > 0$) the layer where mixing is taking place. The layer where these turbulent billows or eddies are thrashing about is called the mixed layer. The state of complete mixing is typified by a uniform potential temperature θ.

Thus, the mixed-layer model is designed to describe the evolution of θ, h and σ in the "middle portion" of the diagram. The layer below is the source of external

forcing (heating from below) and the top layer is the "cap" or brake to keep the mixed layer contained. The heat transport from ocean to air is contained in the $C_T V$ terms, whereas the entrainment of air from above (also a source of heating since θ is greater in the top layer than it is in the middle or mixed layer) is controlled by K terms. W acts as an external forcing on the height of the layer.

A distinguishing feature of this model is that it is spacially homogeneous and is independent of the usual space variables. For purposes of our analysis and data assimilation, the control parameter c consists of three initial conditions, θ_o, h_o and σ_o and two parameters K and W, that is,

$$c = (\theta_o, h_o, \sigma_o, K, W)^t \tag{2.4}$$

where t denotes the transpose. We will assume that the $C_T V$ and γ parameters are known to a much greater degree of accuracy than the set of five parameters we have chosen as our control set. In practice, this is a good assumption. Note that the five control parameters are dimensionally quite dissimilar. This model is a result of the seminal work of Ball (1960) and Lilly (1968) and the reader is referred to these publications for the details of its derivation. Also refer to Busch (1977), Carson (1973) and Tennekes (1973) for further details. This model has enjoyed success in those situations which are typified by strong surface heating over land and water.

In discretizing the model equations (2.1) - (2.3), we use the general Euler scheme that approximates the time derivatives using the forward difference. Let τ denote the time between two successive time intervals and n be the total number of time steps. Let $\theta(i\tau)$ be denoted as θ_i and likewise h_i and σ_i denote $h(i\tau)$ and $\sigma(i\tau)$, respectively. The discrete version of (2.1) - (2.3) then corresponds to

$$\theta_{i+1} = \theta_i + \frac{\tau}{h_i} C_T V (1 + K) (T_i - \theta_i), \tag{2.5}$$

$$h_{i+1} = h_i + \tau K C_T V \frac{[T_i - \theta_i]}{\sigma_i} + \tau W, \tag{2.6}$$

$$\sigma_{i+1} = \sigma_i - \frac{\tau}{h_i} C_T V (1 + K) (T_i - \theta_i) + \gamma \tau K C_T V \frac{[T_i - \theta_i]}{\sigma_i} + \gamma \tau W, \tag{2.7}$$

where $0 \leq i \leq n$. The time interval $n\tau$ is called the *assimilation period.*

Although the Runge-Kutta schemes have less truncation error than the Euler scheme, we choose the Euler scheme because it is explicit and makes the derivation of the adjoint equations easier. This must be compensated by using an especially small time step in the model integration. For more complex models where computational time is a key factor, the luxury of using an Euler scheme would not be warranted.

It is assumed that the values of C_T, V, γ and T (sea-surface temperature) are given. Thus, the time evolution of the system (2.5) – (2.7) depends only on the control vector c.

2.3 Perturbation Analysis: Tangent Linear Model

For purposes of later comparison between the structure of the model equations and that of the adjoint equations, we derive the perturbation equations. We neglect perturbations in the empirical parameters (K and W) since they appear as forcing functions in the perturbation equations that are invariant in time. Inclusion is not difficult but it is also not very informative so this simplification is justified physically.

Let

$$\bar{c} = \left(\bar{\theta}_0,\ \bar{h}_0,\ \bar{\sigma}_0,\ \overline{K},\ \overline{W}\right)^t \tag{2.8}$$

be a point in the control space. The solution of (2.5)-(2.7) based on $\bar{c}$ is called the *base state* of the model and is denoted as $\bar{\theta}_i$, $\bar{h}_i$ and $\bar{\sigma}_i$. for $1 \leq i \leq n$. Let ϵ_0^θ, ϵ_0^h and ϵ_0^σ be (small) perturbations in $\bar{\theta}_0$,$\bar{h}_0$ and $\bar{\sigma}_0$ respectively. Thus, consider a new point

$$c = \left(\bar{\theta}_0 + \epsilon_0^\theta,\ \bar{h}_0 + \epsilon_0^h,\ \bar{\sigma}_0 + \epsilon_0^\sigma,\ \overline{K},\ \overline{W}\right)^t. \tag{2.9}$$

Let θ_i, h_i and σ_i be the solution of (2.5)-(2.7) arising from c. From (2.5)-(2.7) we obtain that

$$\theta_1 = \left(\bar{\theta}_0 + \epsilon_0^\theta\right) + \frac{\tau}{\left(\bar{h}_0 + \epsilon_0^h\right)} C_T V \left(1 + \overline{K}\right)\left(T_0 - \bar{\theta}_0 - \epsilon_0^\theta\right), \tag{2.10}$$

$$h_1 = \left(\bar{h}_0 + \epsilon_0^h\right) + \tau \overline{K} C_T V \frac{\left(T_0 - \bar{\theta}_0 - \epsilon_0^\theta\right)}{\left(\bar{\sigma}_0 + \epsilon_0^\sigma\right)} + \tau \overline{W}, \tag{2.11}$$

$$\sigma_1 = (\overline{\sigma}_0 + \epsilon_0^\sigma) - \frac{\tau}{(\overline{h}_0 + \epsilon_0^h)} C_T V (1 + \overline{K}) (T_0 - \overline{\theta}_0 - \epsilon_0^\theta) \tag{2.12}$$
$$+\gamma\tau\overline{K}C_T V \frac{(T_0 - \overline{\theta}_0 - \epsilon_0^\theta)}{(\overline{\sigma}_0 + \epsilon_0^\sigma)} + \gamma\tau\overline{W}.$$

By definition,

$$\begin{aligned} \epsilon_i^\theta &= \theta_i - \overline{\theta}_i, \\ \epsilon_i^h &= h_i - \overline{h}_i, \\ \epsilon_i^\sigma &= \sigma_i - \overline{\sigma}_i, . \end{aligned} \tag{2.13}$$

Let

$$z_i = (\epsilon_i^\theta,\ \epsilon_i^h,\ \epsilon_i^\sigma\ ). \tag{2.14}$$

By neglecting all the second order terms involving the perturbations, we obtain the following linear relation between z_1 and z_0. That is,

$$z_1 = D_0 z_0 \tag{2.15}$$

where the 3×3 matrix D_0 is given by

$$D_0 = \begin{bmatrix} 1 - \frac{\tau C_T V(1+\overline{K})}{\overline{h}_0} & -\frac{\tau C_T V(1+\overline{K})(T_0 - \overline{\theta}_0)}{\overline{h}_0^2} & 0 \\ -\frac{\tau C_T V \overline{K}}{\overline{\sigma}_0} & 1 & -\frac{\tau C_T V \overline{K}(T_0 - \overline{\theta}_0)}{\overline{\sigma}_0^2} \\ \frac{\tau C_T V(1+\overline{K})}{\overline{h}_0} - \frac{\tau C_T V \gamma \overline{K}}{\overline{\sigma}_0} & \frac{\tau C_T V(1+\overline{K})(T_0 - \overline{\theta}_0)}{\overline{h}_0^2} & 1 - \frac{\tau C_T V \overline{K} \gamma (T_0 - \overline{\theta}_0)}{\overline{\sigma}_0^2} \end{bmatrix}. \tag{2.16}$$

It can be readily seen that the one step propagation of perturbation is indeed given by

$$z_{i+1} = D_i z_i, \tag{2.17}$$

where z_0 is given and D_i is obtained from (2.16) by replacing $\overline{\theta}_0$,$\overline{h}_0$ and $\overline{\sigma}_0$ by $\overline{\theta}_i$,$\overline{h}_i$ and $\overline{\sigma}_i$, respectively.

Rewriting (2.17) for $0 \leq i \leq n-1$ as

$$D_i z_i - z_{i+1} = 0, \tag{2.18}$$

which in turn can be succinctly written in the matrix form as

$$Ez = b, \tag{2.19}$$

where

$$E = \begin{bmatrix} -I_3 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ D_1 & -I_3 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & D_2 & -I_3 & 0 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots & D_{n-1} & -I_3 \end{bmatrix}, \tag{2.20}$$

I_3 is the identity matrix of size 3×3,

$$z = (z_1, z_2, \dots, z_n)^t \tag{2.21}$$

and

$$b = (b_1, b_2, \dots, b_n)^t . \tag{2.22}$$

It can be verified that $b_1 = -D_0 z_0$ and $b_i = 0$ for $i \geq 2$. The matrix (2.20) is called the lower block bi-diagonal matrix. The linear system (2.19) is also known as the *tangent linear model* in the meteorology literature.

The importance of this so-called tangent linear system is that the matrix of the adjoint equation is exactly the negative of the transpose of the matrix E in (2.20). We will discuss more about this in Section 2.5.

2.4 A Performance Measure

Recall that our aim in data assimilation is to find the "best" value for the control vector c^*, such that the solution of the model equation based on c^* closely "resembles" the observed states of the physical system. To develop a mathematically tractable measure of the closeness of the model solution to the observed data, first we need to make several simplifying assumptions.

A1 It is assumed that the observations on the states of the system are available at every value of the time index.

In our case, this implies that the observed values of θ, h and σ namely, $\tilde{\theta}_i$, $\tilde{h}_i$ and $\tilde{\sigma}_i$, are available for all $0 \leq i \leq n$. We hasten to add that this is not often the case in practice. Except for some specially contrived cases, observations in general, may not be available at regularly spaced time and/or space intervals. Since the quality of the estimate for c^* directly depends on the number of available observations, this assumption in spite of its naiveness, provide an ideal set up for the problem on hand. The effect of *not* having *all* the observations and its attendant impact will be discussed in Section 2.6.

It is to be realized that physical measurements are often associated with errors called the measurement error. The following assumptions on the nature and type of this class of errors are standard and greatly simplify the analysis.

A2 The errors are random and that the errors in $\tilde{\theta}_i$ and $\tilde{\theta}_j$ are independent for $i \neq j$, and likewise for $\tilde{h}_i$ and $\tilde{\sigma}_i$.

A3 The errors in $\tilde{\theta}_i$, $\tilde{h}_i$ and $\tilde{\sigma}_i$ are mutually independent.

A4 The random errors are Gaussian with known variance. If this is not the case, then the traditional least squares estimation process should be replaced by the more general maximum likelihood estimation theory that accounts for the non-Gaussian error characteristics. This procedure is more general but also more difficult to handle mathematically.

Thus it is assumed that $\left(\theta_i - \tilde{\theta}_i\right)$ is a mean zero Gaussian random variate with the common variance of δ_θ^2. Likewise, let δ_h^2 and δ_σ^2 be the common variances of the mean zero Gaussian variates, $\left(h_i - \tilde{h}_i\right)$ and $(\sigma_i - \tilde{\sigma}_i)$ respectively.

With these assumptions, define J(c) as

$$J(c) = \sum_{i=0}^{n} \left[\frac{\left(\theta_i - \tilde{\theta}_i\right)^2}{\delta_\theta^2} + \frac{\left(h_i - \tilde{h}_i\right)^2}{\delta_h^2} + \frac{(\sigma_i - \tilde{\sigma}_i)^2}{\delta_\sigma^2} \right]. \tag{2.23}$$

We also experiment with climatology "penalty" terms in the cost function J:

$$J(c) = \sum_{i=0}^{n} \left[\frac{\left(\theta_i - \tilde{\theta}_i\right)^2}{\delta_\theta^2} + \frac{\left(h_i - \tilde{h}_i\right)^2}{\delta_h^2} + \frac{(\sigma_i - \tilde{\sigma}_i)^2}{\delta_\sigma^2} \right] + \frac{\left(\overline{\theta} - \theta_{\text{clim}}\right)^2}{\delta^2_{\text{clim}\theta}} + \frac{\left(\overline{H} - H_{\text{clim}}\right)^2}{\delta^2_{\text{clim}H}} \tag{2.24}$$

where $\theta_{\text{clim}} = 20\ C$, $H_{\text{clim}} = 1000\ m$, $\delta_{\text{clim}\theta} = 3\ C$, and $\delta_{\text{clim}H} = 25\ m$. The purpose of the penalty terms is to limit the range of the solution to climatological values.

Several comments are in order. First, realize that the model solution, θ_i, h_i and σ_i, depend on initial conditions θ_0, h_0 and σ_0 and the *unknown* values of the parameters, K and W, that is, on the control vector

$$c = (\theta_0, h_0, \sigma_0, K, W)^t . \tag{2.25}$$

Since there is no observations available on the parameters K and W, there are no explicit terms in (2.23) involving K and W.

Using the recurrence relation in Section 2.3 which is the discrete counterpart of the model equation, one can in principle express θ_i, h_i and σ_i in terms of θ_0, h_0 , σ_0 , K and W for all $1 \leq i \leq n$. In such a case, we will have J as an explicit function of c with no constraints. Except in trivial and uninteresting cases, such back substitution would be cumbersome and often may not lead to any further insight on the shape of J.

We now restate our data assimilation problem as follows: Given the set of observations, find the value of the control parameter, c, such that $J(c)$ is a minimum when the model solution θ_i, h_i and σ_i depend on c through the model equations (2.5)-(2.7). That is, we are led to an optimization problem under the presence of dynamic constraints.

2.5 The Adjoint Method

The idea is to find the c^* in S such that $J(c^*)$ is the minimum. We propose to find c^* using a variant of the conjugate gradient method, which calls for finding the gradient, ∇J, under the model constraints. However, since J is *not* known

explicitly, this is more easily said than done. In this section, we describe a method which has come to be known as the adjoint method for finding ∇J. To this end, the Lagrangian is introduced (see Lanczos (1970) for the theoretical foundations of minimizing functions subject to a constraint).

$$\begin{aligned} L(c,\lambda,\mu,\eta) &= J(c) \\ &+ \sum_{i=1}^{n} \lambda_i \left[\theta_i - \theta_{i-1} - \frac{\tau}{h_{i-1}} C_T V (1+K)(T_{i-1} - \theta_{i-1}) \right] \\ &+ \sum_{i=1}^{n} \mu_i \left[h_i - h_{i-1} - \tau K C_T V \frac{(T_{i-1} - \theta_{i-1})}{\sigma_{i-1}} - \tau W \right] \\ &+ \sum_{i=1}^{n} \eta_i \left[\sigma_i - \sigma_{i-1} + \frac{\tau}{h_{i-1}} C_T V (1+K)(T_{i-1} - \theta_{i-1}) \right. \\ &\qquad \left. - \tau\gamma K C_T V \frac{(T_{i-1} - \theta_{i-1})}{\sigma_{i-1}} - \tau\gamma W \right], \end{aligned} \tag{2.26}$$

where $\lambda = (\lambda_1, \ldots, \lambda_n)^t$, $\mu = (\mu_1, \ldots, \mu_n)^t$, $\eta = (\eta_1, \ldots, \eta_n)^t$ are the undetermined Lagrangian multipliers.

As can be seen, the extra terms introduced are each equal to zero, so the value of L is equal to the value of J. However, since the variables are dependent on each other through the constraints, the gradients of L and J are *not* equal. Since the λ's, μ's and η's are arbitrary ($3n$ of them in total), it is possible to consider the change (or variation) of L with respect to each of the $3n+5$ variables. We can set $3n$ of the terms in the variation to zero, by letting the λ's, μ's and η's assume values that will cause an arbitrary set of $3n$ of the terms to vanish. Since we are interested in the 5 control variables, θ_0, h_0, σ_0, K and W , the λ's, μ's and η's are chosen such that the factors of $\frac{\partial L}{\partial \theta_i}$, $\frac{\partial L}{\partial h_i}$ and $\frac{\partial L}{\partial \sigma_i}$ $(1 \leq i \leq n)$ in δL (variation of L) are zero. This will give a set of $3n$ equations in the $3n$ unknowns λ_i , μ_i and η_i. We then use this solution to find expressions for $\frac{\partial L}{\partial \theta_0}$, $\frac{\partial L}{\partial h_0}$, $\frac{\partial L}{\partial \sigma_0}$, $\frac{\partial L}{\partial K}$ and $\frac{\partial L}{\partial W}$ which are generally functions of the λ's, μ's and η's. At the minimum point, these derivative expressions vanish. It is sometimes possible to solve these 5 equations (derivatives set to zero) for c^*. For the more challenging problems, however, it is not possible to find analytic (closed form) solutions. In these cases, an iterative procedure is followed where an initial estimate of c^* is made, and the 5 component gradient of L is found at that point in the control

space. A new estimate is generally made using some form of the descent method. With this as an intuitive backdrop on the methodology, the precise expressions used in the process follow:

$$\begin{aligned}\frac{\partial L}{\partial \theta_0} = \; & \lambda_1 \left[-1+\frac{\tau C_T V(1+K)}{h_0}\right]+\mu_1\left[\frac{\tau C_T V K}{\sigma_0}\right] \\ & + \eta_1 \left[-\frac{\tau C_T V(1+K)}{h_0}+\frac{\tau\gamma C_T V K}{\sigma_0}\right]+2\frac{\theta_0-\tilde{\theta}_0}{\delta_\theta^2},\end{aligned} \tag{2.27}$$

$$\begin{aligned}\frac{\partial L}{\partial h_0} = \; & \lambda_1 \left[\frac{\tau C_T V(1+K)\,(T_0-\theta_0)}{h_0^2}\right]-\mu_1 \\ & - \eta_1 \left[\frac{\tau C_T V(1+K)\,(T_0-\theta_0)}{h_0^2}\right]+2\frac{h_0-\tilde{h}_0}{\delta_h^2},\end{aligned} \tag{2.28}$$

$$\begin{aligned}\frac{\partial L}{\partial \sigma_0} = \; & \mu_1 \left[\frac{\tau C_T V K\,(T_0-\theta_0)}{\sigma_0^2}\right] \\ & + \eta_1 \left[-1+\frac{\gamma\tau C_T V K\,(T_0-\theta_0)}{\sigma_0^2}\right]+2\frac{\sigma_0-\tilde{\sigma}_0}{\delta_\sigma^2},\end{aligned} \tag{2.29}$$

$$\begin{aligned}\frac{\partial L}{\partial K} = \; & \lambda_1 \left[-\frac{\tau C_T V\,(T_0-\theta_0)}{h_0}\right]+\mu_1\left[-\frac{\tau C_T V\,(T_0-\theta_0)}{\sigma_0}\right] \\ & + \eta_1 \left[\frac{\tau C_T V\,(T_0-\theta_0)}{h_0}-\frac{\tau\gamma C_T V\,(T_0-\theta_0)}{\sigma_0}\right] \\ & + \lambda_2 \left[-\frac{\tau C_T V\,(T_1-\theta_1)}{h_1}\right]+\mu_2\left[-\frac{\tau C_T V\,(T_1-\theta_1)}{\sigma_1}\right] \\ & + \eta_2 \left[\frac{\tau C_T V\,(T_1-\theta_1)}{h_1}-\frac{\tau\gamma C_T V\,(T_1-\theta_1)}{\sigma_1}\right] \\ & \vdots \\ & + \lambda_n \left[-\frac{\tau C_T V\,(T_{n-1}-\theta_{n-1})}{h_{n-1}}\right]+\mu_n\left[-\frac{\tau C_T V\,(T_{n-1}-\theta_{n-1})}{\sigma_{n-1}}\right] \\ & + \eta_n \left[\frac{\tau C_T V\,(T_{n-1}-\theta_{n-1})}{h_{n-1}}-\frac{\tau\gamma C_T V\,(T_{n-1}-\theta_{n-1})}{\sigma_{n-1}}\right]\end{aligned} \tag{2.30}$$

and

$$\frac{\partial L}{\partial W} = -\tau\left(\mu_1+\mu_2+\ldots+\mu_n\right)-\tau\gamma\left(\eta_1+\eta_2+\ldots+\eta_n\right). \tag{2.31}$$

Let

$$\nabla L = \left(\frac{\partial L}{\partial \theta_0}, \frac{\partial L}{\partial h_0}, \frac{\partial L}{\partial \sigma_0}, \frac{\partial L}{\partial K}, \frac{\partial L}{\partial W}\right)^t. \tag{2.32}$$

The right hand side in (2.27)-(2.31) involve the undetermined multipliers. Our immediate aim is to compute the values of these multipliers. To this end, we set $\frac{\partial L}{\partial \theta_i} = 0$, $\frac{\partial L}{\partial h_i} = 0$ and $\frac{\partial L}{\partial \sigma_i} = 0$ for all $1 \leq i \leq n$. Thus for $1 \leq j \leq n-1$:

$$\begin{aligned}\frac{\partial L}{\partial \theta_j} = \lambda_j + & \quad \lambda_{j+1} \left[-1 + \frac{\tau C_T V(1+K)}{h_j}\right] + \mu_{j+1}\left[\frac{\tau C_T V K}{\sigma_j}\right] \\ & + \quad \eta_{j+1} \left[-\frac{\tau C_T V(1+K)}{h_j} + \frac{\tau\gamma C_T V K}{\sigma_j}\right] + 2\frac{\theta_j - \tilde{\theta}_j}{\delta_\theta^2} = 0\end{aligned} \tag{2.33}$$

and

$$\frac{\partial L}{\partial \theta_n} = \lambda_n + 2\frac{\theta_n - \tilde{\theta}_n}{\delta_\theta^2} = 0. \tag{2.34}$$

For $1 \leq j \leq n-1$:

$$\begin{aligned}\frac{\partial L}{\partial h_j} = \mu_j + & \quad \lambda_{j+1} \left[\frac{\tau C_T V(1+K)(T_j - \theta_j)}{h_j^2}\right] - \mu_{j+1} \\ & - \quad \eta_{j+1} \left[\frac{\tau C_T V(1+K)(T_j - \theta_j)}{h_j^2}\right] + 2\frac{h_j - \tilde{h_j}}{\delta_h^2} = 0\end{aligned} \tag{2.35}$$

and

$$\frac{\partial L}{\partial h_n} = \mu_n + 2\frac{h_n - \tilde{h_n}}{\delta_h^2} = 0. \tag{2.36}$$

Similary, for $1 \leq j \leq n-1$:

$$\begin{aligned}\frac{\partial L}{\partial \sigma_j} = \eta_j + & \quad \mu_{j+1} \left[\frac{\gamma\tau C_T V K(T_j - \theta_j)}{\sigma_j^2}\right] \\ & + \quad \eta_{j+1} \left[-1 + \frac{\gamma\tau C_T V K(T_j - \theta_j)}{\sigma_j^2}\right] + 2\frac{\sigma_j - \tilde{\sigma_j}}{\delta_\sigma^2} = 0\end{aligned} \tag{2.37}$$

and

$$\frac{\partial L}{\partial \sigma_n} = \eta_n + 2\frac{\sigma_n - \tilde{\sigma_n}}{\delta_\sigma^2} = 0. \tag{2.38}$$

Clearly, (2.33)-(2.38) represent a collection of 3n equations relating the elements of λ, μ and η to the values of θ_i, h_i , σ_i , W and K for $1 \leq i \leq n$. In solving these 3n equations for the 3n unknowns, namely, λ, μ and η, we first rewrite (2.33)-(2.38) to exploit the inherent structure of these equations. To this end, let

$$y_i = (\lambda_i, \mu_i, \eta_i)^t . \tag{2.39}$$

Using (2.39), we can write (2.33)-(2.38) as

$$\begin{aligned} I_3 y_i + C_i y_{i+1} &= d_i, \quad \text{for } 1 \leq i \leq n-1, \\ I_3 y_n &= d_n, \end{aligned} \tag{2.40}$$

where

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.41}$$

$$C_i = \begin{bmatrix} -1 + \frac{\tau C_T V(1+K)}{h_i} & \frac{\tau C_T V K}{\sigma_i} & -\frac{\tau C_T V(1+K)}{h_i} + \frac{\tau C_T V \gamma K}{\sigma_i} \\ \frac{\tau C_T V(1+K)(T_i-\theta_i)}{h_i^2} & -1 & -\frac{\tau C_T V(1+K)(T_i-\theta_i)}{h_i^2} \\ 0 & \frac{\tau K C_T V(T_i-\theta_i)}{\sigma_i^2} & -1 + \frac{\tau K C_T V \gamma (T_i-\theta_i)}{\sigma_i^2} \end{bmatrix} \tag{2.42}$$

and

$$d_i = \begin{bmatrix} -\frac{2(\theta_i - \tilde{\theta}_i)}{\delta_\theta^2} \\ -\frac{2(h_i - \tilde{h}_i)}{\delta_h^2} \\ -\frac{2(\sigma_i - \tilde{\sigma}_i)}{\delta_\sigma^2} \end{bmatrix}. \tag{2.43}$$

The system (2.40) can be succinctly written as

$$Ay = d, \tag{2.44}$$

where A is a upper block bi-diagonal matrix,

$$A = \begin{bmatrix} I_3 & C_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & I_3 & C_2 & 0 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & 0 & I_3 & C_{n-1} \\ 0 & \dots & \dots & \dots & \dots & 0 & 0 & I_3 \end{bmatrix}, \tag{2.45}$$

$$y = (y_1, y_2, \dots, y_n)^t \tag{2.46}$$

and

$$d = (d_1, d_2, \dots, d_n)^t . \tag{2.47}$$

Comparing the matrix A in (2.45) with the matrix E in (2.20), it can be verified that

$$A = -E^t. \tag{2.48}$$

The system (2.44) is solved for y_i , backward, in the decreasing order of the indices, namely from n to 1. Now substituting the values of y_i into (2.27) – (2.31), we arrive at the required gradient. Once the gradient is available, an iterative algorithm for searching the optimum value, $c^* = (\theta_o^*, h_o^*, \sigma_o^*, K^*, W^*)^t$ of J, may be stated as follows. Let

$$c(k) = (\theta_o(k), h_o(k), \sigma_o(k), K(k), W(k))^t \tag{2.49}$$

denote the approximation to c^* after k iterations. The iteration starts at an arbitrary chosen point in the parameter space denoted by $c(0)$.

