To the Graduate Council:

I am submitting herewith a thesis written by Robert N. Stewart entitled "Parallelization Methods in the Semi-Lagrangian Advection Scheme." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Mathematics.

Steven M. Serbin

Dr. Steven Serbin, Major Professor

We have read this thesis
and recommend its acceptance:

Charles R. Crabtree

D. Samuel Jordan

Accepted for the Council:

Carolyn Winkel

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Robert N. Stewart

Date 1/4/95

Parallelization Methods in the Semi-Lagrangian Advection Scheme

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Robert N. Stewart

May, 1995

DEDICATION

This thesis is dedicated to my parents, Forrest and Lucy Stewart, and to my newest family - wife Kelly and our baby girl Chelsea.

ACKNOWLEDGEMENTS

I would like to thank Dr. John Drake at Oak Ridge National Laboratories, to whom the direction of this thesis is accredited. I would also like to thank Dr. Steve Serbin, who was officially my thesis advisor, but permitted me the opportunity and convenience of doing my work at Oak Ridge National Laboratories. In addition, I would like to thank Oak Ridge National Laboratories for the use of their facilities, particularly the use of the Intel iPSC/860 parallel computer. I would like to thank my wife, Kelly, for her patience, my parents, Lucy and Forrest, and my sister, Linda, for their unlimited babysitting services for our baby girl, Chelsea, during the final hectic weeks of preparation.

ABSTRACT

The purpose of this thesis was to investigate methods of parallelization in the semi-Lagrangian advection scheme. Semi-Lagrangian advection or transport (SLT) is a numerical scheme for dealing with advective terms often found in systems of partial differential equations. These advective terms occur in various applications including weather prediction and groundwater contamination models. Semi-Lagrangian advection has gained increased attention in the last decade due to its ability to take larger timesteps than Eulerian schemes while maintaining the a regular resolution.

Parallelization of the semi-Lagrangian advection scheme has the potential to further increase the efficiency of the scheme by dividing it into smaller independent sub-tasks that can be performed simultaneously on several processors. Ideally, if the sub-tasks could be evenly assigned to N processors, then the parallel approach should be able to complete the advection N times faster than the sequential method. However, this is usually an unobtainable goal as processor must communicate with each other at certain points in order to complete the total task. This is an additional overhead not needed in a sequential setting which reduces the computational speed up to be less than

N times faster. Since communication is an important factor in computational efficiency, the communication algorithm used becomes important. Two communication algorithms are proposed in this thesis called *process migration* and *data migration.* These two approaches are evaluated and compared under varying circumstances in order to understand the computational behavior of each.

This thesis briefly reviews mathematical transport theory, provides a background for semi-Lagrangian advection, and derives a popular version of SLT known as the three-time-level scheme. Variations of and applications of SLT are discussed. A parallelization of SLT is presented as well as various communication algorithms including data and process migration. Special programming considerations are discussed and a series of tests are conducted on two versions of SLT: the three-time-level and the D_N scheme. These tests serve to evaluate the scalability and performance of the parallelization. The performance of the parallelization depended primarily on the magnitude of the velocity field. The larger the magnitude, the greater the communication requirement. In order to better interpret results, most tests were conducted under a constant velocity field. The results of the tests suggest that the latency period (an initialization period before a message is passed) plays a

significant role in the parallelization. As a result, under the conditions that were tested, data migration was found to be better for the three-time-level scheme and for the D_N scheme until a velocity of about .2. Furthermore, the D_N scheme was found to have a computational advantage over the three-time-level scheme using either communication algorithm. These results should not be extended to non-constant velocity as is illustrated in the final test.

Contents

List of Figures

1 Introduction

It is of interest in fluid flow applications to accurately model material transport. Transport processes refer to the motion of a quantity in a fluid flow. These processes play an important role in many applications from meteorology where large weather systems, such as storm fronts, move with the dominant flow direction to small scale problems such as the motion of pollutant plumes in soil, water or air. Transport processes are referred to with two terms: *convection* and *advection.* Convection refers to circular motions in a fluid caused by thermal differentials. Advection refers to the movement of a mass of fluid that may create changes in the physical properties of the fluid. Throughout the remainder of the paper, the term advection is used to mean either process.

The fundamental equations of fluid flow contain advective terms for the transport of mass, momentum, and energy. The solutions to the flow equations are almost never available in a closed form analytic expression but must be estimated numerically. A central problem in the numerical estimation of these solutions is the treatment of the advective terms. Estimating a flow solution is often impeded by time step restrictions arising from stability con-

siderations associated with these terms. Consequently, it is important to investigate ways of increasing time steps or computational efficiency in order to reduce the execution time of certain models without losing other desirable properties (such as stability).