Step 1. Computation of model output: Let $k = 0$. Given the control vector, $c(k)$, compute the model output, θ_i , h_i and σ_i for $1 \leq i \leq n$, using the equations (2.5)-(2.7).

Step 2. Setting up the adjoint equations: Using the computed values of θ_i , h_i and σ_i evaluate the matrices, C_i, and vectors, d_i for $1 \leq i \leq n$, using (2.42) and (2.43) respectively.

Step 3. Recover the Lagrangian multiplier: Now solve the linear, block, bi-diagonal system (2.44) to recover the set of $3n$ Lagrangian multipliers λ, μ and η.

Step 4. Computation of Gradient: Using the computed values of λ, μ and η, compute the gradient, ∇L, in (2.32) using (2.27)-(2.31).

Step 5. Updating estimates: Compute $c(k+1)$ from $c(k)$ and using one of the many variants of gradient method.

Step 6. Condition for Convergence: Go to step 1 unless $\|c(k) - c(k-1)\|_2 < \epsilon$ for some prespecified ϵ. Then $c^* = c(k)$ and exit.

In the following sections, we use a variation of the Conjugate Gradient method in Step 5. In Step 6, $\|z\|_2$ refers to the two norm which is usual notion of the length of the vector z.

2.6 Analysis of Quality of Estimates

In this section, we look at two ways to analyze the quality of the estimates. The first way is a "twin" experiment conducted to help check the quality of the estimates. Here, we found that we need at least four observations to render our estimates for the control variables meaningful. The second way is to do a Hessian analysis of the structure of the J function near our estimates. From the eigenvalues obtained, we found that the geometry of J is convex in the neighborhood of the estimates, and the contours or level curves of J may be an elongated ellipse in one of the dimensions.

2.6.1 "Twin" Experiment

To render the estimation of the control vector mathematically tractable, in Section 2.4, we have made several simplifying assumptions. Before taking up the estimation of the control vector based on real data, there is a compelling need to develop an appreciation for the impact of these assumptions, in particular, assumption A1. The framework in which such a checking is done has come to be known as the identical twin experiments.

By choosing

$$c^T = (\theta_o^T, h_o^T, \sigma_0^T, K^T, W^T)^t = (10\ C, 200\ m, 1\ C, 0.2, -0.01\ m\ s^{-1})^t, \qquad (2.50)$$

the discrete model in (2.5)-(2.7) was run once (the superscript T denotes the true value). The time evolution of θ_i ,h_i and σ_i for this initial choice of the control vector are shown in Figure 2.2. For purposes of later reference, we call the model solution based on c^T given above as the *true* values of the physical quantities of interest, namely, θ_i^T,h_i^T and σ_i^T, for $1 \leq i \leq n$. It is assumed that our assimilation period is 24 hours, which is divided into $n = 192$ equally spaced intervals, such that $\tau = 450s$. Thus, in Figure 2.2, the value of θ at four hours is, in fact, θ_i for $i = 32$, etc.

A fictitious set of observations are now created as follows. $\tilde{\theta}_i$, the observed value of θ_i , is obtained by adding a mean zero Gaussian random variate, with variance $\delta_\theta^2 = 0.2$, to θ_i^T for $1 \leq i \leq n$. Likewise, $\tilde{h}_i$ and $\tilde{\sigma}_i$, the observed values of h_i and σ_i respectively, are obtained from h_i^T and σ_i^T, by adding mean zero (independent) Gaussian random variates, with variances $\delta_h^2 = 10$ and $\delta_\sigma^2 = 2$ respectively.

Now starting with an arbitrary initial vector $c(0)$, the algorithm in Section 2.5 is applied to compute c^E, an estimate of the optimum value of c^*, (which in this special case) is clearly c^T. The solution of the model equations (2.5)-(2.7) computed from this estimate c^E as the initial conditions, is referred to θ_i^E,h_i^E and σ_i^E, for $1 \leq i \leq n$. To measure the goodness of the estimates as a function of the number of samples, we first introduce some useful notations. Recall that $n\tau$ denote the *assimilation period.* This period is divided into n intervals, each of length τ, given by the set

$$I = \{0, 1, 2, 3, ..., n\} \qquad (2.51)$$

of $(n + 1)$ points. Let $n = rk$ and define

$$I_k = \{0, k, 2k, 3k, ..., rk\} \qquad (2.52)$$

where $I_1 = I$. Let I_k for some $k \geq 1$, denote the set of points at which observations are available. Let $|I_k|$ denote the number of points in I_k. We define a set of three quality indicators for the height variable as follows :

$$Q_{T,E}(I_k) = \frac{1}{|I_k|} \sum_{i \equiv I_k} \left(h_i^T - h_i^E\right)^2, \tag{2.53}$$

$$Q_{T,O}(I_k) = \frac{1}{|I_k|} \sum_{i \equiv I_k} \left(h_i^T - \tilde{h}_i\right)^2 \tag{2.54}$$

and

$$Q_{E,O}(I_k) = \frac{1}{|I_k|} \sum_{i \equiv I_k} \left(h_i^E - \tilde{h}_i\right)^2. \tag{2.55}$$

Since $\tilde{h}_i$ are random, so are h_i^E. Hence all the three quality indicators themselves are random. Now, by keeping c^T fixed, the above computations are repeated N times (say, $N = 100$), by generating N different sets of fictitious observations and the numerical averages, $\overline{Q}_{T,E}(I_k)$, $\overline{Q}_{T,O}(I_k)$ and $\overline{Q}_{E,O}(I_k)$ are computed. From the fact that the statistical properties of the noise remain invariant, it follows that these average values of the above quality indicators depend only on the number, $|I_k| = \frac{n}{k} + 1$, of observations.

Plots of $\overline{Q}_{T,E}(I_k)$, $\overline{Q}_{T,O}(I_k)$ and $\overline{Q}_{E,O}(I_k)$ for various values of k are given in Figure 2.3. In this figure, $n = 144$ and the values are plotted for I_k with $k = 16, 24, 36, 48, 72$ and 144.

Observe that $\overline{Q}_{T,O}(I_k)$ is a measure of the observational error, $\overline{Q}_{T,E}(I_k)$ is a measure of the error in the estimate and $\overline{Q}_{E,O}(I_k)$ is a measure of the model error. Philosophically, unless

$$\overline{Q}_{T,E}(I_k) \leq \overline{Q}_{T,O}(I_k), \tag{2.56}$$

it is pointless to expand energy and time in computing the estimate c^E using any data assimilation scheme. To establish a one-to-one correspondence with the real data experiment, here the assimilation period is 18 hours, which is divided into $n = 144$ equally spaced intervals such that $\tau = 450s$. From Figure 2.3, it is clear that except for the sets I_{144} and I_{72} , the quality estimates satisfies the inequality (2.56) for I_k, for $k \leq 48$. From $|I_k| \geq 4$, for $k \leq 48$, it follows that unless we have four or more observations, in our forecast assimilation problem involving five

parameters, it is pointless to invoke to sophisticated assimilation method to obtain meaningful estimates for the control variables.

Variation of the quality estimates as a function of I_k for other physical variables, namely, θ and σ are given in Figures 2.4 and 2.5 respectively. From Figure 2.4, it can be seen that except for the set I_{144} , the quality estimates satisfies the inequality (2.56) for I_k, for $k \leq 72$. From $|I_k| \geq 3$, for $k \leq 72$, it follows that we need at least three observations in our forecast assimilation problem involving five parameters to obtain meaningful estimates for the control variable θ. From Figure 2.5, the quality estimates satisfies the inequality (2.56) for I_k, for $k \leq 144$.

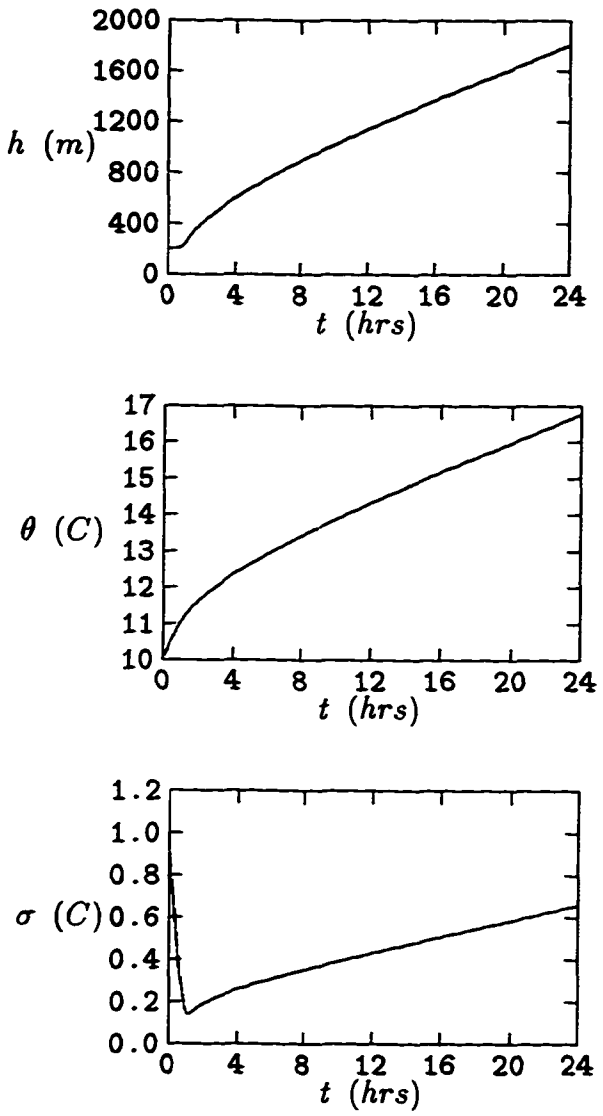
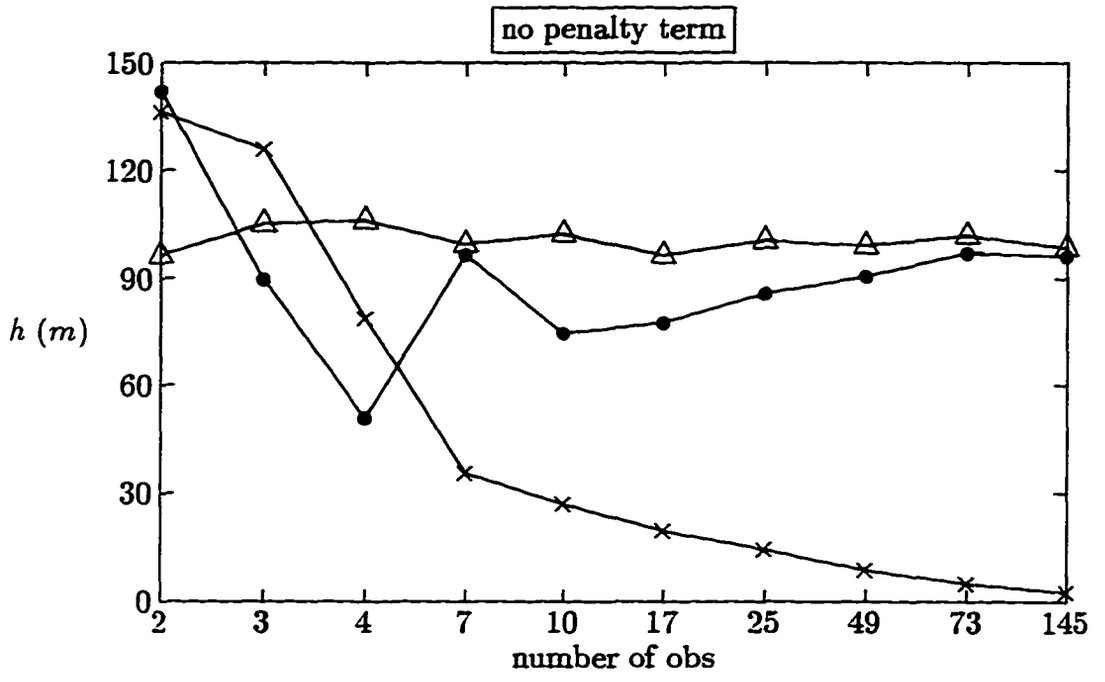


Figure 2.2: Evolution of θ_i, h_i and σ_i for $0 \leq i \leq 192$. $C_T = 0.001$, $V = 10\ m\ s^{-1}$, $K = 0.2$, $\gamma = 0.004$ and $W = -.01\ m\ s^{-1}$. $T(t)$ increases linearly from 15 C to 25 C over 24 hours.



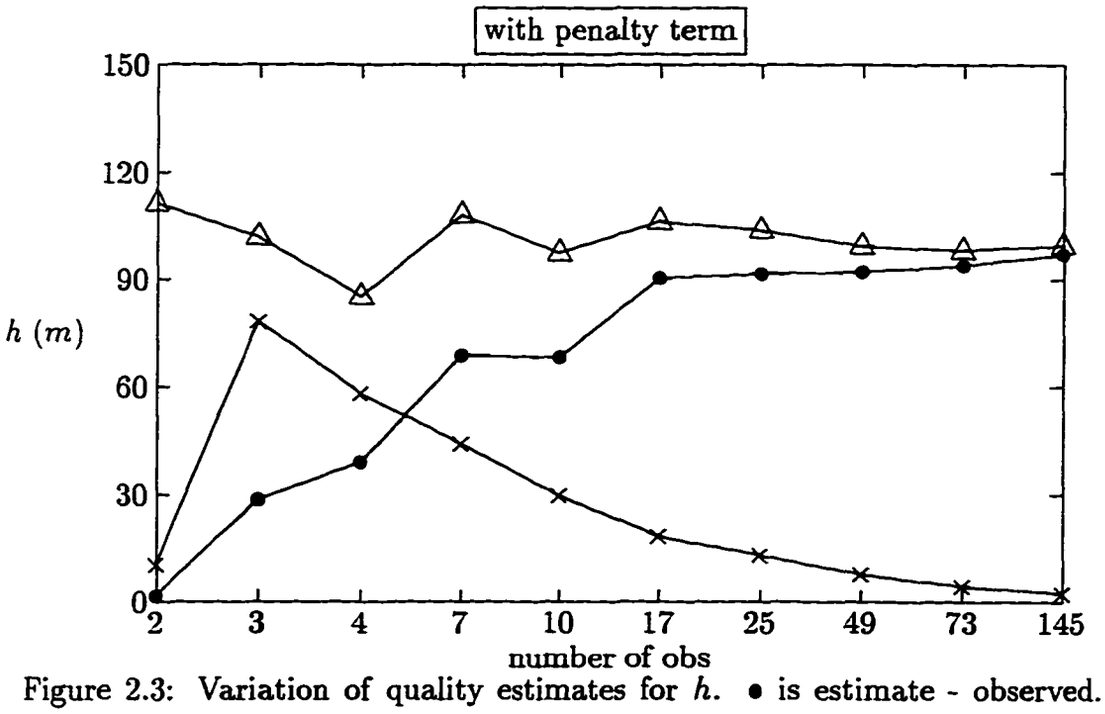
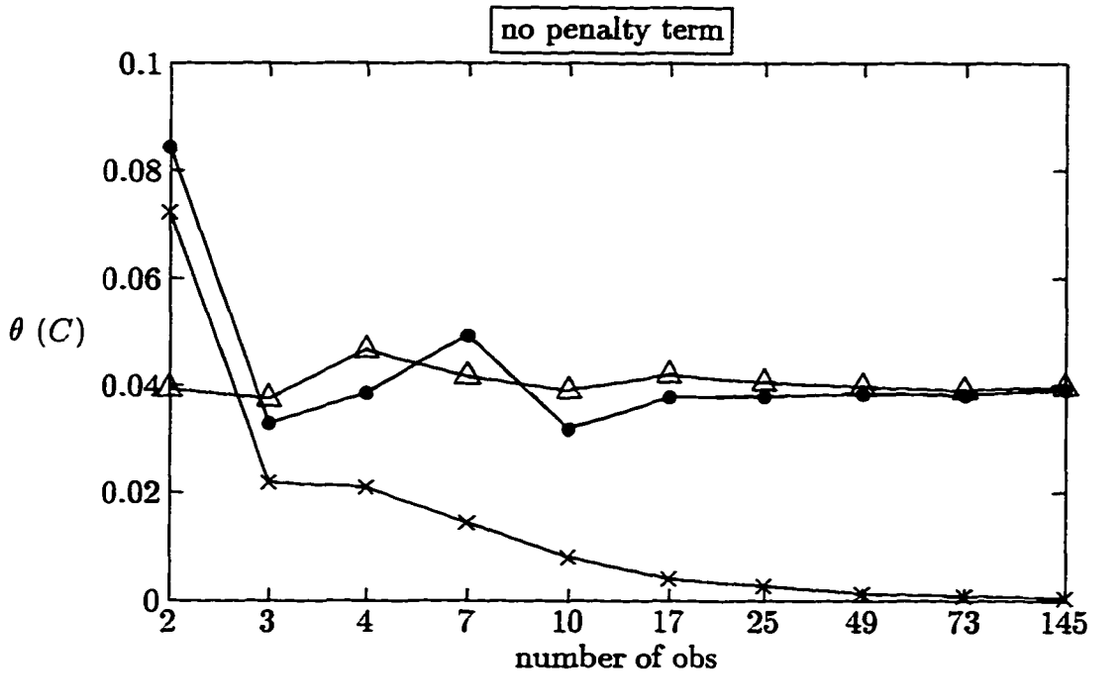


Figure 2.3: Variation of quality estimates for h. • is estimate - observed. × is estimate - truth. △ is observed -truth.



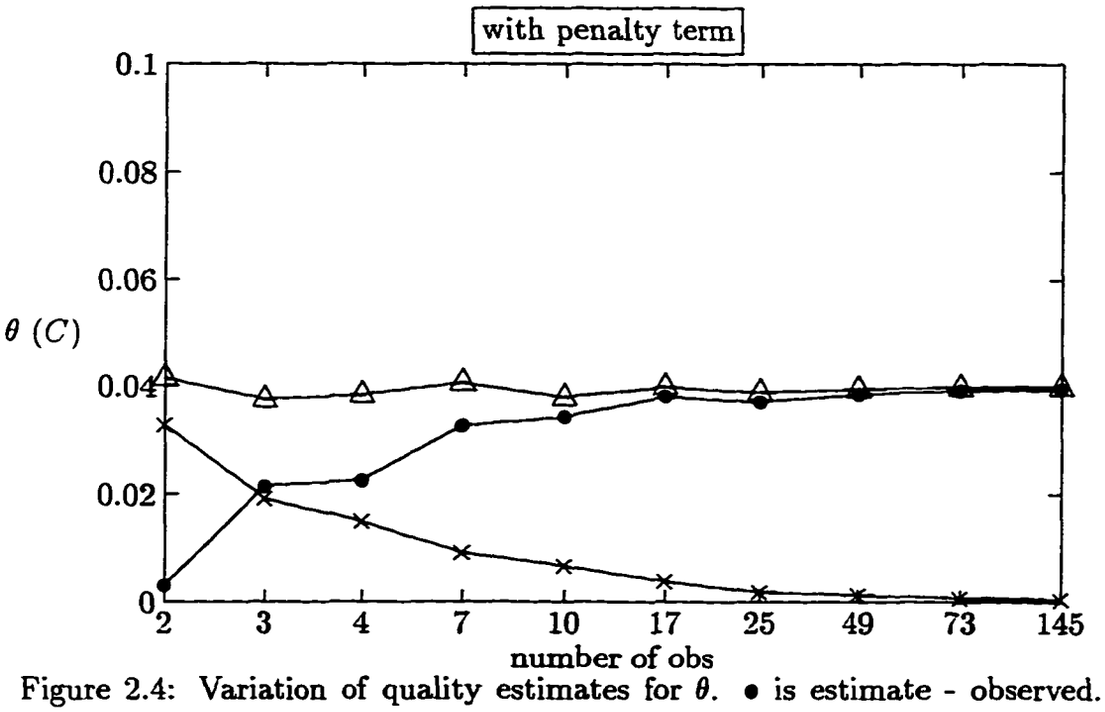
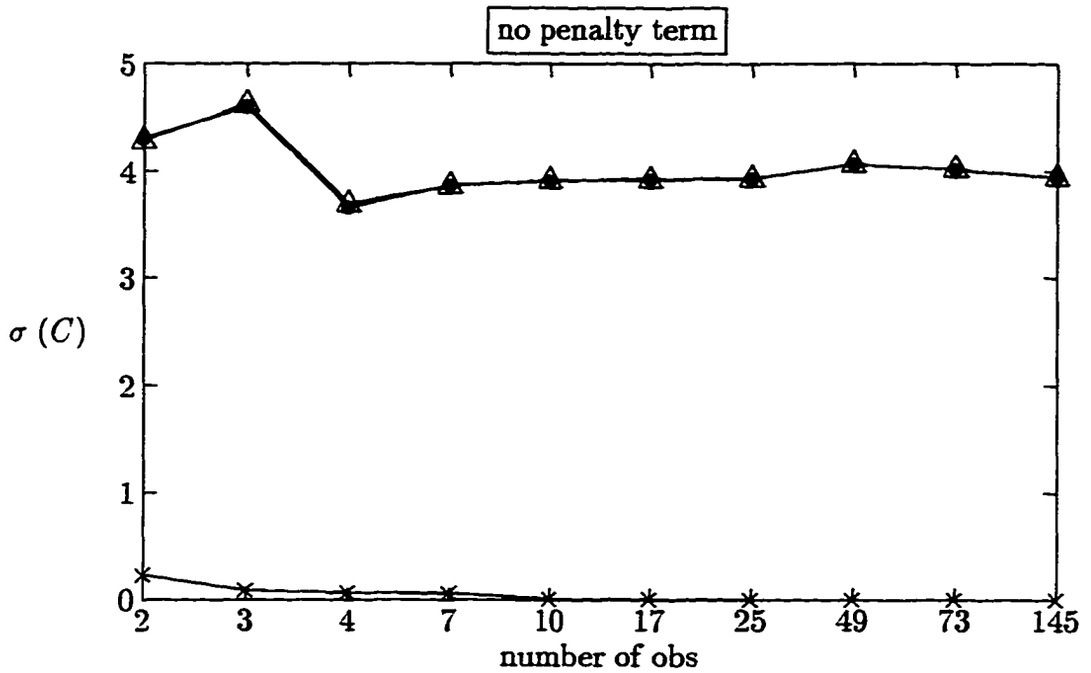


Figure 2.4: Variation of quality estimates for θ. • is estimate - observed. × is estimate - truth. △ is observed -truth.



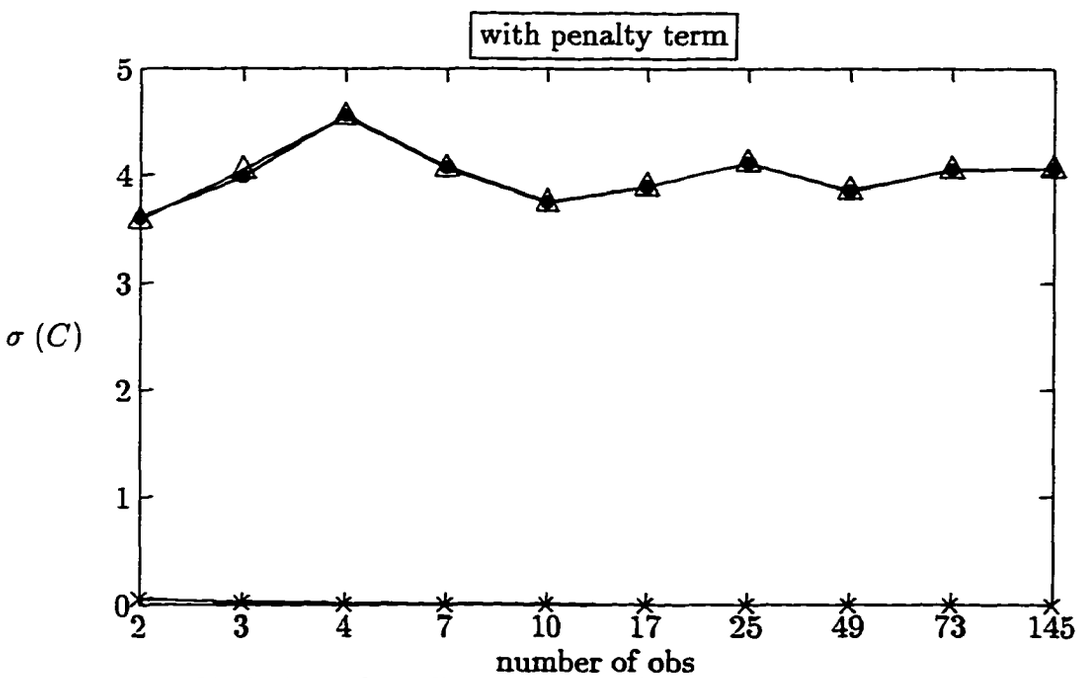


Figure 2.5: Variation of quality estimates for σ. • is estimate - observed. × is estimate - truth. △ is observed -truth.

2.6.2 Hessian Analysis

So far, we have discussed a variational method for fitting numerical models to meteorological observations, known as tha data assimilation problem. The aim is to minimize a cost function (or the functional J), which is quadratic in the differences between the data and their model counterparts, by adjusting independent model variables. (J is dependent on the control variables, i.e., initial conditions, boundary conditions, certain empirical and physical parameters.) The Hessian, when evaluated at the point where the minimum has been estimated, gives us the structure of the J function at that point. Since the curvature is related to the second derivatives of J, we can extract information about the curvature of J, using the Hessian matrix, H of J, which is a symmetric matrix of second partial derivatives.

We have shown that a certain minimum number of observations are necessary to consider data assimilation. Note that the number of observations also directly affect the shape of the J function. When we applied the minimization algorithm to obtain c^E, a question commonly being asked is : does c^E corresponds to a (local) minimum for J ? Note that convergence of the algorithm need not always imply that c^E is close to a true local optimum for J. This is because the iterates may be stuck in a flat plateau.

It is difficult to say which value of the control parameter from a potentially infinite set is appropriate for use in prediction. Thus, it is of interest to determine if c^E is close to a true (local) optimum or if it is a number of a region where J is nearly a constant. This is done by analyzing the Hessian matrix at c^E.

From linear algebra, we know that the function J has a minimum when H is positive-definite (i.e., $c^T Hc > 0$ for all c other than $c_1 = c_2 = \ldots = c_m = 0$). This implies that H has positive eigenvalues. The curvatures of J are closed contours if and only if the eigenvalues of H are all positive. If some of the eigenvalues are very small and positive, the curvatures, while being closed, would be elongated in the direction of the eigenvectors corresponding to the small eigenvalues. Therefore, unless the eigenvalues of H at c^E are all bounded away from zero, it is quite likely that c^E is not a unique local minimum and further analysis must be conducted to examine the terrain around c^E.

%					
5	5.17E+01	8.06E-02	2.97E+02	1.99E+04	3.26E+06
10	5.62E+01	8.06E-02	3.11E+02	2.05E+04	3.27E+06
15	6.23E+01	8.06E-02	3.35E+02	2.16E+04	3.28E+06
20	6.82E+01	8.05E-02	3.70E+02	2.31E+04	3.29E+06
25	4.13E+02	8.03E-02	7.19E+01	2.50E+04	3.31E+06
30	4.64E+02	8.00E-02	7.20E+01	2.74E+04	3.34E+06
35	5.15E+02	7.93E-02	6.64E+01	3.01E+04	3.37E+06
40	5.58E+02	7.73E-02	5.09E+01	3.33E+04	3.40E+06
45	5.72E+02	5.52E-02	1.35E+01	3.70E+04	3.44E+06
50	5.29E+02	9.63E-02	-8.94E+01	4.14E+04	3.48E+06

Table 2.1: Eigenvalues of Hessian Analysis. % is the percentage size of Δc, relative to c^E, used in the construction of the Hessian matrix.

Table 2.1 shows the results of the eigenvalues of the Hessian matrices constructed in the neighborhood of c^E. The analysis of the curvature requires choosing two values of c near c^E. Here we let $c{=}c^E \pm \alpha c^E$, where α ranges from 5 % to 50 %. Each eigenvalue represents the contour or level curve in one dimension. If an eigenvalue is greater than zero, then the contour or level curve in this dimension is convex at c^E; if an eigenvalue is close to zero, then the contour or level curve in this dimension is very flat at c^E and if an eigenvalue is negative, then the contour or level curve in this dimension is not convex.

The geometry of the J function in the vicinity of c^E is convex as shown by the eigenvalues obtained in Table 2.1. Since one column of the eigenvalues is very small, the contours or level curves of J may be an elongated ellipse in the direction of the eigenvectors corresponding to the small eigenvalues. Negative eigenvalue appears as we move far enough away from c^E, thus indicating that the curvature in that dimension is no longer convex.

2.7 Real Data Experiment

The situation over the Gulf of Mexico, where the mixed-layer model is applied, is one where cold continental air streams out over the warmer water of the Gulf. Figure 2.6 shows the path of the air relative to the sea surface over the Gulf of

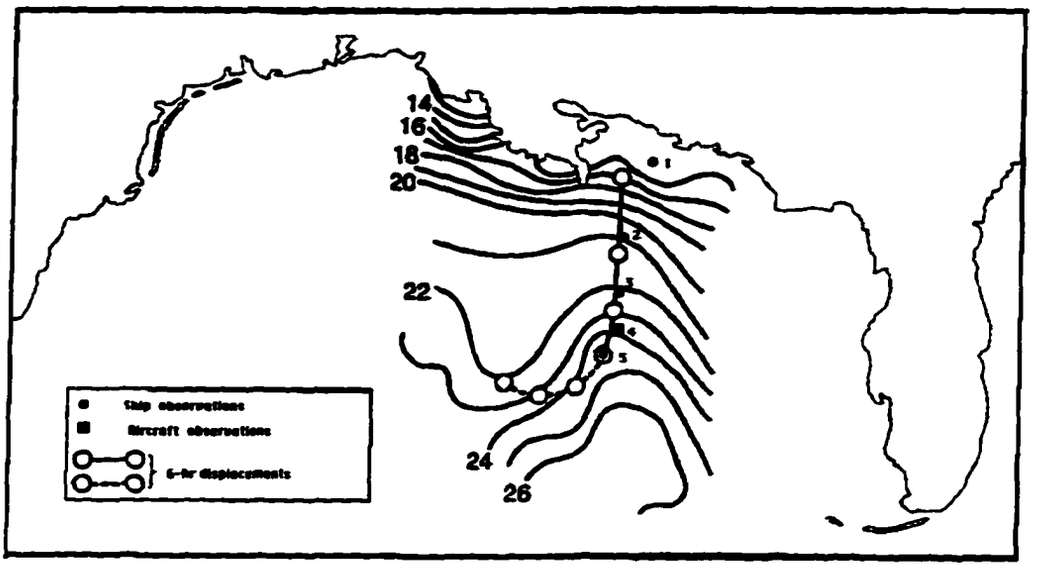


Figure 2.6: Path of air flow over the gulf along with water temperature.

Mexico on Feb. 21-22, 1988. The darkened circles and squares on the figure show the locations of upper-air data collected from a moving ship and an instrumented airplane. Figure 2.7 shows the temperature and height of the mixed-layer at the 5 observation locations, where the labels 1, 2, 3, 4 and 5 refer to the positions of data progressively further from the coastline. Recalling that the identical twin experiment gave good estimates for time interval between observations of 6 hours or less (that is, I_k for $k \leq 48$), the 5 observations collected over 18 hours would appear to be satisfactory for the method. In applying the model to this flow of air over the Gulf of Mexico, it is assumed that the column of air being modified is continually being subjected to a changing lower-level boundary condition, T, the temperature of the water. This should be viewed as a time-dependent problem where the time-dependent boundary condition, is given by the temperature of the water at the times when the air column passes over a certain point on the ocean. (The ship that took the upper air observations, was directed to follow the lower-level winds, and take observations about every 6 hours).

Using the observed $\tilde{\theta}$, $\tilde{h}$ and $\tilde{\sigma}$ at location 1, a forecast over an 18-hour period is shown by the dash curve in Figure 2.8. As can be seen in Figure 2.8, the height forecast of the mixed-layer is systematically too large (by approximately 150 meters). The temperature forecast is systematically too low (by approximately $1C$).

time (hours)	T (C)
0	17.0
1.5	18.8
3.0	19.8
4.5	20.5
6.0	21.0
7.5	21.3
9.0	21.5
10.5	21.8
12.0	22.2
13.5	22.8
15.0	23.5
16.5	24.1
18.0	24.3

Table 2.2: T(t) for the initial 18 hours

time (hours)	T (C)
0	24.3
3	24.1
6	23.5
9	23.0
12	22.5
15	22.0
18	21.3

Table 2.3: T(t) for the forecast period (begins 18 hours after initial time)

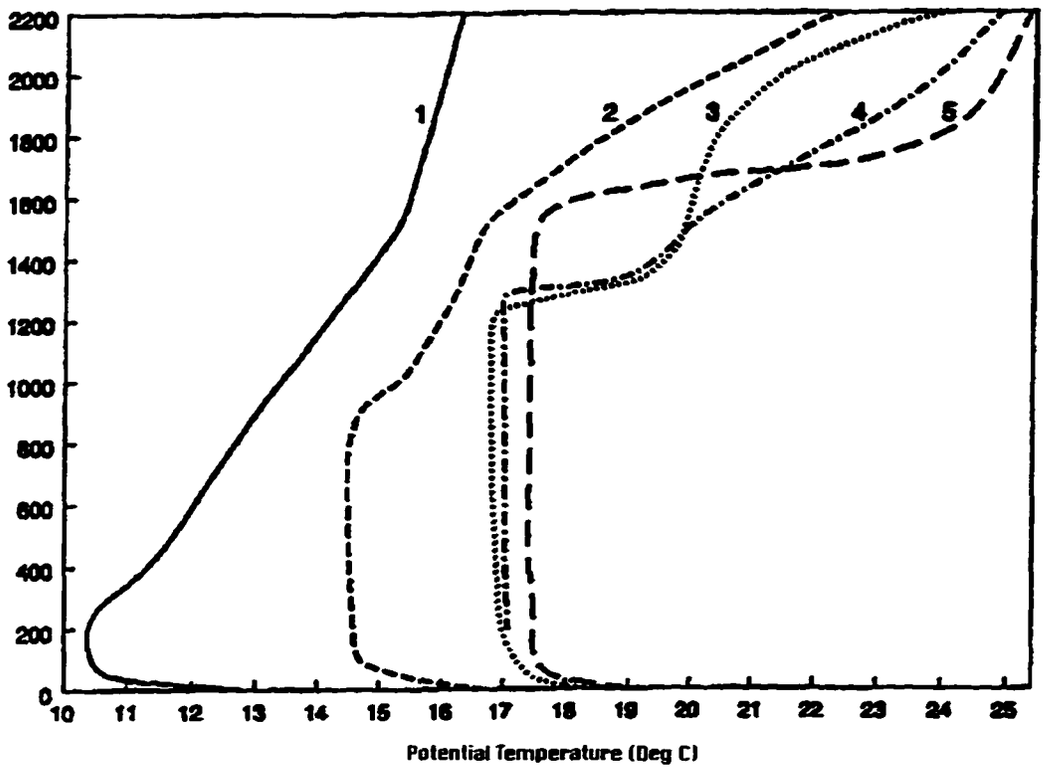


Figure 2.7: Observed sounding of potential temperature at five locations denoted by 1,2,3,4 and 5 in Figure 2.6.

Table 2.2 shows the sea surface temperature, T, encountered by the air parcel as it tracks over the Gulf. Table 2.3 shows the values of the sea surface temperature used for the forecast beyond 18 hours. Figure 2.6 shows the contours of the sea surface temperature on the day the ship and plane observations were taken. These are used to construct Table 2.2. The thermal inertia of the ocean makes it possible to assume that the sea surface temperature is constant at a given location for periods on the order of days. Thus, when we consider the forecasts for a day or less, we assume that the pattern shown in Figure 2.6 is unvarying.

The results of performing the minimization, according to the methodology described in Section 2.5, results in $c^* = (\theta_0^*, h_0^*, \sigma_0^*, K^*, W^*)^t$, where

$$\begin{aligned} \theta_0^* &= 11.4C, \\ h_0^* &= 251.7m, \\ \sigma_0^* &= 1.6C, \\ K^* &= 0.134, \\ W^* &= -0.005ms^{-1}. \end{aligned} \tag{2.57}$$

The forecast values for θ, h and σ, using these optimized values are shown by the solid curve in Figure 2.8. As can be seen, the fit for the variables, θ and h, is much better than when unadjusted observations were used. The values of the two empirical parameters, W and K, also appear to be reasonable and consistent with values obtained by other research (Tennekes 1973), where K was shown to be approximately 0.2. The vertical motion, or variable W, is consistent with the large-scale distribution found over the Gulf on this day (see Figure 2.9).

2.8 Conclusions

We have demonstrated successfully the use of adjoint method in data assimilation for the mixed layer model.

We have also analyzed the dependence of optimal estimates of the control parameters on the quality of data sets. Here, by conducting an identical twin experiment, we found that we need at least four observations to obtain meaningful estimates of the control variables.

We have also examined the effects of the Hessian matrix to access the precision to which these control parameters are determined. From the results of the eigenvalues obtained from the Hessian matrices, we found that the shape of the J function is convex in the neighborhood of our estimates. This shows that we have obtained a local minimum. Another interesting result is the presence of an elongated ellipse in the level curves of the J function as indicated by the big disparity in the eigenvalues.

This clearly explains why scaling of our model variables is so important in order to ensure rapid convergence in our minimization algorithm.

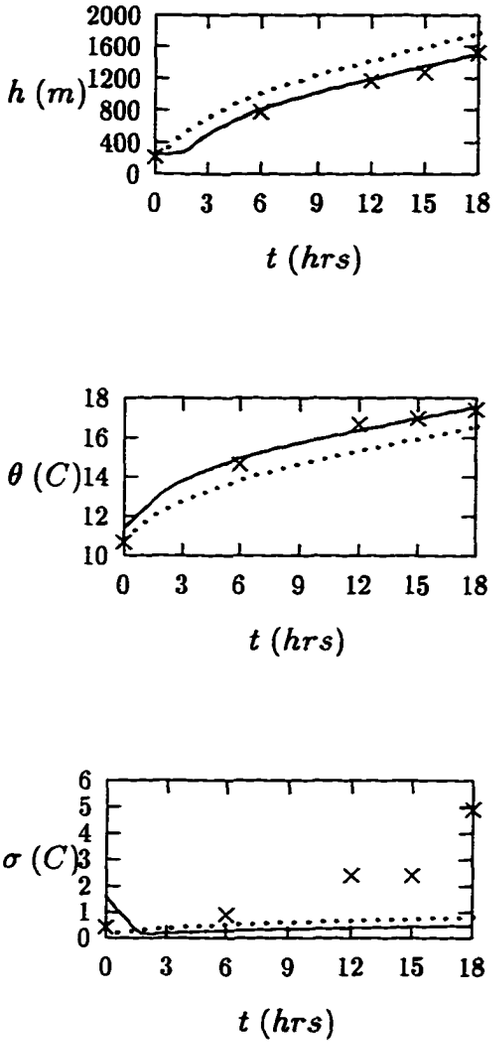


Figure 2.8: An 18 hour forecast for θ, h and σ, based on the optimal estimate for c^* (solid) and based on the initial observations (dash).

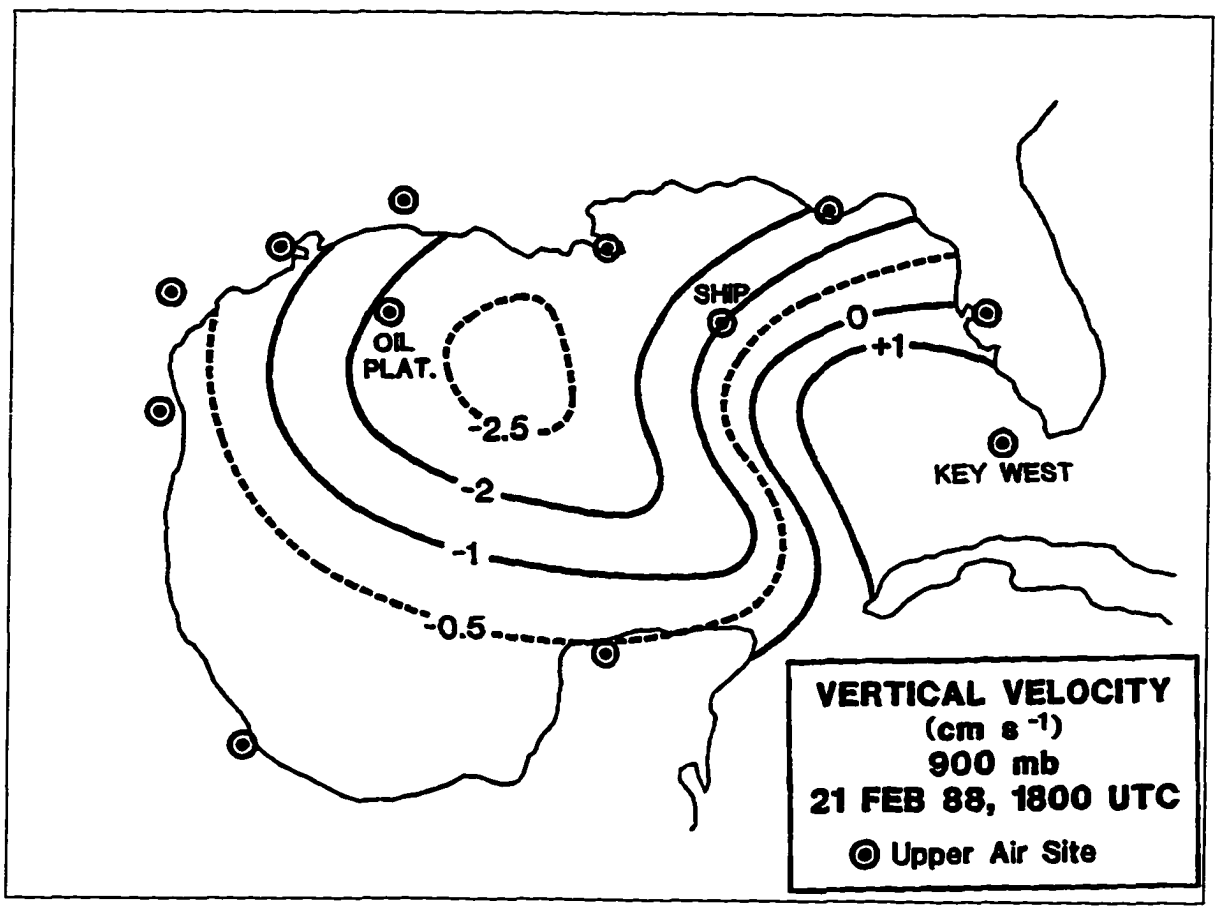


Figure 2.9: The observed values of W.

Chapter 3

Discretization of the Shallow Water Model

3.1 Introduction

In the previous chapter, we have done the data assimilation using the Lagrangian Multiplier Method for the mixed-layer model. In doing so, we have derived a matrix for the tangent linear system (since the model is non-linear) as well as one for the adjoint model. With that as a backdrop, we wanted to employ our experience earned onto a more challenging model, namely, the shallow water model. The shallow water model is widely known in the meteorological literature (Pedlosky 1987).

The shallow water model is linear, but it is both space and time dependent. It also has a larger number of unknown variables/parameters. In this chapter, we wanted to look at the discretization of the shallow water model under two different schemes, namely, the Euler and the leapfrog scheme. We will show that the Euler scheme is an unstable scheme, but the leapfrog scheme is conditionally stable. Under both schemes, we developed the matrix for the forward model and the adjoint model. We also made comparisons of the matrix structure obtained. We will show that a block bi-diagonal system of equations is found for both cases.

3.2 The Shallow Water Model

The shallow water model consists of two coupled linear partial differential equations:

$$\frac{\partial u^*}{\partial t^*} = -g^* \frac{\partial h^*}{\partial x^*}, \tag{3.1}$$

$$b^* \frac{\partial h^*}{\partial t^*} = -\frac{\partial D^* b^* u^*}{\partial x^*}. \tag{3.2}$$

Here the * quantities are dimensional with:

$t^* =$ time (s),

$x^* =$ horizontal distance (m),

$g^* =$ acceleration due to gravity $(m\ s^{-2})$,

$b^*(x^*) =$ breadth of rectangular basin of water (m),

$D^*(x^*) =$ depth of rectangular basin of water (m),

$h^*(x^*, t^*) =$ perturbation about D^* (m).

$u^*(x^*, t^*) =$ horizontal velocity of fluid (ms^{-1}),

$Q^*(x^*, t^*) =$ transport of fluid $(m^3 s^{-1})$.

We will find it convenient to work with a volume flux defined:

$$Q^*(x^*, t^*) \equiv b^*(x^*) D^*(x^*) u^*(x^*, t^*). \tag{3.3}$$

The equations, (3.1) and (3.2), can be found in textbooks (Pedlosky 1987). In shallow water theory, the wavelength of disturbances in h is assumed to be much greater than D. As a result, the pressure gradient beneath the surface is proportional to the slope of the surface, $\frac{\partial h^*}{\partial x^*}$. With the pressure gradient independent of height, the horizontal acceleration of the fluid, $\frac{\partial u^*}{\partial t^*}$, is also independent of height, so the horizontal velocity u^* of the fluid remains independent of height. The equation (3.2) is simply an expression for the conservation of volume of the fluid; the surface will rise where the fluid volume is converging and will fall where the fluid volume is diverging. The rate of flow of volume of fluid, Q^*, is the product of the cross-sectional area, $b^* D^*$, and the fluid velocity, u^*.

We render the model equations dimensionless with using the "typical" or average values [1] for D_0^*, b_0^* and with g^*:

$$t^* = \sqrt{\frac{D_0^*}{g^*}}t, \tag{3.4}$$

$$x^* = D_0^* x, \tag{3.5}$$

$$b^* = b_0^* b, \tag{3.6}$$

$$D^* = D_0^* D, \tag{3.7}$$

$$h^* = \epsilon D_0^* h, \tag{3.8}$$

$$u^* = \epsilon\sqrt{g^* D_0^*}u, \tag{3.9}$$

$$Q^* = \epsilon b_0^* D_0^* \sqrt{g^* D_0^*}Q. \tag{3.10}$$

Here ϵ could be any small number, representing a "typical" wave amplitude. Substitution of (3.3)-(3.10) into (3.1) and (3.2) gives:

$$\frac{\partial Q}{\partial t} = -bD\frac{\partial h}{\partial x}, \tag{3.11}$$

$$\frac{\partial h}{\partial t} = -\frac{1}{b}\frac{\partial Q}{\partial x}. \tag{3.12}$$

Note that neither g^* nor g appears in the dimensionless equations. Furthermore, in layers of constant depth and breadth, we could have $b = 1$ and $D = 1$, which further simplifies the equations and makes the analysis more compact.