Numerical methods for treating advection fall into two categories: Eulerian and Lagrangian. From an Eulerian viewpoint, an observer stands at a fixed point and watches the flow evolve around him. The Lagrangian perspective takes the view of an observer moving with the flow. A hybrid of Eulerian and Lagrangian methods known as semi-Lagrangian transport (SLT) has been found to allow for significantly larger time steps while retaining some of the better numerical traits of both the Eulerian and Lagrangian methods for solving advection processes.

This thesis is concerned primarily with decreasing the computation time of semi-Lagrangian advection through parallelization. Emphasis is placed on a popular version of semi-Lagrangian transport known as three-time-level advection. However an alternative semi-Lagrangian scheme, called D_N, is parallelized and compared with the three-time-level scheme. Section 2 provides an overview of mathematical transport theory and introduce the fluid flow equation central to this thesis. Section 3 introduces the three-time-level

SLT prefaced by a review of its predecessors: Eulerian and Lagrangian advection. Section 4 provides a detailed numerical derivation of the equations used in three-time-level SLT. Section 5 discusses the problematic issue of interpolation in SLT. Section 6 provides an analytic stability analysis of SLT for the one dimensional case with constant velocity. Section 7 discusses variations of SLT including the D_N scheme. In section 8, a few of the more prominent applications of SLT are described. Section 9 discusses parallelization issues while section 10 presents an overview of the the actual parallel computer code. Finally, section 11 discusses parallelization results.

2 Mathematics of Physical Transport

A large body of literature on transport theory and fluid flow dynamics exists in mathematical modeling. Here a broad intersection of physics and mathematics occurs with a merger of physical transport processes and dynamical models. A brief statement will be made about those concepts closely related to the particular problem addressed in this thesis. In particular, the mathematical development of the material derivative and conservation equations in fluid flow will be discussed.

Let Ω be a region in R^n with a smooth boundary $\partial\Omega$ where n = 2 or 3. Consider a fluid flow in Ω as a set of particles following a certain trajectory (or path) called the *particle trajectory.* Let $\mathbf{x_0}$ denote the coordinates of a particle at some initial time t = 0. Writing $\mathbf{x}(t) = \phi(\mathbf{x_0}, t)$ to mean this trajectory with the particle's initial position given by $\phi(\mathbf{x_0}, 0) = \mathbf{x_0}$, the mapping $\phi(\cdot, t) : \Omega \rightarrow \Omega$ gives the coordinates, $\mathbf{x(t)}$, of the particles' positions at time t and moves the particle identified with $\mathbf{x_0}$ with the flow. For any fixed t, $\phi(\cdot, t)$ is a coordinate transformation between $\mathbf{x_0}$ and $\mathbf{x}$. We require that the transform be one-to-one and onto thus possessing an inverse. Here, one-to-one implies that no two particles may be at the same place at

the same time. The velocity of the flow at t is defined as

$$\mathbf{v}(\mathbf{x},t) = \frac{d}{dt}\phi(\mathbf{x_0},t). \tag{1}$$

This is the *velocity field* on Ω [4] (Figure 1).

If R is a region in Ω, then let $R_t = \phi(R,t)$ be the volume R moving with the fluid. Suppose now that a function F(x, t) is defined in $\Omega \times T$ where T is the time interval. The relationship between the flow field and the transport of this function is made in the *Transport Theorem* [4].

> *Transport Theorem: Let* $\mathbf{v}$*(*x*, t) be a* C^2 *vector field on* Ω *with flow* $\phi(\mathbf{x},t)$*, and let F(*x*, t) be a function on* $\Omega \times T$*. Assume that* ϕ *is invertible as a function of* x *for a range of t. Then in this range*
>
> $$\frac{d}{dt}\int_{R_t} F(\mathbf{x},t)dx = \int_{R_t}(\frac{\partial F}{\partial t} + v \cdot \nabla F + F\nabla \cdot v)d\mathbf{x}$$
>
> *where R is any subregion of* Ω *with piecewise smooth boundary.*

The conservation of mass for F states that [4]

$$\frac{d}{dt}\int_{R_t} F = 0. \tag{2}$$

Since R_t is an arbitrary volume it follows that if $\frac{d}{dt}\int_{R_t} F = 0$, then

$$\frac{\partial F}{\partial t} + v \cdot \nabla F + F\nabla \cdot v = 0 \tag{3}$$