A distinguishing feature of this model is that it is both space and time dependent. For purposes of our analysis and data assimilation, the control parameter consists of two initial conditions, $h_i^{(0)}$, for $0 \leq i \leq N-1$ and, $Q_i^{(0)}$, for $1 \leq i \leq N-1$, that is,

$$c = \left(h_0^{(0)}, Q_1^{(0)}, h_1^{(0)}, Q_2^{(0)}, h_2^{(0)}, ..., Q_{N-1}^{(0)}, h_{N-1}^{(0)}\right)^t \tag{3.13}$$

where t denotes the transpose. We will assume that D and b are known, which is a good assumption in practice. Note that the 2 control parameters, h and Q, will both be of the same order, and will both be of order unity if ϵ is chosen appropriately.

[1] Typical values could be, for example, $D_0^* = 100m$ and $b_0^* = 20,000m$.

3.3 Discretization using the Euler scheme

In discretizing the model equations (3.11) and (3.12), we use the general Euler scheme with a staggered grid, that approximates the time derivatives using the forward difference. Let τ denote the time between two successive time intervals, and n be the total number of time steps. Let $h(i, j\tau)$ be denoted $h_i^{(j)}$, and $Q(i, j\tau)$ be denoted $Q_i^{(j)}$. For $1 \leq i \leq N-1$ and $0 \leq j \leq n-1$, the discrete version of (3.11) then corresponds to

$$Q_i = Q_i^{(j)} - \tau b D \left(\frac{h_i^{(j)} - h_{i-1}^{(j)}}{2\Delta x} \right), \tag{3.14}$$

and for $0 \leq i \leq N-1$ and $0 \leq j \leq n-1$, the discrete version of (3.12) corresponds to

$$h_i = h_i^{(j)} - \frac{\tau}{b} \left(\frac{Q_{i+1}^{(j)} - Q_i^{(j)}}{2\Delta x} \right). \tag{3.15}$$

The time interval, $n\tau$, is called the *assimilation period.* The boundary conditions for $0 \leq j \leq n$ are

$$\begin{aligned} Q_0^{(j)} &= 0, \\ Q_N^{(j)} &= 0. \end{aligned} \tag{3.16}$$

Considering only unimodal (i.e. $r = 1$) and bimodal (i.e. $r = 2$) modes only, the initial condition for $0 \leq i \leq N$ is given by

$$Q_i^{(0)} = 0, \tag{3.17}$$

and for $0 \leq i \leq N-1$ is given by

$$h_i^{(0)} = \sum_{r=1}^{2} (h_0)_r \cos \frac{r\pi(2i+1)}{2N}. \tag{3.18}$$

3.3.1 Forward Model

For purposes of later comparison between the structure of the model equations and that of the adjoint equations, we derive the equations for the forward model. Let

$$z^{(i+1)} = \left(h_0^{(i+1)}, Q_1^{(i+1)}, h_1^{(i+1)}, Q_2^{(i+1)}, h_2^{(i+1)}, ..., Q_{N-1}^{(i+1)}, h_{N-1}^{(i+1)}\right)^t. \tag{3.19}$$

We obtain the following linear relation between $z^{(1)}$ and $z^{(0)}$. That is,

$$z^{(1)} = Az^{(0)} + p^{(0)}, \tag{3.20}$$

where the $(2N-1) \times (2N-1)$ matrix A is given by

$$A = \begin{bmatrix} 1 & \frac{-\tau}{2b\Delta x} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{\tau bD}{2\Delta x} & 1 & \frac{-\tau bD}{2\Delta x} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{\tau}{2b\Delta x} & 1 & \frac{-\tau}{2b\Delta x} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{\tau bD}{2\Delta x} & 1 & \frac{-\tau bD}{2\Delta x} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\tau}{2b\Delta x} & 1 & \frac{-\tau}{2b\Delta x} & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \frac{\tau bD}{2\Delta x} & 1 & \frac{-\tau bD}{2\Delta x} \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{\tau}{2b\Delta x} & 1 \end{bmatrix}, \tag{3.21}$$

and the $(2N-1) \times 1$ vector $p^{(0)}$ is given by

$$p^{(0)} = \begin{bmatrix} \frac{\tau Q_0^{(0)}}{2b\Delta x} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ -\frac{\tau Q_N^{(0)}}{2b\Delta x} \end{bmatrix}. \tag{3.22}$$

It can be readily seen that for $0 \leq i \leq n-1$,

$$z^{(i+1)} = Az^{(i)} + p^{(i)}, \tag{3.23}$$

where $z^{(0)}$ and $p^{(0)}$ are given and A is obtained from (3.21).

Note that (3.21) for the forward model is particularly simple for the case where $b = 1$ and $D = 1$. The case where b and D are constants is also of interest because the stability of the forward model can be investigated by analytical means. For that case, Smith (1978) shows that the eigenvalues of A are:

$$\lambda_m = 1 + \sqrt{-\frac{\tau^2 D}{\Delta x^2}} \cos\left(\frac{m\pi}{N+1}\right), \qquad \text{for } 1 \leq m \leq N\ . \tag{3.24}$$

The eigenvalues are complex with $|\lambda_m| > 1$ for all m. Thus the Euler scheme is unstable, whereas the continuous equations are stable. The fact that the Euler scheme is unstable provides the motivation for exploring the stable leapfrog scheme in Section 3.4.

We rewrite (3.23) for $0 \leq i \leq n-1$ as

$$Az^{(i)} - z^{(i+1)} = -p^{(i)}, \tag{3.25}$$

which in turn can be succinctly written in the matrix form as

$$C_E\, z = q, \tag{3.26}$$

where

$$C_E = \begin{bmatrix} -I_{2N-1} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ A & -I_{2N-1} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & A & -I_{2N-1} & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & 0 & A & -I_{2N-1} \end{bmatrix}, \tag{3.27}$$

I_{2N-1} is the identity matrix of size $(2N-1) \times (2N-1)$,

$$z = \left(z^{(1)}, z^{(2)}, \dots\dots\dots\dots, z^{(n)}\right)^t \tag{3.28}$$

and

$$q = \left(q^{(1)}, q^{(2)}, \dots\dots\dots\dots, q^{(n)}\right)^t. \tag{3.29}$$

It can be verified that

$$q^{(1)} = -Az^{(0)} - p^{(0)}, \tag{3.30}$$

and for $i \geq 2$,

$$q^{(i)} = -p^{(i-1)}. \tag{3.31}$$

The matrix C_E in (3.27) is called the lower block bi-diagonal matrix. The importance of this is that the matrix of the adjoint equation is exactly the negative of the transpose of the matrix C_E in (3.27). We will discuss more about this in Section 3.3.3.

3.3.2 A Performance Measure

Recall that our aim in data assimilation is to find the "best" value for the control vector c^* such that the solution of the model equation based on c^* closely "resembles" the observed states of the physical system. To develop a mathematically tractable measure of the closeness of the model solution to the observed data, first we need to make several simplifying assumptions.

A1 It is assumed that the observations on the states of the system are available at every value of the time index.

It is to be realized that physical measurements are often associated with errors called the measurement error. The following assumptions on the nature and type of this class of errors are standard and greatly simplify the analysis.

A2 The errors are random and that the errors in $\tilde{Q}_i^{(j)}$ and $\tilde{Q}_k^{(l)}$ are independent for $i \neq k$, $j \neq l$ and likewise for $\tilde{h}_i^{(j)}$.

A3 The errors in $\tilde{Q}_i^{(j)}$ and $\tilde{h}_i^{(j)}$ are mutually independent.

A4 The random errors are Gaussian with known variance. If this is not the case, then the traditional least squares estimation process should be replaced by the more general maximum likelihood estimation theory that accounts for the non-Gaussian error characteristics. This procedure is more general but also more difficult to handle mathematically.

Thus, it is assumed that $\left(Q_i^{(j)} - \tilde{Q}_i^{(j)}\right)$ is a mean zero Gaussian random variate with the common variance of δ_Q^2. Likewise, let δ_h^2 be the common variances of the mean zero Gaussian variates $\left(h_i^{(j)} - \tilde{h}_i^{(j)}\right)$.

With these assumptions, define $J(c)$ as

$$J(c) = \sum_{i=0}^{N} \sum_{j=0}^{n} \frac{[Q_i^{(j)} - \tilde{Q}_i^{(j)}]^2}{\delta_Q^2} + \sum_{i=0}^{N-1} \sum_{j=0}^{n} \frac{[h_i^{(j)} - \tilde{h}_i^{(j)}]^2}{\delta_h^2}. \tag{3.32}$$

Several comments are in order. First, realize that the model solution $Q_i^{(j)}$ and $h_i^{(j)}$ depend on initial conditions $Q_i^{(0)}$ and $h_i^{(0)}$, that is, on the control vector c.

Using the recurrence relation in Section 3.2 which is the discrete counterpart of the model equation, one can in principle express $Q_i^{(j)}$ in terms of $Q_i^{(0)}$, for all $1 \leq i \leq N-1$, $1 \leq j \leq n$, and $h_i^{(j)}$ in terms of $h_i^{(0)}$, for all $0 \leq i \leq N-1$, $1 \leq j \leq n$. In such a case, we will have J as an explicit function of c with no constraints. Except in trivial and uninteresting cases, such back substitution would be cumbersome and often may not lead to any further insight on the shape of J.

We now restate our data assimilation problem as follows: Given the set of observations, find the value of the control parameter c such that $J(c)$ is a minimum when the model solutions $Q_i^{(j)}$ and $h_i^{(j)}$ depend on c through the model equations (3.14) and (3.15). That is, we are lead to an optimization problem under the presence of dynamic constraints.

3.3.3 The Adjoint Method

The idea is to find the c^* in S such that $J(c^*)$ is the minimum. We propose to find c^* using a variant of the conjugate gradient method, which calls for finding the gradient ∇J under the model constraints. However, since J is *not* known explicitly, this is more easily said than done. In this section, we describe a method which has come to be known as the adjoint method for finding ∇J . To this end, the Lagrangian is introduced (see Lanczos,1970 for the theoretical foundations of minimizing functions subject to constraint).

$$L(c, \lambda, \mu) \;=\; J(c)$$

$$+\sum_{i=0}^{N-1}\sum_{j=1}^{n}\lambda_i^{(j)}\left[h_i^{(j)}-h_i^{(j-1)}+\frac{\tau}{2b\Delta x}\left(Q_{i+1}^{(j-1)}-Q_i^{(j-1)}\right)\right]$$
$$+\sum_{i=1}^{N-1}\sum_{j=1}^{n}\mu_i^{(j)}\left[Q_i^{(j)}-Q_i^{(j-1)}+\frac{\tau bD}{2\Delta x}\left(h_i^{(j-1)}-h_{i-1}^{(j-1)}\right)\right] \tag{3.33}$$

where $\lambda_i^{(j)}$ and $\mu_i^{(j)}$ are the undetermined Lagrangian multipliers.

As can be seen, the extra terms introduced are each equal to zero, so the value of L is equal to the value of J. However, since the variables are dependent on each other through the constraints, the gradients of L and J are *not* equal. Since the λ's and μ's are arbitrary ($(2N-1)n$ of them in total), it is possible to consider the change (or variation) of L with respect to each of the $(2N-1)(n+1)$ variables. We can set $(2N-1)n$ of the terms in the variation to zero by letting the λ's and μ's assume values that will cause an arbitrary set of $(2N-1)n$ of the terms to vanish. Since we are interested in the $(2N-1)$ control variables, $Q_i^{(0)}$ for $1 \le i \le N-1$ and $h_i^{(0)}$ for $0 \le i \le N-1$, the λ's and μ's are chosen such that the factors of $\frac{\partial L}{\partial Q_i^{(j)}}$ $(1 \le i \le N-1, 1 \le j \le n)$ and $\frac{\partial L}{\partial h_i^{(j)}}$ $(0 \le i \le N-1, 1 \le j \le n)$ in δL (variation of L) are zero. This will give a set of $(2N-1)n$ equations in the $(2N-1)n$ unknowns, $\lambda_i^{(j)}$ and $\mu_i^{(j)}$. We then use this solution to find expressions for $\frac{\partial L}{\partial Q_i^{(0)}}$ $(1 \le i \le N-1)$ and $\frac{\partial L}{\partial h_i^{(0)}}$ $(0 \le i \le N-1)$, which are generally functions of the λ's and μ's. At the minimum point, these derivative expressions vanish. It is sometimes possible to solve these $2N-1$ equations (derivatives set to zero) for c^*. For the more challenging problems, however, it is not possible to find analytic (closed form) solutions. In these cases, an iterative procedure is followed where an initial estimate of c^* is made, and the $2N-1$ component gradient of L is found at that point in the control space. A new estimate is generally made using some form of the descent method. With this as an intuitive backdrop on the methodology, the precise expressions used in the process follow:

$$\frac{\partial L}{\partial h_0^{(0)}} = -\lambda_0^{(1)} - \mu_1^{(1)}\frac{\tau bD}{2\Delta x} + 2\frac{h_0^{(0)} - \tilde{h}_0^{(0)}}{\delta_h^2}. \tag{3.34}$$

For $1 \leq i \leq N-2$,

$$\frac{\partial L}{\partial h_i^{(0)}} = -\lambda_i^{(1)} + \mu_i^{(1)} \frac{\tau b D}{2\Delta x} - \mu_{i+1}^{(1)} \frac{\tau b D}{2\Delta x} + 2\frac{h_i^{(0)} - \tilde{h}_i^{(0)}}{\delta_h^2}, \tag{3.35}$$

$$\frac{\partial L}{\partial h_{N-1}^{(0)}} = -\lambda_{N-1}^{(1)} + \mu_{N-1}^{(1)} \frac{\tau b D}{2\Delta x} + 2\frac{h_{N-1}^{(0)} - \tilde{h}_{N-1}^{(0)}}{\delta_h^2}, \tag{3.36}$$

and for $1 \leq i \leq N-1$,

$$\frac{\partial L}{\partial Q_i^{(0)}} = -\mu_i^{(1)} + \lambda_{i-1}^{(1)} \frac{\tau}{2b\Delta x} - \lambda_i^{(1)} \frac{\tau}{2b\Delta x} + 2\frac{Q_i^{(0)} - \tilde{Q}_i^{(0)}}{\delta_Q^2}. \tag{3.37}$$

Let

$$\nabla L = \left[\frac{\partial L}{\partial h_0^{(0)}}, \frac{\partial L}{\partial Q_1^{(0)}}, \frac{\partial L}{\partial h_1^{(0)}}, \cdots, \frac{\partial L}{\partial Q_{N-1}^{(0)}}, \frac{\partial L}{\partial h_{N-1}^{(0)}} \right]^t. \tag{3.38}$$

The right hand side in (3.34)-(3.37) involve the undetermined multipliers. Our immediate aim is to compute the values of these multipliers.

To this end, we set $\frac{\partial L}{\partial Q_i^{(j)}} (1 \leq i \leq N-1, 1 \leq j \leq n)$ and $\frac{\partial L}{\partial h_i^{(j)}} (0 \leq i \leq N-1, 1 \leq j \leq n)$ equal to zero. Thus for $1 \leq i \leq N-1, 1 \leq j \leq n-1$,

$$\frac{\partial L}{\partial Q_i^{(j)}} = \mu_i^{(j)} - \mu_i^{(j+1)} + \lambda_{i-1}^{(j+1)} \frac{\tau}{2b\Delta x} - \lambda_i^{(j+1)} \frac{\tau}{2b\Delta x} + 2\frac{Q_i^{(j)} - \tilde{Q}_i^{(j)}}{\delta_Q^2} = 0 \tag{3.39}$$

and for $1 \leq i \leq N-1$,

$$\frac{\partial L}{\partial Q_i^{(n)}} = \mu_i^{(n)} + 2\frac{Q_i^{(n)} - \tilde{Q}_i^{(n)}}{\delta_Q^2} = 0. \tag{3.40}$$

Similarly, for $1 \leq j \leq n-1$,

$$\frac{\partial L}{\partial h_0^{(j)}} = \lambda_0^{(j)} - \lambda_0^{(j+1)} - \mu_1^{(j+1)} \frac{\tau b D}{2\Delta x} + 2\frac{h_0^{(j)} - \tilde{h}_0^{(j)}}{\delta_h^2} = 0, \tag{3.41}$$

$$\frac{\partial L}{\partial h_{N-1}^{(j)}} = \lambda_{N-1}^{(j)} - \lambda_{N-1}^{(j+1)} + \mu_{N-1}^{(j+1)} \frac{\tau b D}{2\Delta x} + 2\frac{h_{N-1}^{(j)} - \tilde{h}_{N-1}^{(j)}}{\delta_h^2} = 0, \tag{3.42}$$

and for $1 \leq i \leq N-2$, $1 \leq j \leq n-1$,

$$\frac{\partial L}{\partial h_i^{(j)}} = \lambda_i^{(j)} - \lambda_i^{(j+1)} + \mu_i^{(j+1)} \frac{\tau b D}{2\Delta x} - \mu_{i+1}^{(j+1)} \frac{\tau b D}{2\Delta x} + 2\frac{h_i^{(j)} - \tilde{h}_i^{(j)}}{\delta_h^2} = 0 \tag{3.43}$$

and for $0 \leq i \leq N-1$,

$$\frac{\partial L}{\partial h_i^{(n)}} = \lambda_i^{(n)} + 2\frac{h_i^{(n)} - \tilde{h}_i^{(n)}}{\delta_h^2} = 0. \tag{3.44}$$

Clearly, (3.39)-(3.44) represent a collection of $(2N-1)n$ equations relating the elements of λ and μ to the values of $Q_i^{(j)}$, for $1 \leq i \leq N-1$, $1 \leq j \leq n$, and $h_i^{(j)}$, for $0 \leq i \leq N-1$, $1 \leq j \leq n$. In solving these $(2N-1)n$ equations for the $(2N-1)n$ unknowns, namely λ and μ, we first rewrite (3.39)-(3.44) to exploit the inherent structure of these equations. To this end, let

$$y^{(j+1)} = \left(\lambda_0^{(j+1)}, \mu_1^{(j+1)}, \lambda_1^{(j+1)}, \mu_2^{(j+1)}, \ldots, \mu_{N-1}^{(j+1)}, \lambda_{N-1}^{(j+1)}\right)^t. \tag{3.45}$$

Using (3.45), we can write (3.39)-(3.44) for $1 \leq j \leq n-1$ as

$$\begin{aligned} I_{2N-1} y^{(j)} + E_E y^{(j+1)} &= k^{(j)}, \\ I_{2N-1} y^{(n)} &= k^{(n)}, \end{aligned} \tag{3.46}$$

where I_{2N-1} is the identity matrix of size $(2N-1) \times (2N-1)$,

$$E_E = \begin{bmatrix} -1 & \frac{-\tau b D}{2\Delta x} & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ \frac{\tau}{2b\Delta x} & -1 & \frac{-\tau}{2b\Delta x} & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & \frac{\tau b D}{2\Delta x} & -1 & \frac{-\tau b D}{2\Delta x} & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & \frac{\tau}{2b\Delta x} & -1 & \frac{-\tau}{2b\Delta x} & 0 & \ldots & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & \frac{\tau}{2b\Delta x} & -1 & \frac{-\tau}{2b\Delta x} \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & \frac{\tau b D}{2\Delta x} & -1 \end{bmatrix}, \tag{3.47}$$

and

$$k^{(j)} = \begin{bmatrix} -2\frac{h_0^{(j)} - \tilde{h}_0^{(j)}}{\delta_h^2} \\ -2\frac{Q_1^{(j)} - \tilde{Q}_1^{(j)}}{\delta_Q^2} \\ -2\frac{h_1^{(j)} - \tilde{h}_1^{(j)}}{\delta_h^2} \\ \vdots \\ -2\frac{Q_{N-1}^{(j)} - \tilde{Q}_{N-1}^{(j)}}{\delta_Q^2} \\ -2\frac{h_{N-1}^{(j)} - \tilde{h}_{N-1}^{(j)}}{\delta_h^2} \end{bmatrix}. \tag{3.48}$$

The system (3.46) can be succinctly written as

$$F_E\, y = k, \tag{3.49}$$

where F_E is a upper block bi-diagonal matrix,

$$F_E = \begin{bmatrix} I_{2N-1} & E_E & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & I_{2N-1} & E_E & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & 0 & I_{2N-1} & E_E \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & I_{2N-1} \end{bmatrix}, \tag{3.50}$$

$$y = \left[y^{(1)}, y^{(2)}, \dots, y^{(n)}\right]^t \tag{3.51}$$

and

$$k = \left[k^{(1)}, k^{(2)}, \dots, k^{(n)}\right]^t. \tag{3.52}$$

Comparing the matrix F_E in (3.50) with the matrix C_E in (3.27), it can be verified that

$$F_E = -C_E^t. \tag{3.53}$$

The system (3.49) is solved for $y^{(j)}$, backward, in the descending order of the indices, namely from n to 1. Now substituting the values of $y^{(j)}$ into (3.39)-(3.44), we arrive at the required gradient.

3.4 Discretization using the Leapfrog scheme

Let τ, n, $h(i, j\tau)$ and $Q(i, j\tau)$ be defined as in Section 3.3. For $1 \leq i \leq N-1$ and $0 \leq j \leq n-1$, the discrete version of (3.11) then corresponds to

$$Q_i^{(j+1)} = Q_i^{(j)} - \tau bD\left(\frac{h_i^{(j)} - h_{i-1}^{(j)}}{2\Delta x}\right) \qquad \text{for } j = 0 \tag{3.54}$$

and

$$Q_i^{(j+1)} = Q_i^{(j-1)} - \tau bD\left(\frac{h_i^{(j)} - h_{i-1}^{(j)}}{\Delta x}\right) \qquad \text{for } j \geq 1\,. \tag{3.55}$$

For $0 \leq i \leq N-1$ and $0 \leq j \leq n-1$, the discrete version of (3.12) corresponds to

$$h_i^{(j+1)} = h_i^{(j)} - \frac{\tau}{b}\left(\frac{Q_{i+1}^{(j)} - Q_i^{(j)}}{2\Delta x}\right) \qquad \text{for } j = 0 \tag{3.56}$$

and

$$h_i^{(j+1)} = h_i^{(j-1)} - \frac{\tau}{b}\left(\frac{Q_{i+1}^{(j)} - Q_i^{(j)}}{\Delta x}\right) \qquad \text{for } j \geq 1\,. \tag{3.57}$$

The boundary conditions and the initial conditions are given in (3.16) and (3.17)-(3.18) respectively.

3.4.1 Forward Model

As in Section 3.3.1, let $z^{(i+1)}$ be defined by (3.19). We obtain the following linear relation between $z^{(1)}$ and $z^{(0)}$. That is,

$$z^{(1)} = Az^{(0)} + \frac{1}{2}p^{(0)}, \tag{3.58}$$

where the $(2N-1) \times (2N-1)$ matrix A is given by (3.21), and the $(2N-1) \times 1$ vector $p^{(0)}$ is given by (3.22). It can be readily seen that for $1 \leq i \leq n-1$,

$$z^{(i+1)} = Bz^{(i)} + z^{(i-1)} + p^{(i)}. \tag{3.59}$$

The $(2N-1) \times (2N-1)$ matrix B is given by

$$B = \begin{bmatrix} 0 & \frac{-\tau}{b\Delta x} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{\tau b D}{\Delta x} & 0 & \frac{-\tau b D}{\Delta x} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{\tau}{b\Delta x} & 0 & \frac{-\tau}{b\Delta x} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{\tau b D}{\Delta x} & 0 & \frac{-\tau b D}{\Delta x} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\tau}{b\Delta x} & 0 & \frac{-\tau}{b\Delta x} & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \frac{\tau b D}{\Delta x} & 0 & \frac{-\tau b D}{\Delta x} \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{\tau}{b\Delta x} & 0 \end{bmatrix}. \tag{3.60}$$

The stability of the leapfrog scheme for constant b and D is presented in Mesinger & Arakawa (1976). There it is shown that the scheme is stable for

$$\frac{\tau}{\Delta x} \leq 1 \tag{3.61}$$

or

$$\frac{\tau^*}{\Delta x^*} \leq \frac{1}{\sqrt{g^* D_0^*}}. \tag{3.62}$$

Recall that the Euler scheme was always unstable. A stable scheme, like the leapfrog scheme, is certainly preferred in practical applications.