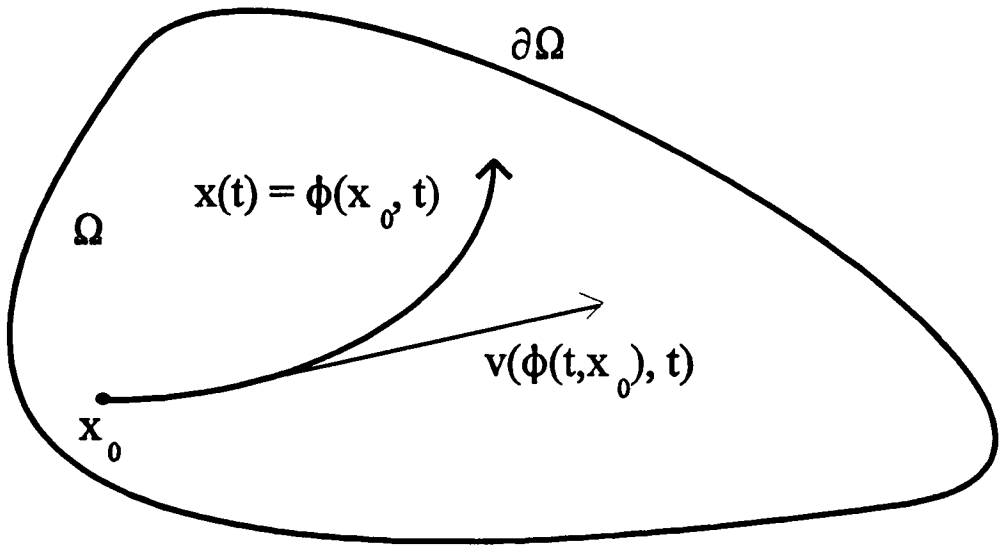

Figure 1: Particle trajectory and velocity.

at each point. If the fluid is incompressible, then $\nabla \cdot v = 0$ [4] and

$$\frac{\partial F}{\partial t} + v \cdot \nabla F = 0. \tag{4}$$

The left-hand side of (4) is the *total time* or *material* derivative of F.

The material derivative represents the change in F as it moves along a trajectory. The derivation of the material derivative comes from application of the chain rule to the total time derivative [15].

> *Chain Rule: If F is a function of* $u_1, u_2..., u_n$ *and each* u_i *is a function of one variable t such that* $\frac{du_i}{dt}$ *is continuous, then F is a function of t and*
>
> $$\frac{dF}{dt} = \frac{\partial F}{\partial u_1}\frac{du_1}{dt} + \frac{\partial F}{\partial u_2}\frac{du_2}{dt} + ... + \frac{\partial F}{\partial u_n}\frac{du_n}{dt}.$$

Setting n = 2, $u_1(t) =$ t, and $u_2(t) = x(t)$ gives

$$\frac{dF}{dt} = \frac{\partial F}{\partial t} + \frac{\partial F}{\partial x}\frac{dx}{dt}. \tag{5}$$

By the definition of velocity, $v(t) = \frac{dx(t)}{dt}$, and extending $x(t)$ into higher dimensions, gives

$$\frac{dF}{dt} = \frac{\partial F}{\partial t} + v \cdot \nabla F. \tag{6}$$

F does not change along trajectories if the material derivative is zero [13] as in (4). In this case, the quantity F is advected by the flow with no diffusion or dispersion.

This thesis deals with the pure advection problem given in (4). In particular, given a velocity field v and initial conditions $F(\mathbf{x}, 0) = F_0(\mathbf{x})$, $F(\mathbf{x}, t)$ is determined as a solution to (4) for all $t \in [0, T]$ and $\mathbf{x} \in \Omega$ by application of the semi-Lagrangian scheme.

3 Overview of Advection Schemes

The semi-Lagrangian advection scheme can be viewed as a hybrid of two well known numerical methods for solving advection problems: Eulerian and Lagrangian advection. Semi-Lagrangian advection has been found to offer advantages over each in solving advection problems [13]. A brief overview of the Eulerian and Lagrangian transport and a detailed explanation of semi-Lagrangian transport follows.

3.1 Eulerian Advection

The Eulerian frame of reference places an observer at a point on a fixed grid [13]. Any time or spatial derivative estimates are made from information around that point only. Consider the one dimensional advection problem given by (4).