Rewriting (3.58) and (3.59) in matix form, we have

$$C_L \, z = q, \tag{3.63}$$

where

$$C_L = \begin{bmatrix} -I_{2N-1} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ B & -I_{2N-1} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ I_{2N-1} & B & -I_{2N-1} & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & I_{2N-1} & B & -I_{2N-1} & \dots & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & I_{2N-1} & B & -I_{2N-1} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & I_{2N-1} & B & -I_{2N-1} \end{bmatrix}. \tag{3.64}$$

I_{2N-1}, z and q are defined in Section 3.3.1. It can be verified that

$$q^{(1)} = -Az^{(0)} - \frac{1}{2}p^{(0)}, \tag{3.65}$$

$$q^{(2)} = -z^{(0)} - p^{(1)} \tag{3.66}$$

and for $i \geq 3$,

$$q^{(i)} = -p^{(i-1)}. \tag{3.67}$$

The matrix C_L in (3.64) can be partitioned in such a way to form a lower block bi-diagonal matrix. The importance of this is that the matrix of the adjoint equation is exactly the negative of the transpose of the matrix C_L. We will discuss more about this in Section 3.4.3.

3.4.2 A Performance Measure

The underlying assumptions as described in Section 3.3.2 are also applicable here under the leapfrog scheme. The cost function $J(c)$ as given in (3.32) remains unchanged.

3.4.3 The Adjoint Method

The Lagrangian using the leapfrog scheme is defined as

$$\begin{aligned} L(c,\lambda,\mu) &= J(c) \\ &+ \sum_{i=0}^{N-1} \lambda_i^{(1)} \left[h_i^{(1)} - h_i^{(0)} + \frac{\tau}{2b\Delta x} \left(Q_{i+1}^{(0)} - Q_i^{(0)} \right) \right] \\ &+ \sum_{i=0}^{N-1} \sum_{j=2}^{n} \lambda_i^{(j)} \left[h_i^{(j)} - h_i^{(j-2)} + \frac{\tau}{b\Delta x} \left(Q_{i+1}^{(j-1)} - Q_i^{(j-1)} \right) \right] \\ &+ \sum_{i=1}^{N-1} \mu_i^{(1)} \left[Q_i^{(1)} - Q_i^{(0)} + \frac{\tau bD}{2\Delta x} \left(h_i^{(0)} - h_{i-1}^{(0)} \right) \right] \\ &+ \sum_{i=1}^{N-1} \sum_{j=2}^{n} \mu_i^{(j)} \left[Q_i^{(j)} - Q_i^{(j-2)} + \frac{\tau bD}{\Delta x} \left(h_i^{(j-1)} - h_{i-1}^{(j-1)} \right) \right], \end{aligned} \tag{3.68}$$

where $\lambda_i^{(j)}$ and $\mu_i^{(j)}$ are the undetermined Lagrangian multipliers.

The precise expressions used in the process follow:

$$\frac{\partial L}{\partial h_0^{(0)}} = -\lambda_0^{(1)} - \lambda_0^{(2)} - \mu_1^{(1)} \frac{\tau bD}{2\Delta x} + 2\frac{h_0^{(0)} - \tilde{h}_0^{(0)}}{\delta_h^2}. \tag{3.69}$$

For $1 \leq i \leq N-2$,

$$\frac{\partial L}{\partial h_i^{(0)}} = -\lambda_i^{(1)} - \lambda_i^{(2)} + \mu_i^{(1)} \frac{\tau bD}{2\Delta x} - \mu_{i+1}^{(1)} \frac{\tau bD}{2\Delta x} + 2\frac{h_i^{(0)} - \tilde{h}_i^{(0)}}{\delta_h^2}, \tag{3.70}$$

$$\frac{\partial L}{\partial h_{N-1}^{(0)}} = -\lambda_{N-1}^{(1)} - \lambda_{N-1}^{(2)} + \mu_{N-1}^{(1)} \frac{\tau bD}{2\Delta x} + 2\frac{h_{N-1}^{(0)} - \tilde{h}_{N-1}^{(0)}}{\delta_h^2}, \tag{3.71}$$

and for $1 \leq i \leq N-1$,

$$\frac{\partial L}{\partial Q_i^{(0)}} = -\mu_i^{(1)} - \mu_i^{(2)} + \lambda_{i-1}^{(1)} \frac{\tau}{2b\Delta x} - \lambda_i^{(1)} \frac{\tau}{2b\Delta x} + 2\frac{Q_i^{(0)} - \tilde{Q}_i^{(0)}}{\delta_Q^2}. \tag{3.72}$$

Let ∇L be defined as in (3.38). The right hand side in (3.69)-(3.72) involve the undetermined multipliers. Our immediate aim is to compute the values of these

multipliers. To this end, we set $\frac{\partial L}{\partial Q_i^{(j)}}(1 \le i \le N-1, 1 \le j \le n)$ and $\frac{\partial L}{\partial h_i^{(j)}}(0 \le i \le N-1, 1 \le j \le n)$ equal to zero. Thus for $1 \le i \le N-1, 1 \le j \le n-2$,

$$\frac{\partial L}{\partial Q_i^{(j)}} = \mu_i^{(j)} - \mu_i^{(j+2)} + \lambda_{i-1}^{(j+1)} \frac{\tau}{b\Delta x} - \lambda_i^{(j+1)} \frac{\tau}{b\Delta x} + 2\frac{Q_i^{(j)} - \tilde{Q}_i^{(j)}}{\delta_Q^2} = 0, \tag{3.73}$$

and for $1 \le i \le N-1$,

$$\frac{\partial L}{\partial Q_i^{(n-1)}} = \mu_i^{(n-1)} + \lambda_{i-1}^{(n)} \frac{\tau}{b\Delta x} - \lambda_i^{(n)} \frac{\tau}{b\Delta x} + 2\frac{Q_i^{(n-1)} - \tilde{Q}_i^{(n-1)}}{\delta_Q^2} = 0, \tag{3.74}$$

and

$$\frac{\partial L}{\partial Q_i^{(n)}} = \mu_i^{(n)} + 2\frac{Q_i^{(n)} - \tilde{Q}_i^{(n)}}{\delta_Q^2} = 0. \tag{3.75}$$

Similarly for $1 \le j \le n-2$,

$$\frac{\partial L}{\partial h_0^{(j)}} = \lambda_0^{(j)} - \lambda_0^{(j+2)} - \mu_1^{(j+1)} \frac{\tau b D}{\Delta x} + 2\frac{h_0^{(j)} - \tilde{h}_0^{(j)}}{\delta_h^2} = 0, \tag{3.76}$$

and

$$\frac{\partial L}{\partial h_{N-1}^{(j)}} = \lambda_{N-1}^{(j)} - \lambda_{N-1}^{(j+2)} + \mu_{N-1}^{(j+1)} \frac{\tau b D}{\Delta x} + 2\frac{h_{N-1}^{(j)} - \tilde{h}_{N-1}^{(j)}}{\delta_h^2} = 0, \tag{3.77}$$

and for $1 \le i \le N-2$, $1 \le j \le n-2$,

$$\frac{\partial L}{\partial h_i^{(j)}} = \lambda_i^{(j)} - \lambda_i^{(j+2)} + \mu_i^{(j+1)} \frac{\tau b D}{\Delta x} - \mu_{i+1}^{(j+1)} \frac{\tau b D}{\Delta x} + 2\frac{h_i^{(j)} - \tilde{h}_i^{(j)}}{\delta_h^2} = 0. \tag{3.78}$$

$$\frac{\partial L}{\partial h_0^{(n-1)}} = \lambda_0^{(n-1)} - \mu_1^{(n-1)} \frac{\tau b D}{\Delta x} + 2\frac{h_0^{(n-1)} - \tilde{h}_0^{(n-1)}}{\delta_h^2} = 0, \tag{3.79}$$

and

$$\frac{\partial L}{\partial h_{N-1}^{(n-1)}} = \lambda_{N-1}^{(n-1)} + \mu_{N-1}^{(n)} \frac{\tau b D}{\Delta x} + 2\frac{h_{N-1}^{(n-1)} - \tilde{h}_{N-1}^{(n-1)}}{\delta_h^2} = 0. \tag{3.80}$$

For $1 \le i \le N-2$,

$$\frac{\partial L}{\partial h_i^{(n-1)}} = \lambda_i^{(n-1)} + \mu_i^{(n)} \frac{\tau b D}{\Delta x} - \mu_{i+1}^{(n)} \frac{\tau b D}{\Delta x} + 2\frac{h_i^{(n-1)} - \tilde{h}_i^{(n-1)}}{\delta_h^2} = 0. \tag{3.81}$$

For $0 \leq i \leq N-1$,

$$\frac{\partial L}{\partial h_i^{(n)}} = \lambda_i^{(n)} + 2\frac{h_i^{(n)} - \tilde{h}_i^{(n)}}{\delta_h^2} = 0. \tag{3.82}$$

Similarly we can rewrite (3.73)-(3.82) to solve for the (2N-1)n unknowns, namely, λ and μ. Let $y^{(j+1)}$ be defined by (3.45). Using (3.45), we can write (3.73)-(3.82) for $1 \leq j \leq n-2$ as

$$\begin{aligned} I_{2N-1}y^{(j)} - I_{2N-1}y^{(j+2)} + E_L y^{(j+1)} &= k^{(j)}, \\ I_{2N-1}y^{(n-1)} + E_L y^{(n)} &= k^{(n-1)}, \\ I_{2N-1}y^{(n)} &= k^{(n)}, \end{aligned} \tag{3.83}$$

where I_{2N-1} is the identity matrix of size $(2N-1) \times (2N-1)$,

$$E_L = \begin{bmatrix} 0 & -\frac{\tau bD}{\Delta x} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{\tau}{b\Delta x} & 0 & -\frac{\tau}{b\Delta x} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{\tau bD}{\Delta x} & 0 & -\frac{\tau bD}{\Delta x} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{\tau}{b\Delta x} & 0 & -\frac{\tau}{b\Delta x} & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \frac{\tau}{b\Delta x} & 0 & -\frac{\tau}{b\Delta x} \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{\tau bD}{\Delta x} & 0 \end{bmatrix}, \tag{3.84}$$

and $k^{(j)}$ is defined in (3.48). The system (3.83) can be succinctly written as

$$F_L\, y = k, \tag{3.85}$$

where F_L can be partitioned to be an upper block bi-diagonal matrix.

$$F_L = \begin{bmatrix} I_{2N-1} & E_L & -I_{2N-1} & 0 & 0 & \dots & 0 & 0 \\ 0 & I_{2N-1} & E_L & -I_{2N-1} & 0 & \dots & 0 & 0 \\ 0 & 0 & I_{2N-1} & E_L & -I_{2N-1} & \dots & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & \dots & I_{2N-1} & E_L \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & I_{2N-1} \end{bmatrix}, \tag{3.86}$$

y and k are given by (3.51) and (3.52) respectively. Comparing the matrix F_L in (3.86) with the matrix C_L in (3.64), it can be verified that

$$F_L = -C_L^t. \tag{3.87}$$

The system (3.85) is solved for $y^{(j)}$, backward, in the descending order of the indices, namely from n to 1. Now substituting the values of $y^{(j)}$ into (3.73)-(3.82), we arrive at the required gradient.

3.5 Comparison of matrix structures

As illustrated above, we obtained block bi-diagonal matrices from the forward and adjoint models using both the Euler and the leapfrog schemes.

Under the Euler scheme, the discretization of the forward model is given in (3.27). The diagonal blocks are all identity blocks and the sub-diagonal blocks are as depicted in (3.21). It can be seen that the structure of the sub-diagonal blocks are tridiagonal and they are diagonally dominant. The block size is $(2N-1)\times(2N-1)$. The discretization of the adjoint model is given in (3.50). The diagonal blocks are also identity blocks and the sub-diagonal blocks are given in (3.47). Similarly, the structure of the sub-diagonal blocks are tridiagonal and they are diagonally dominant. The block size remains unchanged.

Under the leapfrog scheme, the discretization of the forward model is given in (3.64) and the adjoint model is given in (3.86). A block bi-diagonal matrix structure can be obtained by re-blocking or partitioning. The size of the block structure will be $(4N-2)\times(4N-2)$, twice that of the Euler scheme. The diagonal blocks obtained are lower(upper) triangular block matrices for the forward(adjoint) model respectively. The sub-diagonal blocks obtained are upper(lower) triangular block matrices for the forward (adjoint) model respectively.

To obtain a matrix structure similar to that of the Euler scheme, a simple transformation can be done. For the matrix C_L given in (3.64), let

$$X = \begin{bmatrix} -I_{2N-1} & 0 \\ B & -I_{2N-1} \end{bmatrix}, \tag{3.88}$$

$$Y = \begin{bmatrix} I_{2N-1} & B \\ 0 & I_{2N-1} \end{bmatrix}, \tag{3.89}$$

where X is the $(4N-2) \times (4N-2)$ (new) diagonal block, and Y is the $(4N-2) \times (4N-2)$ (new) sub-diagonal block.

We can rewrite (3.64) as follows:

$$C_L = \begin{bmatrix} X & 0 & 0 & 0 & \dots \\ Y & X & 0 & 0 & \dots \\ 0 & Y & X & 0 & \dots \\ 0 & 0 & Y & X & \dots \\ \dots & & & & \end{bmatrix}. \tag{3.90}$$

To convert all the diagonal blocks, X, in (3.90), to identity blocks, simply pre-multiply C_L by a matrix with all elements equal to zero, except for the diagonal blocks, which are the inverse of the diagonal blocks of the matrix to be converted, namely X^{-1}. We call this matrix D where

$$D = \begin{bmatrix} X^{-1} & 0 & 0 & 0 & \dots \\ 0 & X^{-1} & 0 & 0 & \dots \\ 0 & 0 & X^{-1} & 0 & \dots \\ 0 & 0 & 0 & X^{-1} & \dots \\ \dots & & & & \end{bmatrix}. \tag{3.91}$$

Pre-multiplying C_L by D, we obtain the desired result where

$$DC_L = \begin{bmatrix} I_{4N-2} & 0 & 0 & 0 & \dots \\ X^{-1}Y & I_{4N-2} & 0 & 0 & \dots \\ 0 & X^{-1}Y & I_{4N-2} & 0 & \dots \\ 0 & 0 & X^{-1}Y & I_{4N-2} & \dots \\ \dots & & & & \end{bmatrix}. \tag{3.92}$$

It is noted that the computation of X^{-1} can be easily obtained because of the inherent structure of the matrix X itself. The inverse of X is given by

$$X^{-1} = \begin{bmatrix} -I_{2N-1} & 0 \\ -B & -I_{2N-1} \end{bmatrix}. \tag{3.93}$$

The (new) sub-diagonal block, $X^{-1}Y$, in (3.92) can be easily computed as follows:

$$X^{-1}Y = \begin{bmatrix} -I_{2N-1} & -B \\ -B & -B^2 - I_{2N-1} \end{bmatrix}. \tag{3.94}$$

Let

$$\alpha \equiv \frac{\tau}{\Delta x}. \tag{3.95}$$

An example of the structure of the (new) sub-diagonal block, $X^{-1}Y$, is shown below:

$$X^{-1}Y = \begin{bmatrix}
-1 & 0 & 0 & 0 & \dots & 0 & \frac{\alpha}{b} & 0 & 0 & \dots \\
0 & -1 & 0 & 0 & \dots & -\alpha bD & 0 & \alpha bD & 0 & \dots \\
0 & 0 & -1 & 0 & \dots & 0 & -\frac{\alpha}{b} & 0 & \frac{\alpha}{b} & \dots \\
0 & 0 & 0 & -1 & \dots & 0 & 0 & -\alpha bD & 0 & \dots \\
\vdots & & & & & & & & & \\
0 & \frac{\alpha}{b} & 0 & 0 & \dots & \alpha^2 D - 1 & 0 & -\alpha^2 D & 0 & \dots \\
-\alpha bD & 0 & \alpha bD & 0 & \dots & 0 & 2\alpha^2 D - 1 & 0 & -\alpha^2 D & \dots \\
0 & -\frac{\alpha}{b} & 0 & \frac{\alpha}{b} & \dots & -\alpha^2 D & 0 & 2\alpha^2 D - 1 & 0 & \dots \\
0 & 0 & -\alpha bD & 0 & \dots & 0 & -\alpha^2 D & 0 & 2\alpha^2 D - 1 & \dots \\
\vdots & & & & & & & & &
\end{bmatrix}. \tag{3.96}$$

Clearly, $X^{-1}Y$ is dense and diagonally dominant for small α.

The only extra cost incurred is in computing B^2, where B is defined in (3.60) for the foward model.

A similar transformation on the matrix for the adjoint model given in (3.86) can be done, except E_L^2 has to be computed where E_L is defined in (3.84).

The (new) sub-diagonal blocks are dense and diagonally dominant in both cases.

3.6 Conclusions

In this chapter, we discretized the shallow water model equations using two different schemes. Under the Euler scheme, we obtained a block bi-diagonal matrix structure with the diagonal blocks being identitiy blocks. The sub-diagonal blocks are tridiagonal matrices which are diagonally dominant. The Euler scheme is shown to be unstable because the modulus of some of the eigenvalues of the forward model are greater than one.

Under the stable leapfrog scheme, we also obtained a block bi-diagonal matrix structure after re-blocking or partitioning the matrix. To obtain a similar form of the block bi-diagonal matrix structure as in the Euler scheme, we need to transform the matrix. By doing so, we increased the block size of the sub-diagonal blocks and modified the structure so that it is now dense. The block size under the leapfrog scheme is twice that under the Euler scheme.

In the following chapter, we will look at four vector/parallel algorithms which belong to a class of direct parallel methods for solving the block bi-diagonal system of equations.

Chapter 4

Algorithms for Solving Block Bi-Diagonal Systems

4.1 Introduction

In the previous chapter, we discretized the shallow-water model using both the Euler and leapfrog schemes. Both models give rise to a block bi-diagonal system. The diagonal blocks are identity blocks and the sub-diagonal blocks are either tridiagonal or dense in structure. In general, such systems can be solved by the application of either direct or iterative methods, and, done either in parallel or in serial. A general discussion of iterative parallel and direct parallel methods can by found in Ortega (1988).

Here we apply four direct, vector/parallel algorithms to the block bi-diagonal system. These direct parallel methods exploit the block bi-diagonal structure of the matrices. Algorithm # 1 is also known as the "divide and conquer" method (Lakshmivarahan & Dhall 1990). Algorithm # 2 is a variation of the divide and conquer approach (Conn & Podrazik 1994, Van der Vorst 1988, Meyer & Podrazik 1987, Meyer & Podrazik 1989). In Algorithms #1 and #2, the original problem is partitioned into a sequence of reduced linear first-order recurrence subproblems. The difference between them lies in the order the sequence of reduced recurrence subproblems are computed. Algorithm # 3 is the partition algorithm (Van der Vorst & Dekker 1989). It exploits the structure of the matrix by utilizing partitions, *i.e.*, by re-blocking the original problem. Algorithm # 4 is commonly known as the cyclic reduction method. This method permits only limited parallelism and is preferred for a limited number of processors which is not dependent on the size of the problem (Lakshmivarahan & Dhall 1990).

The four algorithms are detailed in Section 4.2. Section 4.3 contains comparisons of the CPU time and wall clock time for the various algorithms.

4.2 Vector/Parallel Algorithms

Consider the following block first-order bi-diagonal problem with block size m. Given the scalars N and $m \geq 2$, given the Nm—vector $d = (d_1, d_2, \ldots, d_N)$ and given the sequence of $m \times m$ matrices $A = (A_2, \ldots, A_N)$, compute the Nm—vector $x = (x_1, x_2, \ldots, x_N)$, such that

$$x_1 = d_1,$$

$$x_i = A_i x_{i-1} + d_i \qquad (2 \leq i \leq N), \tag{4.1}$$

where x_i and d_i are m—vectors, for $i = 1, 2, \ldots, N$. (When $m = 1$, the problem becomes a scalar first-order linear recurrence system.) This problem is written in matrix notation as

$$Bx = d, \tag{4.2}$$

where B is a lower block bi-diagonal matrix.

Algorithm (4.1) will be referred to as the serial algorithm. Note that the serial algorithm takes $T(m, N) = 2m^2(N-1)$ steps to complete for dense sub-diagonal blocks, A_i $(2 \leq i \leq N)$. If the sub-diagonal blocks, A_i $(2 \leq i \leq N)$, are tridiagonal, then the above serial algorithm will take $T(m, N) = 6m(N-1)$ steps to complete.

4.2.1 Algorithm # 1

This algorithm for solving the above block bi-diagonal problem is known as the "divide and conquer" method (Lakshmivarahan & Dhall 1990). It is made up of two parts: (a) a parallel Algorithm A for computing the matrix-matrix multiplications as shown and (b) a parallel Algorithm x for computing matrix-vector multiplications. The complete algorithm is illustrated in Figure 4.1. Because both algorithms work in parallel, the total number of processors is the sum of the processors used in the two algorithms, but the time taken is the maximum of the time needed by both algorithms. When computing for the time taken by the algorithm, it should be noted that we assume a multiplication and an addition take a unit of time each.

```
For i = 1 to N DO IN PARALLEL x_i = d_i;
For j = 2 to N step w DO IN PARALLEL
    For i = j to j + w - 2 DO
        x_i = A_i x_{i-1} + x_i;
        A_i = A_i A_{i-1};
    EndFor
EndFor
For k = 2 to log_2 p DO
    For j = 2^{(k-1)} w + 1 to N step 2v DO IN PARALLEL
        For i = j to j + v - 1 DO
            x_i = A_i x_{i-1} + x_i;
            A_i = A_i A_{i-1};
        EndFor
    EndFor
    v = 2v
EndFor
For i = N/2 + 1 to N DO
    x_i = A_i x_{i-1} + x_i;
EndFor
```

Figure 4.1: Algorithm # 1

Algorithm x uses p processors $P_0, P_1 \ldots P_{p-1}$, and Algorithm A uses q processors $Q_0, Q_1 \ldots Q_{q-1}$, where $q = \frac{p}{2}$. It should be noted that the values of the A_i function are made available at least one step ahead of the time they are needed in Algorithm x.

Let $N, p, w, v, A = (A_2, \ldots, A_N)$ and $d = (d_1, d_2, \ldots, d_N)$ be given, where $N = pw$ and $w = v$.

To compute the complexity of this algorithm, we note that Algorithm A involves matrix-matrix multiplications which take a lot longer than matrix-vector computations involved in Algorithm x. So the total time taken is decided by the running time of Algorithm A. We also note that since the last step of stage 3 in Algorithm x does not take place until the values of the appropriate A's are determined, we have to take that time needed into consideration when computing the total time taken.

In stages 1 and 2 of Algorithm A, since each update involves a matrix-matrix multiplication, these stages take $\frac{N}{p}(2m^3 - m^2)$ units of time each.

In stage 3, each of the steps can be completed in no more than $\frac{N}{p}(2m^3 - m^2)$ units of time. This stage takes $\frac{N}{p}(2m^3 - m^2)(\log p - 1)$ units of time.

Thus, the entire Algorithm A takes no more than $\frac{N}{p}(2m^3 - m^2)(\log p + 1)$ units of time, and the total number of operations is given by $N(2m^3 - m^2)(\log p + 1)$.

In stages 1 and 2 of Algorithm x, since each update involves a matrix-vector multiplication and a vector addition, these stages take $\frac{N}{p}(2m^2 - m)$ units of time each. The total number of operations required is $2N(2m^2 - m)$.

In stage 3, each group in step s takes the same amount of time equal to $\frac{N}{p}(2m^2 - m)$ units of time, and the number of operations is $N(2m^2 - m)$.

Therefore, the divide-and-conquer algorithm requires a total of

$$2Nm^3(\log p + 1) - Nm^2(\log p - 5) - 3Nm$$

scalar operations and takes

$$T_{\frac{3}{2}p}(m, N, p) = \frac{N}{p}2m^3(\log p + 1) + \frac{N}{p}m^2(1 - \log p) - \frac{N}{p}m \tag{4.3}$$

units of time using a total of $\frac{3}{2}p$ processors. Algorithm # 1 also involves concurrent reads which requires extra communication cost. This is illustrated in the example on Table 4.1. Computations of $x_{1:5}$, $x_{1:6}$, $x_{1:7}$ and $x_{1:8}$ requires concurrent reads of $x_{1:4}$ and computations of $x_{9:13}$, $x_{9:14}$, $x_{9:15}$ and $x_{9:16}$ requires concurrent reads of $x_{9:12}$, etc. The overhead in this algorithm lies mainly in the matrix-matrix multiplications that are required in all stages except for the last step in the final stage.

Note that we are using the following notation. For $i \leq j$,

$$\begin{aligned} x_{i:j} \;=\; & A_j A_{j-1} \cdots A_{j-(j-i-1)} d_i \\ & + A_j A_{j-1} \cdots A_{j-(j-i-2)} d_{i+1} \\ & \cdots \\ & + A_j d_{j-1} \\ & + d_j \end{aligned}$$

and

$$A_{i:j} = A_i A_{i+1} \cdots A_{i+(j-i-1)} A_j. \tag{4.4}$$

Stage 1 & 2	Stage 3 (s=1)	Stage 3 (s=2)
x_1		
$x_{1:2}$		
$x_{1:3}$		
$x_{1:4}$		
x_5	$x_{1:5}$	
$x_{5:6}$, $A_{5:6}$	$x_{1:6}$	
$x_{5:7}$, $A_{5:7}$	$x_{1:7}$	
$x_{5:8}$, $A_{5:8}$	$x_{1:8}$	
x_9		$x_{1:9}$
$x_{9:10}$, $A_{9:10}$		$x_{1:10}$
$x_{9:11}$, $A_{9:11}$		$x_{1:11}$
$x_{9:12}$, $A_{9:12}$		$x_{1:12}$
x_{13}	$x_{9:13}$, $A_{9:13}$	$x_{1:13}$
$x_{13:14}$, $A_{13:14}$	$x_{9:14}$, $A_{9:14}$	$x_{1:14}$
$x_{13:15}$, $A_{13:15}$	$x_{9:15}$, $A_{9:15}$	$x_{1:15}$
$x_{13:16}$, $A_{13:16}$	$x_{9:16}$, $A_{9:16}$	$x_{1:16}$

Table 4.1: Algorithm # 1 example

4.2.2 Algorithm # 2

Algorithm # 2 is a variation of the "divide and conquer" method (Conn & Podrazik 1994, Meyer & Podrazik 1989, Van der Vorst 1988, Meyer & Podrazik 1987). It is similar to Algorithm # 1 for the first two stages. In stage 3, if the x's are computed serially, then no matrix-matrix multiplications are necessary. But as we have already observed in Algorithm # 1, if we employed a "divide and conquer" strategy to compute the x's, a high overhead is incurred in doing the matrix-matrix multiplications, and is thus not very cost-effective. In stage 4, computations of the x's can be done in parallel. Algorithm #2 is illustrated in Figure 4.2. The total number of processors used here is the same as in Algorithm # 1 ; with Algorithm x using p processors, and Algorithm A using $q = \frac{p}{2}$ processors. The time taken is the

```
For i = 1 to N DO IN PARALLEL x_i = d_i;
For j = 2 to N step w DO IN PARALLEL
    For i = j to j + w - 2 DO
        x_i = A_i x_{i-1} + x_i;
        A_i = A_i A_{i-1};
    EndFor
EndFor
For i = 2w to N step w
    x_i = A_i x_{i-1} + x_i;
EndFor
For j = w + 1 to N step w DO IN PARALLEL
    For i = j to j + w - 2 DO
        x_i = A_i x_{i-1} + x_i;
    EndFor
EndFor
```

Figure 4.2: Algorithm # 2

sum of the time taken by Algorithm A in stages 1 and 2 and that of Algorithm x for stages 3 and 4.

Let $N, p, w, A = (A_2, \ldots, A_N)$ and $d = (d_1, d_2, \ldots, d_N)$ be given, where $N = pw$.

To compute the complexity of this algorithm, we first look at stages 1 and 2 of Algorithm A. These stages take $\frac{N}{p}(2m^3 - m^2)$ units of time each and $2N(2m^3 - m^2)$ number of operations. Since stage 3 of Algorithm x is computed in serial with only matrix-vector multiplications, the entire stage takes $(p-1)(2m^2 - m)$ units of time. Again, only matrix-vector multiplications are done in stage 4 of Algorithm x, the time taken is no more than $\frac{N}{p^2}(2m^2 - m)$ units. The total number of operations needed to compute Algorithm x is $2N(2m^2 - m) + (p-1)(2m^2 - m) + \frac{N}{p}(2m^2 - m)$.

Therefore, the variation of the divide-and-conquer algorithm requires a total of

$$4Nm^3 + 2Nm^2(1 + \frac{1}{p}) - Nm(2 + \frac{1}{p}) + m(p-1)(2m-1)$$

scalar operations and takes

$$T_{\frac{3}{2}p}(m, N, p) = \frac{2N}{p}(2m^3 - m^2) + (2m^2 - m)(\frac{N}{p^2} + p - 1) \qquad (4.5)$$

units of time using a total of $\frac{3}{2}p$ processors. An example of this algorithm is given in Table 4.2. Concurrent reads are also needed in this algorithm. The main overhead again lies in the matrix-matrix multiplications in stages 1 and 2.

Stage 1 & 2	Stage 3 step 1	Stage 3 step2	Stage 3 step3	Stage 4
x_1				
$x_{1:2}$				
$x_{1:3}$				
$x_{1:4}$				
x_5				$x_{1:5}$
$x_{5:6}$, $A_{5:6}$				$x_{1:6}$
$x_{5:7}$, $A_{5:7}$				$x_{1:7}$
$x_{5:8}$, $A_{5:8}$	$x_{1:8}$			
x_9				$x_{1:9}$
$x_{9:10}$, $A_{9:10}$				$x_{1:10}$
$x_{9:11}$, $A_{9:11}$				$x_{1:11}$
$x_{9:12}$, $A_{9:12}$		$x_{1:12}$		
x_{13}				$x_{1:13}$
$x_{13:14}$, $A_{13:14}$				$x_{1:14}$
$x_{13:15}$, $A_{13:15}$				$x_{1:15}$
$x_{13:16}$, $A_{13:16}$			$x_{1:16}$	

Table 4.2: Algorithm # 2 example

4.2.3 Algorithm # 3

This algorithm is otherwise known as the partition algorithm (Van der Vorst & Dekker 1989). In this algorithm, we assume for simplicity that the problem size N

can be factored as $N = \beta k$. We partition B given in (4.2) as a $\beta \times \beta$ block matrix, in which the blocks are $k \times k$ block matrices of size $m \times m$ each, as shown below:

$$B = \begin{bmatrix} I & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -A_2 & I & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -A_3 & I & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & 0 & -A_N & I \end{bmatrix}$$

$$= \begin{bmatrix} D_1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -\mathcal{A}_2 & D_2 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\mathcal{A}_3 & I & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & 0 & -\mathcal{A}_\beta & D_\beta \end{bmatrix}, \tag{4.6}$$

where

$$D_i = \begin{bmatrix} I & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -A_{(i-1)k+2} & I & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -A_{(i-1)k+3} & I & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & 0 & -A_{(i-1)k+k} & I \end{bmatrix}, \tag{4.7}$$

and

$$\mathcal{A}_i = \begin{bmatrix} 0 & \dots & 0 & A_{(i-1)k+1} \\ 0 & \dots & 0 & 0 \\ \vdots & & & \\ 0 & \dots & 0 & 0 \end{bmatrix}. \tag{4.8}$$

Note that the $\mathcal{A}_i$ have only one nonzero element in their $(1,k)$ positions. Similarly, the vectors x and d are partitioned in segments $\tilde{x}_i$ and $\tilde{d}_i$ of length mk. Note that x_i denotes the i'th element of x, $\tilde{x}_i$ denotes the i'th segment. Let

$$D = \begin{bmatrix} D_1 & 0 & \dots & 0 \\ 0 & D_2 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & D_\beta \end{bmatrix}, \tag{4.9}$$

$$\mathcal{A} = D - B \tag{4.10}$$

and

$$V = D^{-1}\mathcal{A}. \tag{4.11}$$

Therefore,

$$V_i = D_i^{-1}\mathcal{A}_i = \begin{bmatrix} 0 & \dots & 0 & A_{(i-1)k+1} \\ 0 & \dots & 0 & A_{(i-1)k+1} \cdot A_{(i-1)k+2} \\ \vdots & & & \\ 0 & \dots & 0 & A_{(i-1)k+1} \cdot A_{(i-1)k+2} \cdot \dots A_{(i-1)k+k} \end{bmatrix}. \tag{4.12}$$

Using (4.9), we can rewrite (4.2) as follows:

$$D(I - V)x = d. \tag{4.13}$$

The partition algorithm consists of two steps. The first step is to ignore V and to solve h from

$$Dh = d. \tag{4.14}$$

Once h is obtained, the second (and final) step is to solve for x from

$$(I - V)x = h. \tag{4.15}$$

Figure 4.3 shows the algorithm h for solving h from $Dh = d$ in (4.14). Note that the elements of d and A are overwritten by those of h and V. Figure 4.4 shows the

```
For j = 2 to N step k DO IN PARALLEL
    For i=j to j + k − 2 DO
        d_i = A_i d_{i−1} + d_i;
    EndFor
EndFor
For j = k + 2 to N step k DO IN PARALLEL
    For i=j to j + k − 2 DO
        A_i = A_i A_{i−1};
    EndFor
EndFor
```

Figure 4.3: Algorithm h for Algorithm # 3

```
For j = k + 1 to N step k DO
    For i=j to j + k − 1 DO IN PARALLEL
        d_i = A_i d_{i−1} + d_i;
    EndFor
EndFor
```

Figure 4.4: Algorithm x for Algorithm # 3

algorithm x for solving x from $(I - V)x = h$ in (4.15). The solution x is obtained in d from (4.15).

Let N, β, k, and $A = (A_2, \ldots, A_N)$ and $d = (d_1, d_2, \ldots, d_N)$ be given where $N = \beta k$.

To compute the complexity of this algorithm, we first analyze step 1. From (4.9), D is a matrix consisting of only β diagonal elements, D_i $(1 \leq i \leq \beta)$. Each D_i is a $k \times k$ block bi-diagonal system given in (4.7). Recall that each of the elements in D_i is of size $m \times m$. Therefore, step 1 of this algorithm involves solving for $D_i h_i = d_i$ $(1 \leq i \leq \beta)$, β times. Assume there are p processors available P_0, P_1, $\ldots$,P_{p-1}. Each processor P_i $(0 \leq i \leq p-1)$ computes $D_i h_i = d_i$ $(1 \leq i \leq \beta)$ in parallel. Assume for simplicity that $p = \beta$. Thus, this step will take $2m^2(k-1)$ units of time to complete. Before going to step 2 of this algorithm, we first need to compute $V = D^{-1}\mathcal{A}$ given in (4.11). Matrix V consists of $\beta - 1$ elements, $V_i = D_i^{-1}\mathcal{A}_i$

$(1 \leq i \leq \beta - 1)$. Each V_i, as shown in (4.12), is a $k \times k$ block system. Because of the structure of A_i, given in (4.8), it can be readily verified that V_i is a matrix with only nonzero elements in its k'th column. Each of these nonzero elements is a matrix of size $m \times m$. Computation of V will take $(k-1)(2m^3 - m^2)$ units of time with $\beta - 1$ processors, and a total of $(k-1)(\beta-1)(2m^3 + m^2 - m)$ number of operations. Now the final step of this algorithm is to compute for x in (4.15). Since the x's in the first partition are final and completed, with the other x's in the $\beta - 1$ partitions being in their intermediate stages, there now remains $\beta - 1$ stages needed to compute for those x's. Computation within each stage can be done in parallel by the β processors. In each stage, we have to do a matrix-vector multiplication and a vector additoin of size $(m \times 1)$ k times. Therefore, step 2 of this algorithm will take $2m^2k\frac{\beta-1}{\beta} + \frac{\beta-1}{\beta}(k-1)(2m^3 - m^2)$ units of time with β processors, and a total of $2m^2k(\beta-1) + (\beta-1)(k-1)(2m^3 - m^2)$ number of operations.

Therefore, the partition algorithm requires a total of

$$4(k-1)(\beta-1)m^3 + 2k(\beta-1)m^2 - (k-1)(\beta-1)m$$

scalar operations and takes

$$T_\beta(m, N, \beta) = \frac{2(\beta-1)(N-\beta)m^3}{\beta^2} + \left[\frac{(\beta-1)(N+\beta)}{\beta} + 2(N-\beta)\right]\frac{m^2}{\beta} \quad (4.16)$$

units of time using $p = \beta$ processors.

If only p processors are available where $p \leq \beta$, then we estimate that

$$T_p(m, N, \beta, p) = \frac{2(\beta-1)(N-\beta)m^3}{\beta p} + \left[\frac{(\beta-1)(N+\beta) + 2\beta(N-\beta)}{\beta}\right]\frac{m^2}{p}. \quad (4.17)$$

In this algorithm, if we make the number of partitions, β, small by increasing the partition size, k, then a lot of matrix-matrix multiplications are needed. In fact, the number of matrix-matrix multiplications in V_i is proportional to k. The overhead incurred in this algorithm is largely dominated by these matrix-matrix multiplications. If we make $\beta = N$, essentially the algorithm performs like a serial algorithm, *i.e.*, no matrix-matrix multiplication is needed.

Table 4.3 gives an example of this algorithm. The bottleneck is due to the concurrent reads in step 2. In stage 1, concurrent reads of $x_{1:4}$ are needed to compute $x_{1:5}$ to $x_{1:8}$. In stage 2, concurrent reads of $x_{1:8}$ are needed to compute $x_{1:9}$ to $x_{1:12}$. In stage 3, concurrent reads of $x_{1:12}$ are needed to compute $x_{1:13}$ to $x_{1:16}$.

These concurrent reads will require extra communication cost.

Step 1	Step 2		
	Stage 1	Stage 2	Stage 3
x_1 $x_{1:2}$ $x_{1:3}$ $x_{1:4}$			
x_5 $x_{5:6}$, $A_{5:6}$ $x_{5:7}$, $A_{5:7}$ $x_{5:8}$, $A_{5:8}$	$x_{1:5}$ $x_{1:6}$ $x_{1:7}$ $x_{1:8}$		
x_9 $x_{9:10}$, $A_{9:10}$ $x_{9:11}$, $A_{9:11}$ $x_{9:12}$, $A_{9:12}$		$x_{1:9}$ $x_{1:10}$ $x_{1:11}$ $x_{1:12}$	
x_{13} $x_{13:14}$, $A_{13:14}$ $x_{13:15}$, $A_{13:15}$ $x_{13:16}$, $A_{13:16}$			$x_{1:13}$ $x_{1:14}$ $x_{1:15}$ $x_{1:16}$

Table 4.3: Algorithm # 3 example

4.2.4 Algorithm # 4

Algorithm # 4, commonly known as the cyclic reduction algorithm, is basically divided into 2 phases, namely, the reduction phase and the back-substitution phase (Lakshmivarahan & Dhall 1990).

Let $N, n, A = (A_2, \ldots, A_N)$ and $d = (d_1, d_2, \ldots, d_N)$ be given, where $N = 2^n$. Define for $1 \leq i \leq N$ and $1 \leq j \leq n$,

$$\begin{aligned} A_i^{(0)} &= A_i, \\ d_i^{(0)} &= d_i, \\ A_i^{(j)} &= A_i^{(j-1)} A_{i-2^{j-1}}^{(j-1)}, \\ d_i^{(j)} &= A_i^{(j-1)} d_{i-2^{j-1}}^{(j-1)} + d_i^{(j-1)}. \end{aligned} \tag{4.18}$$

In the reduction phase as shown in Figure 4.5, we first eliminate all the odd-indexed terms. It is then followed by eliminating terms with indices that are odd multiples of 2, 4, 8, ..., etc.. After iterating j times, when $j = n = \log N$, we obtain

$$x_N = d_N^n. \tag{4.19}$$

Now, x_{2^j}, for $j = 0, 1, 2, \ldots, n$ are already found, so the remainder of the $x's$ are obtained by back-substitution as illustrated in Figure 4.6.

To compute the complexity of this algorithm, we first look at the reduction phase. This part of the algorithm consists of $n = \log N$ iterations. For each iteration, j, we eliminate terms with indices that are odd multiples of 2^{j-1}. At the jth step, 2^{n-j} variables are eliminated and each such elimination requires $2m^3 + m^2$ operations. Thus the total number of operations required in this phase is

$$(2m^3 + m^2) \sum_{j=1}^{n} 2^{n-j} = (2m^3 + m^2)(2^n - 1) = (2m^3 + m^2)(N - 1). \tag{4.20}$$

```
For j = 1 to n DO
    For i = 2^j to 2^n step 2^j DO IN PARALLEL
        x_i = A_i^(j) x_{i-2^j} + d_i^(j);
    EndFor
EndFor
```

Figure 4.5: Reduction phase for Algorithm # 4

```
For r = n − 2 to 0 step −1 DO
    For i = 2^r + 2^{r+1} to N − 1 step 2^{r+1} DO IN PARALLEL
        x_i = ( x_{i+2^r} − d^{(r)}_{i+2^r} ) / A^{(r)}_{i+2^r} ;
    EndFor
EndFor
```

Figure 4.6: Back-substitution phase for Algorithm # 4

If there are $\frac{N}{2}$ processors available, the first iteration in the reduction phase takes $2m^3 + m^2$ units of time. For the rest of the iterations in this phase, since the size of the problem is halved, the matrix-matrix multiplications in the computation of A_i's can be done in parallel with the matrix-vector multiplications in the computation of d_i's. Since it takes longer to compute the matrix-matrix multiplications, the time required is $(2m^3 - m^2 + m)$ units. The entire reduction phase requires a total of

$$(2m^3 - m^2 + m)(n-1) + 2m^3 + m^2 = (2m^3 - m^2 + m)\log N + 2m^2 - m$$

units of time.

In the back-substitution phase, with x_{2^s} known for $s = 0, 1, \ldots, \log N$, the remaining of the x_i's are computed. From Figure 4.6, it can be seen that at the r'th step, $2^{n-r-1} - 1$ variables are computed and each computaion takes a matrix-vector multiplication and vector-vector subtraction, *i.e.*, $2m^2 - m + m = 2m^2$ operations.

The total number of operations in this phase are given by

$$2m^2 \sum_{r=0}^{n-2} \left(2^{n-r-1} - 1\right) = 2m^2(N - \log N - 1).$$

Given $\frac{N}{2}$ processors, this phase takes a total of $2m^2(n-1) = 2m^2(\log N - 1)$ units of time.

Therefore the cyclic reduction algorithm requires a total of

$$\begin{aligned}&\left(2m^3 + m^2\right)(N-1) + 2m^2(N - \log N - 1)\\ &\quad = N\left(2m^3 + 3m^2\right) - \left(2m^3 + 3m^2\right) - 2m^2 \log N\end{aligned}$$

$$= (N-1)\left(2m^3+3m^2\right) - 2m^2 \log N \tag{4.21}$$

scalar operations and takes

$$\begin{aligned} T_{\frac{N}{2}}(m,N) &= (2m^3 - m^2 + m)\log N + 2m^2 - m + 2m^2(\log N - 1) \\ &= (2m^3 + m^2 + m)\log N - m \end{aligned} \tag{4.22}$$

steps to complete using $\frac{N}{2}$ processors. If only p processors are available, where $p \leq \frac{N}{2}$, then

$$T_p(m,N,p) = \frac{N}{2p}\left(2m^3 + m^2 + m\right)\log N - \frac{Nm}{2p}. \tag{4.23}$$

An example of the algorithm is shown in Table 4.4. Here, the first five columns of Table 4.4 represent the reduction phase where $x_{1:1}$, $x_{1:2}$, $x_{1:4}$, $x_{1:8}$ and $x_{1:16}$ are computed. Once $x_{1:16}$ is computed, the remaining of the x's are obtained by back substitution.

Since the problem to be solved is a block bi-diagonal system, the reduction phase involves matrix-matrix multiplications and the back-substitution phase involves the use of some factorization method which limits parallelism. The algorithm is also found to be unstable except when the sub-blocks $A_i (2 \leq i \leq N)$ are diagonally dominant (Heller 1976, Fujishiro, Ikebe, Harashima & Watanabe 1989, Buzbee, Golub & Nielson 1970).

$x_1 = d_1$							
$x_2 = d_2$	$x_{1:2}, A_{1:2}$						
$x_3 = d_3$							$x_{1:3}$
$x_4 = d_4$	$x_{3:4}, A_{3:4}$	$x_{1:4}, A_{1:4}$					
$x_5 = d_5$							$x_{1:5}$
$x_6 = d_6$	$x_{5:6}, A_{5:6}$					$x_{1:6}$	
$x_7 = d_7$							$x_{1:7}$
$x_8 = d_8$	$x_{7:8}, A_{7:8}$	$x_{5:8}, A_{5:8}$	$x_{1:8}, A_{1:8}$				
$x_9 = d_9$							$x_{1:9}$
$x_{10} = d_{10}$	$x_{9:10}, A_{9:10}$					$x_{1:10}$	
$x_{11} = d_{11}$							$x_{1:11}$
$x_{12} = d_{12}$	$x_{11:12}, A_{11:12}$	$x_{9:12}, A_{9:12}$			$x_{1:12}$		
$x_{13} = d_{13}$							$x_{1:13}$
$x_{14} = d_{14}$	$x_{13:14}, A_{13:14}$					$x_{1:14}$	
$x_{15} = d_{15}$							$x_{1:15}$
$x_{16} = d_{16}$	$x_{15:16}, A_{15:16}$	$x_{13:16}, A_{13:16}$	$x_{9:16}, A_{9:16}$	$x_{1:16}, A_{1:16}$			

Table 4.4: Algorithm # 4 example.

4.3 Benchmarks

In this Section, the four vector/parallel algorithms for the block, first-order, bidiagonal problem are compared against each other and against the vectorized serial algorithm. The sum total CPU time of all the processors and the wall clock time are recorded. The goal is to discover whether or not any of these four vector/parallel algorithms can be cost-effective when compared with the vectorized serial algorithm. To be deemed *cost-effective* requires that at least the wall-clock time for a vector/parallel algorithm be less than that of the vectorized serial algorithm. The serial algorithm has far fewer number of operations, the sum total CPU time for a vector/parallel algorithm is not expected to be less for a vector/parallel algorithm than for a serial algorithm. Depending on the application and the users needs, various weight would be put on the sum total CPU time in the determination of whether

or not an algorithm is deemed cost-effective. For example, in a weather forecasting situation with access to infinite processors, the only measure of cost-effectiveness of an algorithm is the wall-clock time. In a research mode on a crowded machine, with many jobs waiting in the queue, the sum total CPU time would be much more important.

The CrayJ90 belongs to a class of parallel computers where the multiprocessors have a shared memory organization. In other words, an array of processors and an array of common bank of memory units are connected through a fast bus or one of many types of multistage dynamic interconnection networks. The processors communicate by writing onto and reading from the common or shared memory. The tests are done on the CrayJ90 using either p=1, 2, 4 or 8 vector processors. In all the tests, a problem of size N=64 is being used.

The vector/parallel Algorithms # 1 through # 4 are tested for two different structures of the sub-diagonal blocks, $A_i (2 \leq i \leq N)$, namely tridiagonal and dense. These eight categories allow for many variations: either row-wise or column-wise storage of matrices A and vector d (Golub & Loan 1989), and either saxpy or dot product operations (Golub & Loan 1989). In Algorithm # 3, there is also a partition factor β to be accounted for. In order to facilitate finding the most cost-effective variant for these eight algorithms, we first conduct timings with block size of $m = 32$, so that clear "losers" can be identified for which no further testing is required. The eight "winners" are allowed to proceed to a test with $m = 128$.

The values of the sub-diagonal blocks A_i and the vectors d_i are generated using a random number generator. By row-wise or column-wise storage of matrix A, we meant the blocks of matrices A_i $(2 \leq i \leq N)$, are stored either row-wise or column-wise, into a matrix A of size $(1 : m) \times (1 : (N-1)m)$ or $(1 : (N-1)m) \times (1 : m)$ respectively. Similarly, vector d is a collection of all the vectors d_i $(1 \leq i \leq N)$, which are stored either row-wise or column-wise, of size $1 \times Nm$ or $Nm \times 1$ respectively.

In algorithms that access arrays, it is important to organize the computations so that the inner loops access contiguous data. In the first round of testing, we found that there is about a 10% decrease in the timings when using saxpy, rather than dot product operations. Likewise, there is about a 15% reduction when using row-wise storage rather than column-wise storage. Such reductions are expected because the saxpy operations, combined with the row-wise storage scheme of A_i,

using FORTRAN 90, essentially access the data contiguously in the inner loops. For Algorithm #3, the larger the β, the better the performances of the algorithm. When $\beta = N$, Algorithm #3 is a serial algorithm and no matrix-matrix multiplications are needed. Therefore the "winners" in the first round of testing were saxpy operations using row-wise storage, and $\beta = N$ for Algorithm #3.

We recall that in the previous chapter, we obtained two different types of sub-diagonal block structure under the Euler and the leapfrog discretization schemes. For the Euler scheme, the sub-diagonal block structure is tridiagonal. For the leapfrog scheme, the sub-diagonal block structure is dense. In Section 4.3.1 we will discuss the benchmarkings for the case where the sub-matrices, A_i (for $2 \leq i \leq N$), are dense. In Section 4.3.2 we will discuss the benchmarkings for the case where the sub-matrices, A_i (for $2 \leq i \leq N$), are tridiagonal.

We summarize the total number of operations in the limit of large m:

- **tridiagonal serial:**

$$6m(N-1) \tag{4.24}$$

- **dense serial :**

$$2m^2(N-1) \tag{4.25}$$

- **Algorithm #1:**

$$2Nm^3(\log p + 1) \tag{4.26}$$

- **Algorithm #2:**

$$4Nm^3 \tag{4.27}$$

- **Algorithm #3:**

$$4(k-1)(\beta-1)m^3 \tag{4.28}$$

- **Algorithm #4:**

$$2(N-1)m^3 \tag{4.29}$$

We summarize the complexity formulas in the limit of large m:

- **tridiagonal serial:**

$$T(m,N) = 6m(N-1) \tag{4.30}$$

- **dense serial :**

$$T(m,N) = 2m^2(N-1) \tag{4.31}$$

- **Algorithm #1:**

$$T_{\frac{3}{2}p}(m,N,p) = \frac{N}{p}2m^3(\log p + 1) \tag{4.32}$$

- **Algorithm #2:**

$$T_{\frac{3}{2}p}(m,N,p) = \frac{4Nm^3}{p} \tag{4.33}$$

- **Algorithm #3:**

$$T_p(m,N,\beta,p) = \frac{2(\beta-1)(N-\beta)m^3}{\beta p} \tag{4.34}$$

- **Algorithm #4:**

$$T_p(m,N,p) = \frac{Nm^3}{p}\log N \tag{4.35}$$

Note that we have not listed tridiagonal variants of the complexity formulas for Algorithms #1 through #4. Although the sub-diagonal blocks may be tridiagonal in structure, they do not remain so after a matrix-matrix multiplication. After the first stage of the algorithms, the sub-blocks essentially become dense. Thus we can take only very limited advantage of the tridiagonal structure of the sub-diagonal blocks in Algorithms #1 through #4. On the other hand, since the serial algorithm does not require matrix-matrix multiplications, we are able to realize enormous savings in the number of operations when the sub-blocks are tridiagonal.

4.3.1 Dense Case

In the limit of large N, we expect the following redundancy factors for the dense case:

- **Algorithm #1:**

$$(\log p + 1)m$$

- **Algorithm #2:**

$$2m$$

- **Algorithm #3:**

$$2\frac{(1-k-\beta)}{(N-1)}m$$

- **Algorithm #4:**

$$m$$

In the limit of large N, we expect the following ratios of wall-clock time for the dense case:

- **Algorithm #1:**

$$\frac{m}{p}(\log p + 1)$$

- **Algorithm #2:**

$$\frac{2m}{p}$$

- **Algorithm #3:**

$$\frac{(\beta-1)(1-\frac{\beta}{N})m}{\beta p}$$

- **Algorithm #4:**

$$\frac{m}{2p}\log N$$

Note that for Algorithm #2, we expect p will need to be at least $2m$ for the parallel algorithm to break even with the wall-clock time of the serial algorithm. For Algorithm #3, if p is sufficiently large in the limit of large N, the value of p will need to be at least m before the parallel algorithm can break even. In Algorithms #1 and #4, the value of p needed to break even will be larger still because of the $\log p$ and the $\log N$ terms. On the CrayJ90. which was used for the experiments, there are only 8 processors available. Thus, we do not expect to be able to demonstrate the effectiveness of these algorithms with the values of m used here. Furthermore, most modern meteorological models will have m of order 10^6, so a very large number of processors would be needed for these parallel algorithms to be effective in such a case.

Figures 4.7 through 4.10 show the theoretical and practical results of the ratio of the wall-clock time of Algorithms #1 through #4 to the wall-clock time of the serial

algorithm. The figures also show the measured ratio of CPU time. The computed ratios are based on a serial timing of .067 s for $m = 32$ and .908 s for $m = 128$. On the CrayJ90, it is possible to turn off the vectorization capabilities of a processor. The timings for the serial algorithm, without vector processing, would be about four times greater than these values.

Except for Algorithm #4 in Figure 4.10 and for $\beta = 8$ in Algorithm #3 in Figure 4.9, we see a substantial increase in CPU time as more processors are used. The theoretical ratio of CPU time, based on the complexity formulas, would be a flat curve because the complexity formulas do not account for any extra "overhead" of multi-processing. It is interesting to note that both Algorithm #4 and Algorithm #3 (for $\beta = 8$), actually approach the ideal of a constant CPU time.

For Algorithm #3 in Figure 4.9, when $\beta = N(= 64)$, no matrix-matrix multiplications are needed, and the algorithm is a serial algorithm. Therefore, increasing p has very little effect on the wall-clock time, but the increase in CPU time is dramatic as all but one of the processors remain idle. Nevertheless, $\beta = 64$ is still the fastest configuration we were able to test.

For $\beta < N$ the algorithm allows for β concurrent matrix- matrix multiplications. For example, when $\beta = 32$, the partition size is 2 and there are 32 matrix-matrix multiplications that could, in principle, be done concurrently. A steady improvement in wall-clock time would, in principle, occur as p is increased to 32. However, we were able to observe a decrease in wall-clock time for only up to $p = 8$, the limit of our CrayJ90. We are able to see for $\beta = 2$ and $\beta = 4$, that there is no improvement in wall-clock time for $p > \beta$. Thus, the results here illustrate the principle that the best use of this parallel algorithm is with $\beta = p$.

In summary, the rate of increase of the ratio of the CPU time is due to the high redundancy factors in the parallel algorithms. We find that the theory (based on the complexity formulas and the redundancy factors) give excellent guidance about the wall-clock time of the parallel algorithms. The experimental ratio is typically within a factor of two of the theory. In most circumstances, a decision about which parallel algorithm to use, or whether to use one of these algorithms at all, could be confidently made with the use of the complexity formulas. The discrepancies between the theory and practice could invite a further explanation involving issues of "latency", "bandwidth", computer architecture and hardware. Such a discussion

is beyond the scope of this investigation. For one reason, these issues are rendered irrelevant in determining cost-effectiveness, in light of the overwhelming effect of simply the number of arithmetic operations involved in the algorithm.

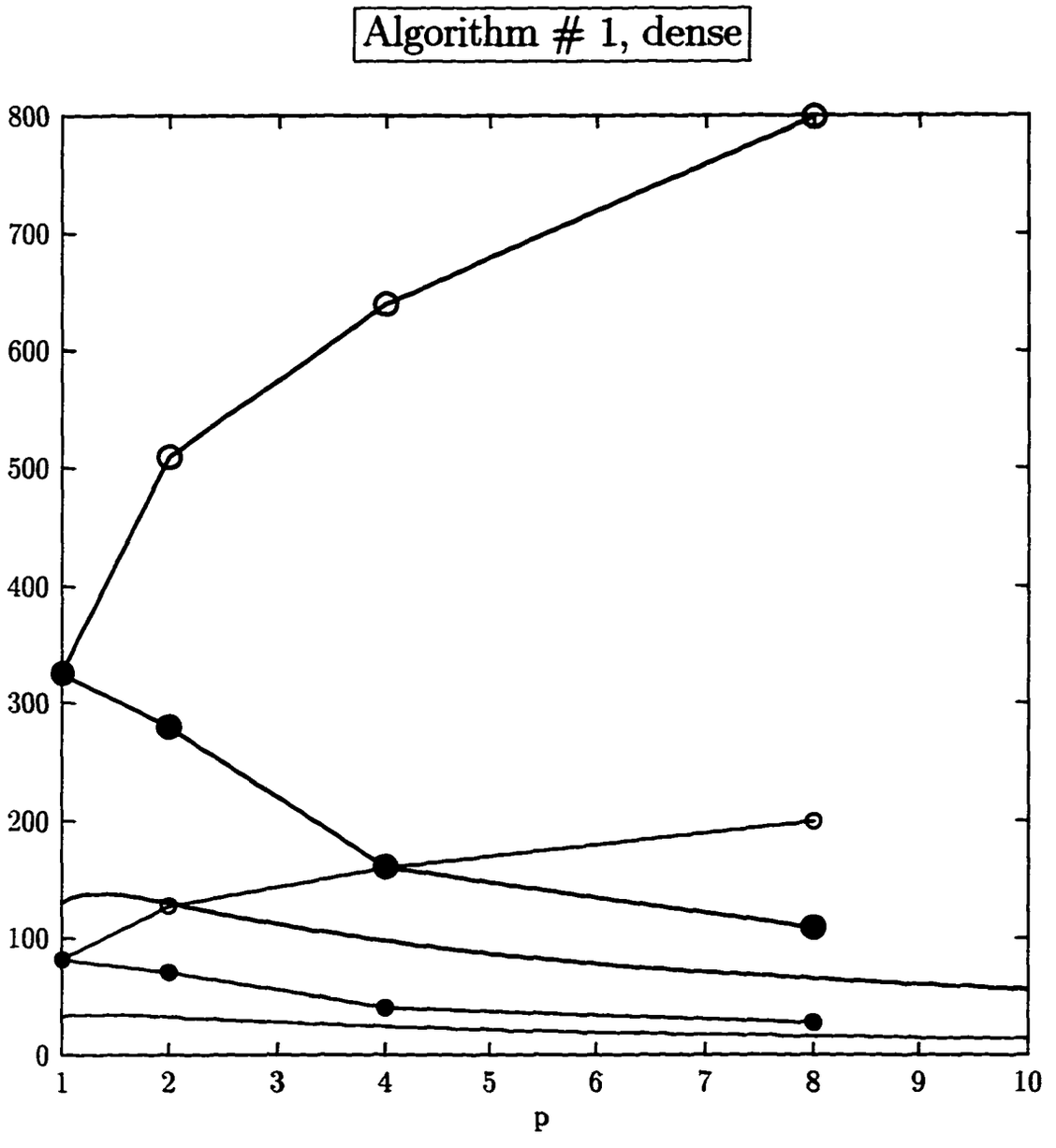


Figure 4.7: Algorithm #1 with dense sub-blocks: the ratio of the wall-clock time and CPU time to that of the serial algorithm. The ● indicates the wall-clock benchmark for $m = 128$, the ○ indicates the CPU time for $m = 128$. The • indicates the wall-clock benchmark for $m = 32$, the ∘ indicates the CPU time for $m = 32$. The lines without the symbols are the theoretical wall-clock ratio derived from the complexity formulas. The uppermost is for $m = 128$, the lower is for $m = 32$.

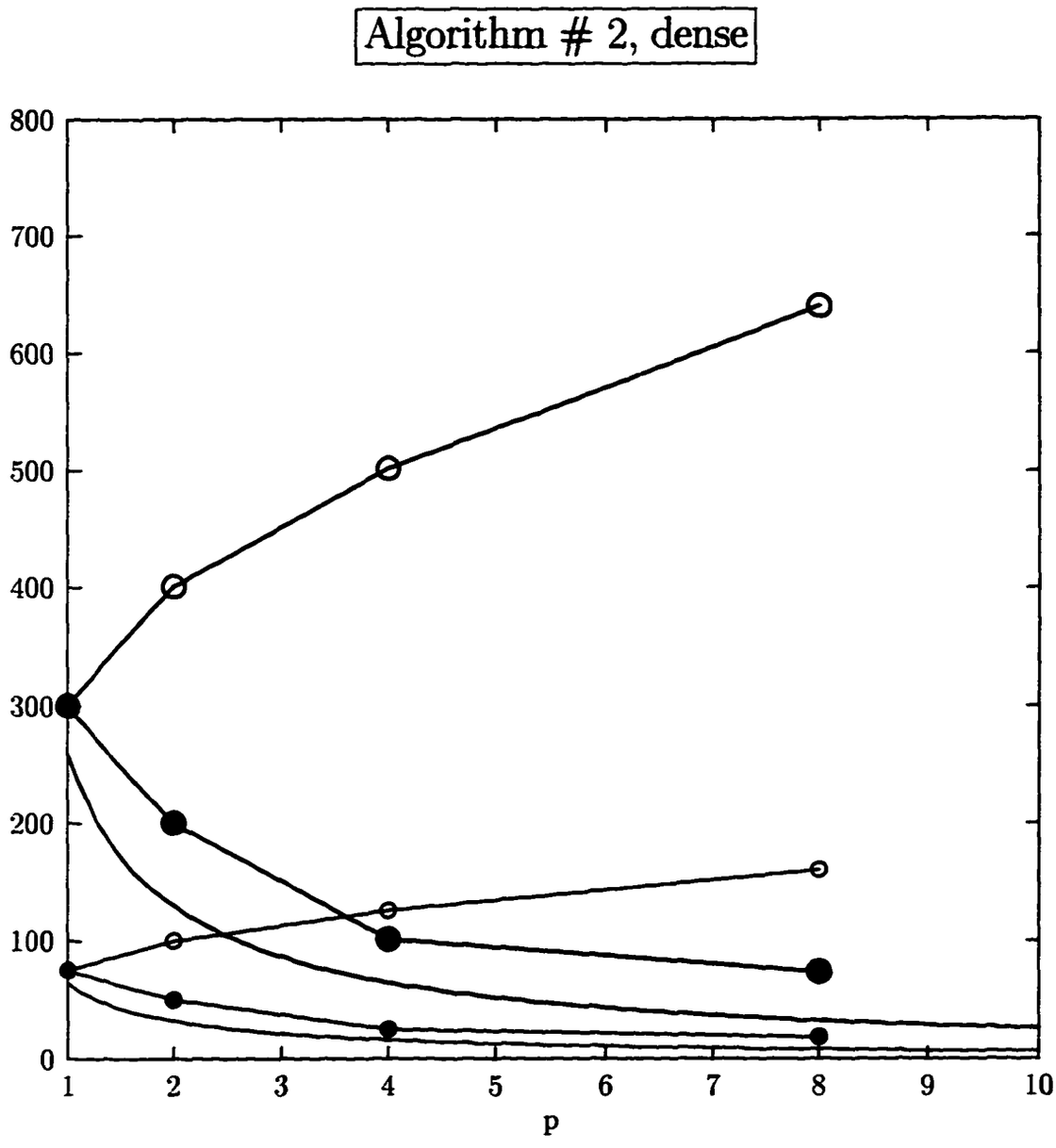


Figure 4.8: As in Figure 4.7, but Algorithm #2 with dense sub-blocks.

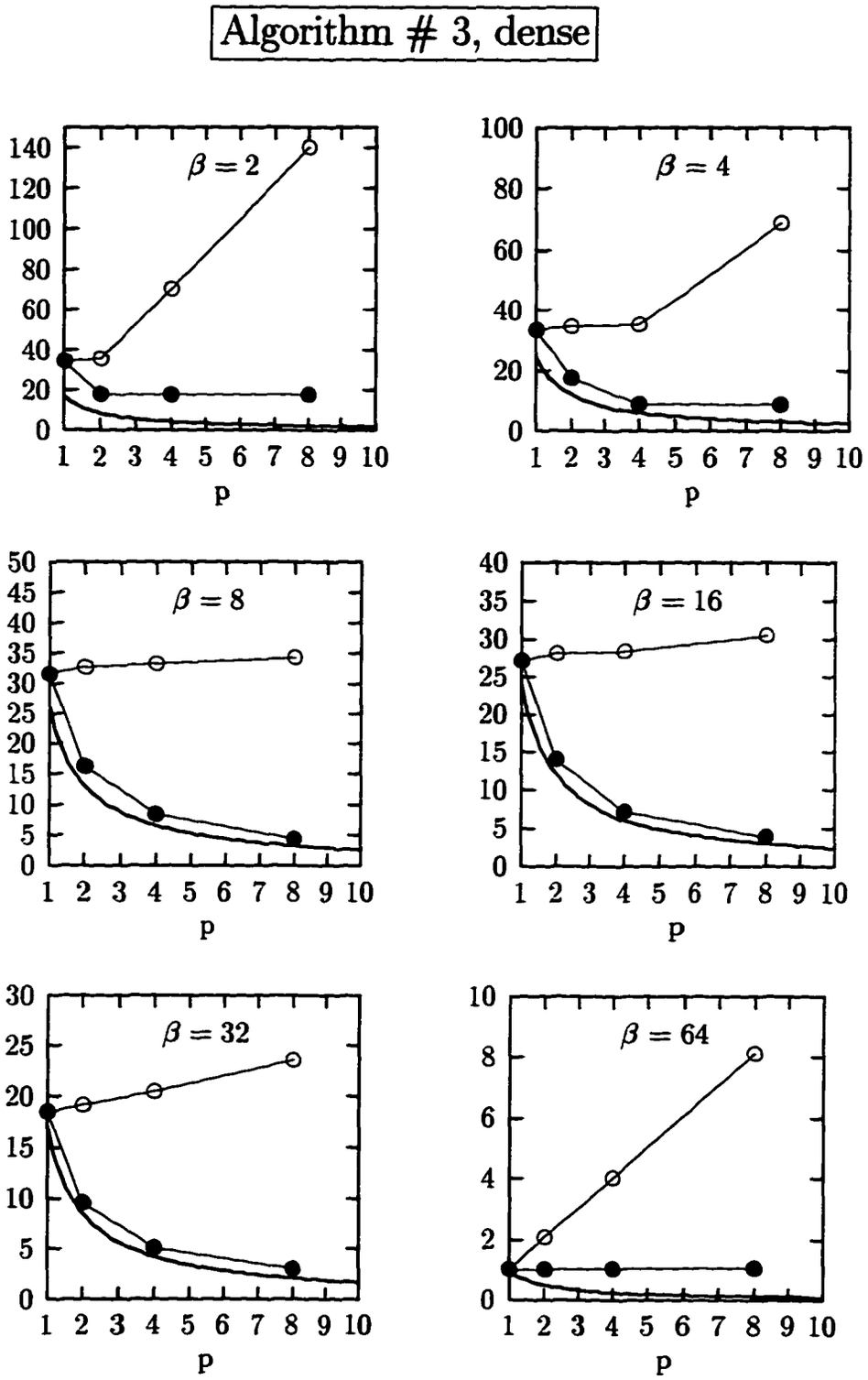


Figure 4.9: Algorithm #3 with dense sub-blocks and $m = 32$. The ratio of the wall-clock time • and CPU time o to that of the serial algorithm. The line without the symbols is the theoretical wall-clock ratio derived from the complexity formulas.

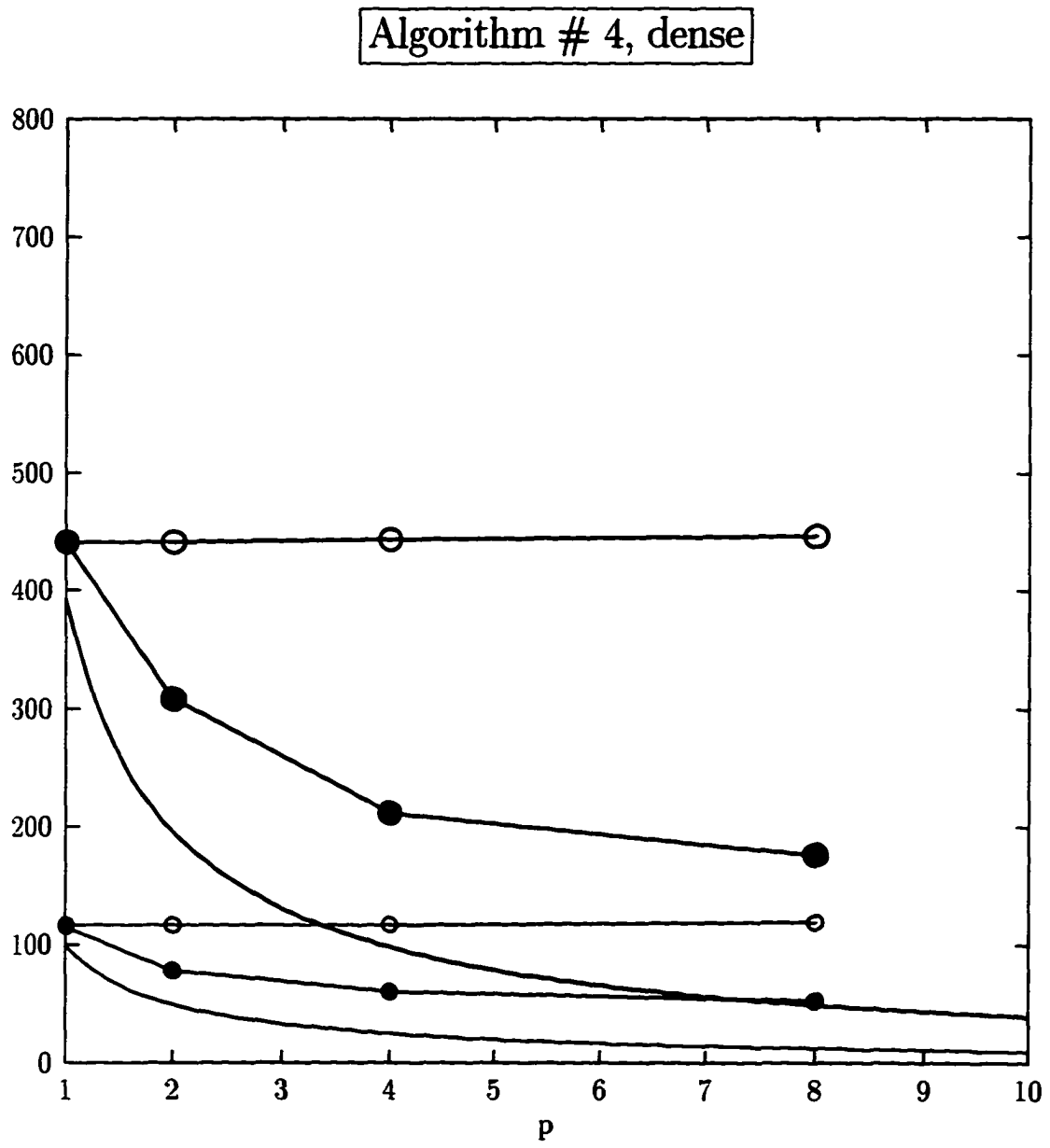


Figure 4.10: As in Figure 4.7, but Algorithm #4 with dense sub-blocks.

4.3.2 Tridiagonal Case

In the limit of large N, we expect the following redundancy factors for the tridiagonal case:

- **Algorithm #1:**
$$\frac{1}{3}(\log p + 1)m^2$$
- **Algorithm #2:**
$$\frac{2m^2}{3}$$
- **Algorithm #3:**
$$\frac{2}{3}\frac{(1-k-\beta)}{(N-1)}m^2$$
- **Algorithm #4:**
$$\frac{m^2}{3}$$

In the limit of large N, we expect the following ratios of wall-clock time for the tridiagonal case:

- **Algorithm #1:**
$$\frac{m^2}{3p}(\log p + 1)$$
- **Algorithm #2:**
$$\frac{2m^2}{3p}$$
- **Algorithm #3:**
$$\frac{(\beta-1)(1-\frac{\beta}{N})m^2}{3\beta p}$$
- **Algorithm #4:**
$$\frac{m^2}{6p}\log N$$

Figures 4.11 through 4.14 are for the tridiagonal case. They are similar to figures 4.7 through 4.10 for the dense case. Except for the scale on the ordinate, the two sets of figures are nearly the same. Note that "savings" are realized only in the

serial algorithm, which makes the relative performance of the parallel algorithms proportional to m^2. The figures are based on a serial timing of .007 s for $m = 32$ and .022 s for $m = 128$.

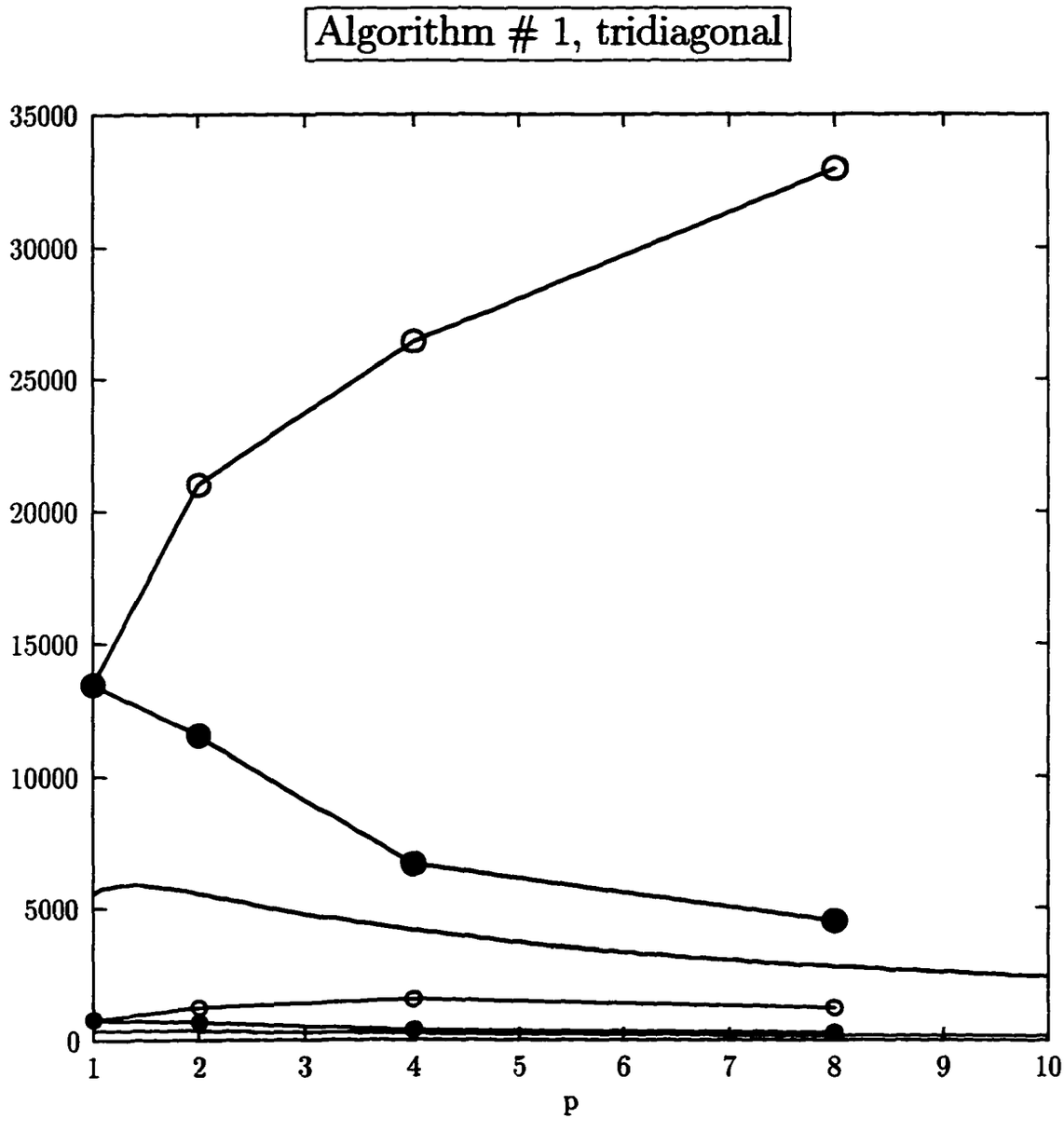


Figure 4.11: As in Figure 4.7, but Algorithm #1 with tridiagonal sub-blocks.

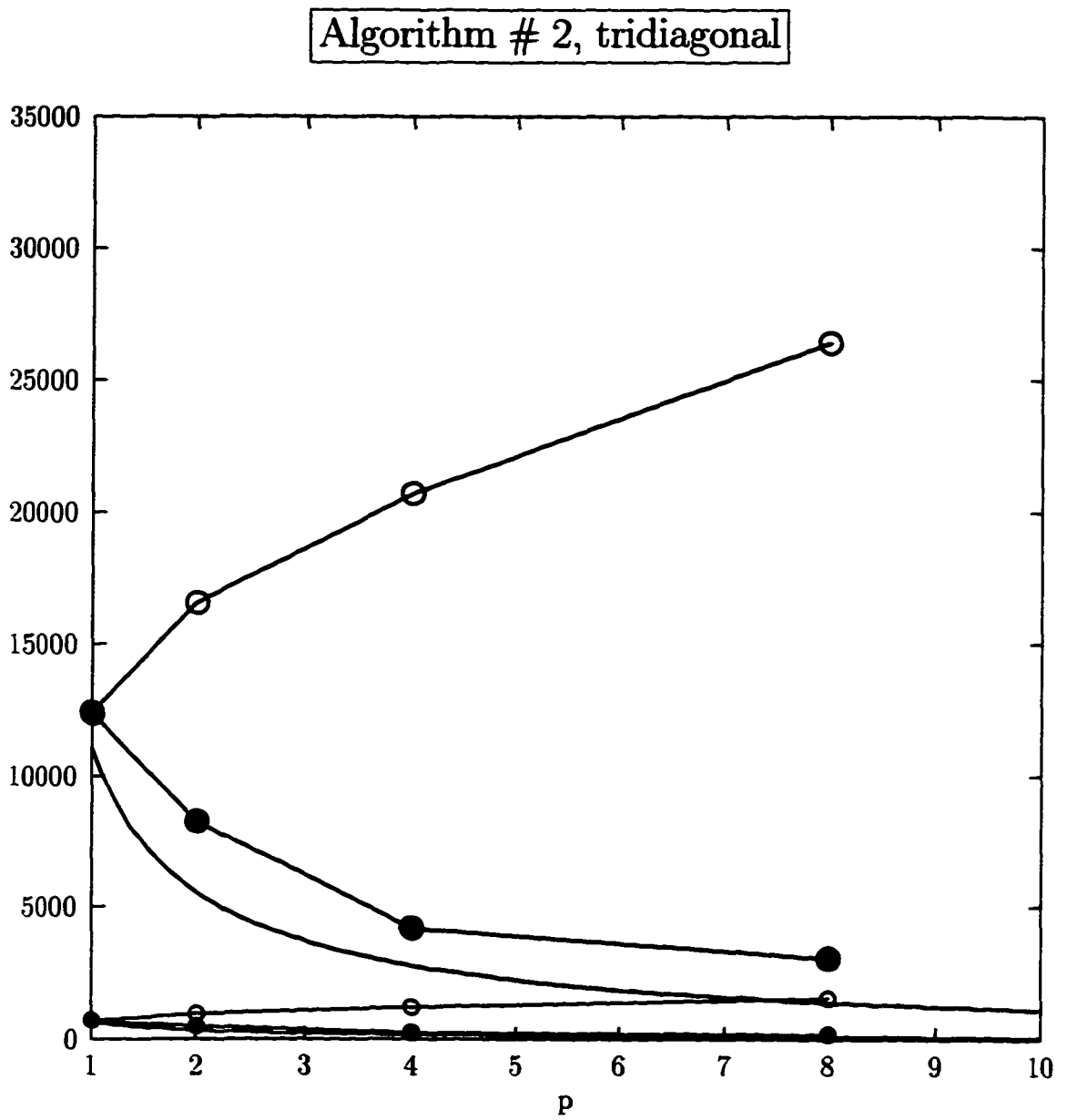


Figure 4.12: As in Figure 4.7, but Algorithm #2 with tridiagonal sub-blocks.

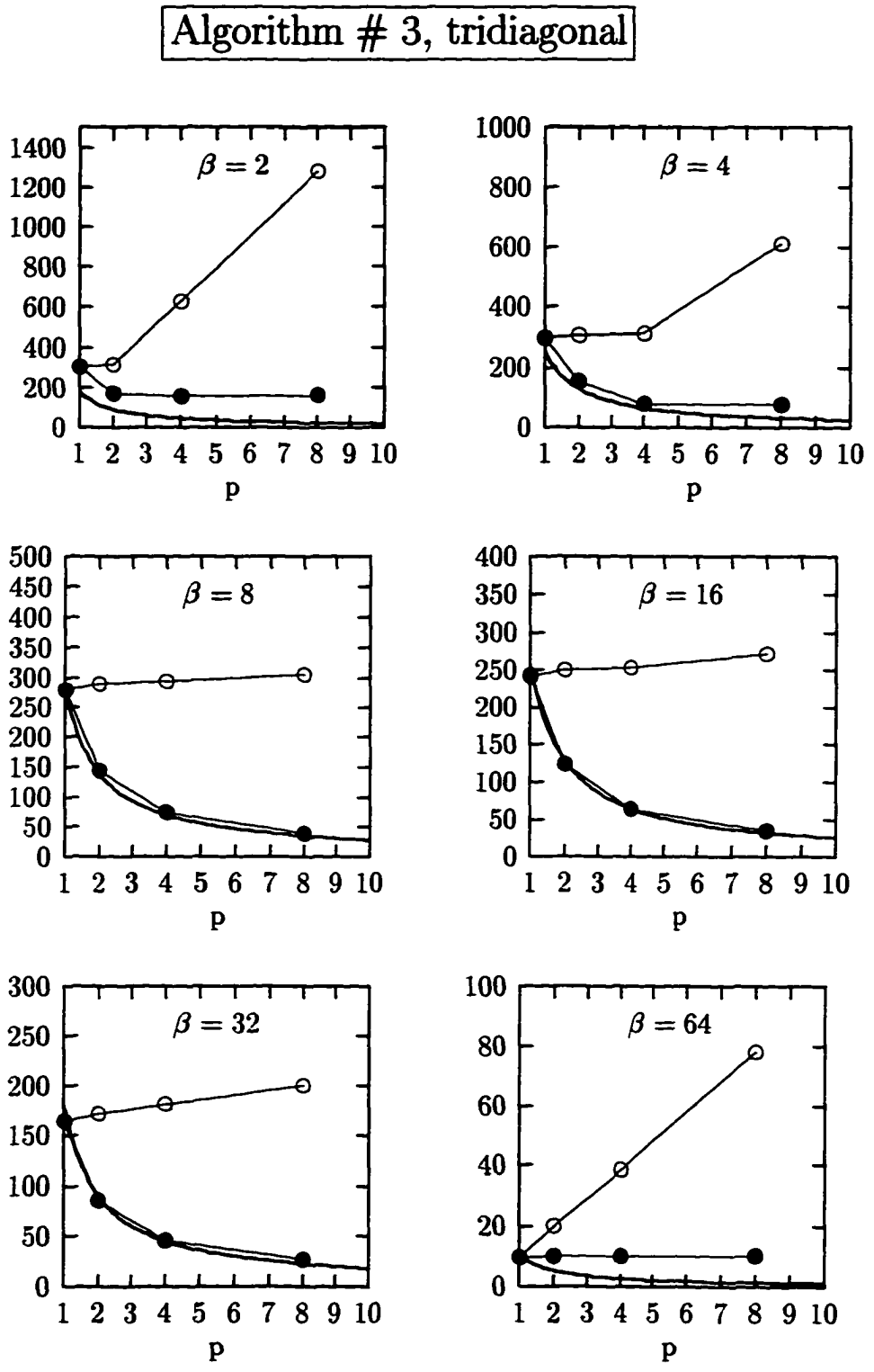


Figure 4.13: As in Figure 4.9, but with tridiagonal sub-blocks.

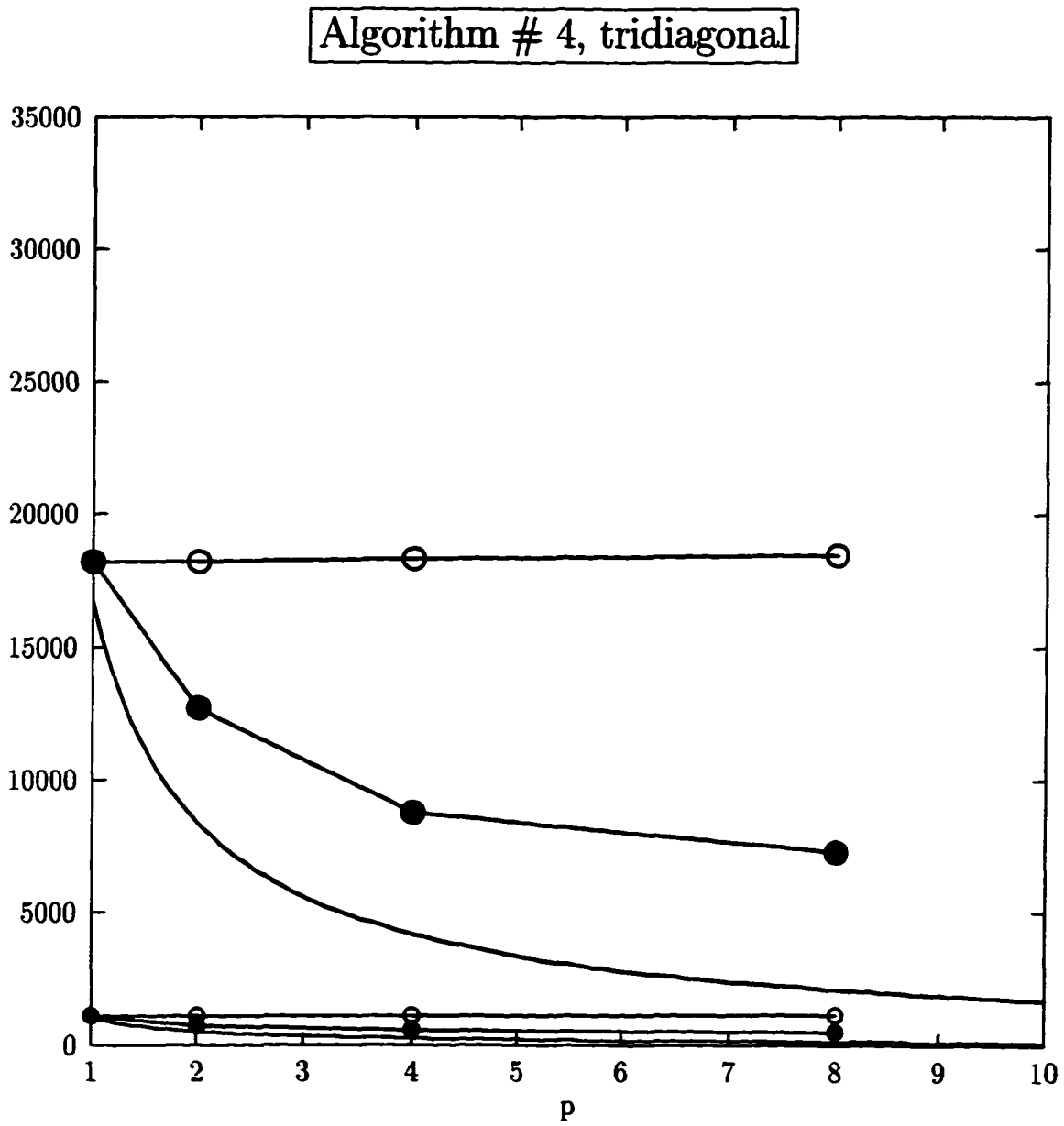


Figure 4.14: As in Figure 4.7, but Algorithm #4 with tridiagonal sub-blocks.

4.4 Conclusions

The four vector/parallel algorithms are found to be not cost-effective on the CrayJ90, to which exclusive access has been obtained for the tests. There is not much distinction to be made among the four vector/parallel algorithms, since the number of operations in all of them is proportional to m^3. From the results obtained, it can be seen that the vector/parallel algorithms when compared to the vectorized serial algorithm, fare worse for the tridiagonal sub-matrices than for the dense sub-matrices. This is because the serial algorithm is able to fully exploit the presence of a sparse structure of a tridiagonal matrix, thus greatly reducing the number of operations. Only about a 50% savings is realized in the vector/parallel algorithms for tridiagonal sub-matrices.

Chapter 5

Conclusions

We have demonstrated the use of the adjoint method on data assimilation for two meteorological models. The first model, the mixed-layer model, is nonlinear and has a "small" number of unknown variables/parameters The second model is the shallow water model. It is linear and has a "large" number of unknown variables/parameters.

From the mixed-layer model, it is found that there are at least two factors that can affect the rate of convergence of the iterates in determining the optimal estimates of the control parameters.

Firstly, it depends on the quality of the data sets, where a minimum number of observations are needed to estimate the control vector to a desired degree of accuracy. In our example, we need at least four observations in order to render our estimates meaningful.

Secondly, proper scaling of the model variables is needed for the minimization algorithm to not depend on the units of measure of the variables/parameters. This is crucial in order to obtain accurate and precise estimates.

The solutions of the forward model and the adjoint model of the shallow water equations give rise to a block bi-diagonal system. Our motivation in solving the computatoinal aspects of this model led us to do a comparative analysis of four vector/parallel algorithms which are found not to be cost effective when compared with the vectorized serial algorithm on the CrayJ90. This is because the parallel algorithms are rich in matrix-matrix multiplications. These give rise to huge amount of memory traffic, not to mention the actual amount of arithmetic involved. Matrix-matrix multiplication is a level 3-operation which involves a quadratic amount of data ($O(m^2)$) and a cubic amount of work ($O(m^3)$).

The design of matrix algorithms that are rich in high-level linear algebra operations is an area of intensive research. The efficiency of such algorithms depend on many things such as the amount of arithmetic and storage required by a procedure. However, less obvious but often crucial to performance are memory access, stride, and a host of other data movement overheads which are extremely important for algorithms involving matrix-matrix multiplications.

A possibly more fruitful avenue for developing algorithms for the very large matrices encountered in meteorology is to return to the fundamental serial algorithm:

$$\begin{aligned} x_1 &= d_1, \\ x_i &= A_i x_{i-1} + d_i \qquad (2 \leq i \leq N), \end{aligned} \tag{5.1}$$

where x_i and d_i are m-vectors, for $i = 1, 2, \ldots, N$. With A_i being a sparse $m \times m$ matrix, with m of order 10^6, perhaps effective parallel algorithms could be applied to the matrix-vector product $A_i x_{i-1}$. For example, if there are p processors available, then m/p rows of A_i could be put out in parallel to p processors for multiplication by x_{i-1}. Such parallelization would thus resemble the physical domain decomposition that is oftened used in a forward meteorological model.

Also, one of the four vector/parallel algorithms, namely, the cyclic reduction algorithm, is found to be unstable unless the sub-blocks are diagonally dominant. It would be of interest to determine a variant version of the cyclic reduction algorithm (i.e., analogous to the Buneman's algorithm for the block tri-diagonal system) for this case.

Reference List

Ball, F. K. (1960), 'Control of inversion height by surface heating', *Quart. J. Roy. Meteor. Soc.* **86**, 483–494.

Burk, S. & Thompson, W. (1992), 'Airmass modification over the Gulf of Mexico : Mesoscale model and airmass transformation model forecasts', *J. Appl. Meteror.* **31**, 925–937.

Busch, N. E. (1977), Fluxes in the surface boundary layer over the sea, *in* E. B. Krauss, ed., 'Modelling and Prediction of the Upper Layers of the Ocean', Pergamon Press, pp. 72–91.

Buzbee, B. L., Golub, G. H. & Nielson, C. W. (1970), 'On direct methods for solving Poisson's equations', *SIAM J. Numer. Anal.* **7**, 627–656.

Callies, V. & Eppel, D. P. (1992), On the eigenvalue spectrum of the Hessian matrix when fitting discretized hyperbolic equations to measurements, Technical report, GKSS, Germany.

Carson, D. J. (1973), 'The development of a dry inversion - capped convectively unstable boundary layer', *Quart. J. Roy. Meteor. Soc.* **99**, 450–467.

Chung, W. L. (1996), Analysis of data distribution in variational assimilation schemes of interest in meteorology using Burger's equation, Ph. d. dissertation, Department of Computer Science, Univ. of Oklahoma.

Conn, H. E. & Podrazik, L. J. (1994), 'Parallel recurrence solvers for vector and SIMD supercomputers', *J. of Parallel and Distributed Computing* **23**, 435–445.

Crisp, C. A. & Lewis, J. M. (1992), 'Return flow in the Gulf of Mexico. part 1 : A classificatory approach with a global historical perspective', *J. Appl. Meteor.* **31**, 868–881.

Daley, R. (1991), *Atmospheric Data Analysis*, Cambridge Uni. Press.

Fujishiro, I., Ikebe, Y., Harashima, A. & Watanabe, M. (1989), 'A note on cyclic reduction Poisson solvers with application to bioconvective phenomena problems', *Computers and Fluids* **17**, 419–435.

Ghil, M. & Malanotte-Rizzoli, P. (1991), 'Data assimilation in meteorology and oceanography', *Advances in Geophysics* **33**, 141–265.

Golub, G. H. & Loan, C. F. V. (1989), *Matrix computations*, Johns Hopkins University Press.

Heller, D. (1976), 'Some aspects of the cyclic reductin algorithm for block tridiagonal linear systems', *SIAM J. Numer. Anal.* **13**, 484–496.

Lakshmivarahan, S. & Dhall, S. (1990), *Analysis and Design of Parallel Algorithms: Arithmetic and Matrix problems*, McGraw-Hill.

Lanczos, C. (1970), *The Variational Principles of Mechanics*, Univ. of Toronto Press, chapter 2.

Lewis, J. M. & Crisp, C. A. (1992), 'Return flow in the Gulf of Mexico. part II : Variability in return-flow thermodynamics inferred from trajectories over the Gulf', *J. Appl. Meteor.* **31**, 882–898.

Lewis, J. M., Hayden, C. M., Merrill, R. & Schneider, J. M. (1989), 'GUFMEX: A study of return flow in the Gulf of Mexico', *Bull. Amer. Meteor. Soc.* **70**, 24–29.

Li, Y., Droegemeier, K. K. & Lewis, J. M. (1991), Multiple minima in the cost functional of variational four-dimensional data assimilation methods : Their origin and role in the predictability of nonlinear dynamical systems, *in* '9th conference on Numerical Weather Prediction'.

Lilly, D. K. (1968), 'Models of cloud-topped mixed layers under a strong inversion', *Quart. J. Roy. Meteor. Soc.* **94**, 292–309.

Liu, Q., Lewis, J. M. & Schneider, J. M. (1992), 'A study of cold-air modification over the Gulf of Mexico using in site data and mixed-layer modeling', *J. Appl. Meteor.* **31**, 909–924.

Luenberger, D. G. (1973), *Introduction to Linear and Nonlinear Programming*, Addison Wesley.

Mesinger, F. & Arakawa, A. (1976), Numerical methods used in atmospheric models, Report, World Meteorological Organization GARP publications series no. 17.

Meyer, G. & Podrazik, L. (1987), 'A parallel first-order linear recurrence solver', *J. of Parallel and Distributed Computing* pp. 117–132.

Meyer, G. & Podrazik, L. (1989), Parallel iterative algorithms for optimal control, *in* 'Proceedings of the Third SIAM Conference on parallel processing for scientific computing', pp. 250–254.

Ortega, J. M. (1988), *Introduction to parallel and vector solution of linear systems*, Plenum.

Pedlosky, J. (1987), *Geophysical Fluid Dynamics*, Springer-Verlag.

Smith, G. (1978), *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford University Press.

Sun, J. & Flicker, D. W. (1991), 'Recovery of three-dimensional wind and temperature fields from simulated single-doppler radar data', *J. Atmos. Sci.* **48**, 876–890.

Tennekes, H. (1973), 'A model for the dynamics of the inversion above a convective boundary layer', *J. Atmos. Sci.* **30**, 558–567.

Thacker, W. C. (1989), 'The role of the Hessian matrix in fitting models to measurements', *Journal of Geophysical Research* **94**, 6177–6196.

Thacker, W. C. & Long, R. B. (1988), 'Fitting dynamics to data', *Journal of Geophysical Research* **93**, 1127–1240.

Van der Vorst, H. A. (1988), 'Parallel solution of bidiagonal systems coming from discretized pde's', *IEEE Trans. Magnetics* **24**, 286–290.

Van der Vorst, H. A. & Dekker, K. (1989), 'Vectorization of linear recurrence relations', *SIAM J. Sci. Sta. Comput.* **10**, 27–35.

Wolfsberg, D. G. (1987), Retrieval of three-dimensional wind and temperature fields from single doppler radar data, Report, CIMMS, U. of Oklahoma.