The time derivatives $\frac{\partial F}{\partial t}$ can be approximated by finite difference formulas as

$$\frac{\partial F}{\partial t}(x_j, t^n) \approx \frac{F(x_j, t^{n+1}) - F(x_j, t^n)}{\Delta t}, \tag{7}$$

where $\Delta t = t^{n+1} - t^n$. Similarly, using spatially-centered finite differences,

$$\left(v\frac{\partial F}{\partial x}(x_j, t^n)\right) \approx v(x_j, t^n)\frac{F(x_{j+1}, t^n) - F(x_{j-1}, t^n)}{2\Delta x}, \tag{8}$$

where the points $\{x_j\}$ are uniformly spaced and $\Delta x = x_j - x_{j-1}$. Using the standard notation $F_j^n = F(t^n, x_j)$, these difference approximations can be substituted into (4) to obtain the backward Euler method

$$F_j^{n+1} = F_j^n - \frac{\Delta t}{2\Delta x}v_j^n(F_{j+1}^n - F_{j-1}^n), \tag{9}$$

as an example of an Eulerian scheme. Other Eulerian schemes differ only by the approximations used for the derivatives in both time and space. Other examples are One-Sided, Lax-Freidrichs, and Leap Frog. Methods based on an integrated form of (4) have also been developed. The Godonov methods, and their variants such as Total Variation Diminishing (TVD), Essentially Non-Oscillatory (ENO), or Flux Connected Transport (FCT) schemes have received wide application [6].

Explicit-in-time Eulerian advection schemes must adhere to a Courant-Freidrichs-Lewy (CFL) criterion in order to maintain stability. From a practical standpoint this criterion often limits them to stringently small time steps [11]. The CFL criterion involves the concept of domain of dependence. Consider the solution to the advection equation $F(\mathbf{x}, t)$ at any point $(\mathbf{x}, t)$.

The solution at this point depends on the initial data $F(\mathbf{x_0}, 0)$ only at the point $\mathbf{x_0}$ such that $(\mathbf{x}, t)$ lies on the trajectory through $\mathbf{x_0}$. Hence, one could change the initial data at any other points in the domain, and the solution $F(\mathbf{x}, t)$ would not be affected. This singleton $\mathbf{x_0}$ is the *domain of dependence* of the point $(\mathbf{x}, t)$. The numerical domain of dependence, $D_M(\mathbf{x}, t)$ is similarly defined as the set of points that can affect the numerical solution at $(\mathbf{x}, t)$. The CFL criterion is as follows.

> *CFL Criterion: The domain of dependence, $D_M(\mathbf{x}, t)$, of a finite difference method should include the domain of dependence, $D(\mathbf{x}, t)$, of the partial differential equation, i.e.,*
>
> $$D(\mathbf{x}, t) \subset D_M(\mathbf{x}, t)$$
>
> *where M represents the method.*

From this theorem, comes the Courant number derived in [6]. For the backward Euler scheme, the Courant number is $|\frac{v\Delta t}{\Delta x}|$. To insure stability, it is necessary that,

$$|\frac{v\Delta t}{\Delta x}| < 1. \tag{10}$$

Hence for a fine grid, a stringent condition is placed on the time step. Eulerian schemes are particularly inefficient if the maximum time step for stability

reasons is less than the maximum time step for accuracy considerations [6].

3.2 Lagrangian Advection

In a Lagrangian framework, an observer rides on a moving particle and observes the world evolve as he travels. In this approach, the grid points move with the trajectory so that there is no fixed grid. Consider again the one dimensional scalar advection scheme. Let $\{x_j^0\}$ be a set of points in the region at time equal to zero (denoted by superscript 0). The Lagrangian approach is to estimate the solution F by

$$\begin{cases} F(x_j(t^{n+1}), t^{n+1}) = F(x_j(t^n), t^n) \\ \frac{dx_j}{dt} = v(x_j(t^n), t^n) \approx \frac{x_j(t^{n+1}) - x_j(t^n)}{\Delta t}. \end{cases}$$

Here the particle's trajectory is being tracked as the time level and position are changing. The position of a given particle at the beginning of a time step is referred to as the *departure point.* The particle's position at the end of the time interval is the *arrival point.* Since the particle's advection is considered only in terms of the particle path, spatial derivatives are removed and the problem reduces to a set of ordinary differential equations (ODEs) in time. Hence the CFL criteria is not a constraint in the estimation. The only restriction is that the system be integrated with a suitable numerical

ODE scheme. This is an attractive property of the scheme that allows for larger time steps. However, it also suffers from the unfortunate property that initially regularly spaced particles become irregularly spaced (see Figure 2). Depending on the particular flow, the particles may become clumped and require a regridding step. This is often unwanted, as the concept of a fixed grid is natural to many physical problems. In addition, features of the flow may not be well represented [13].

3.3 Semi-Lagrangian Advection

Various versions of semi-Lagrangian advection have been tried in numerical prediction models from as early as 1955. Only in the past decade has semi-Lagrangian transport been demonstrated to have significant advantages [2]. In this approach, particle trajectories are tracked in the flow but at the end of each step, values are returned to the fixed grid. This is achieved by choosing to track those particles which arrive at fixed grid points at the end of the time level. Particles are selected at grid points at the end of the time level (*arrival points*) and their trajectories tracked back in time to the particles' positions at the start time(*departure points*). In particular, one picks a grid

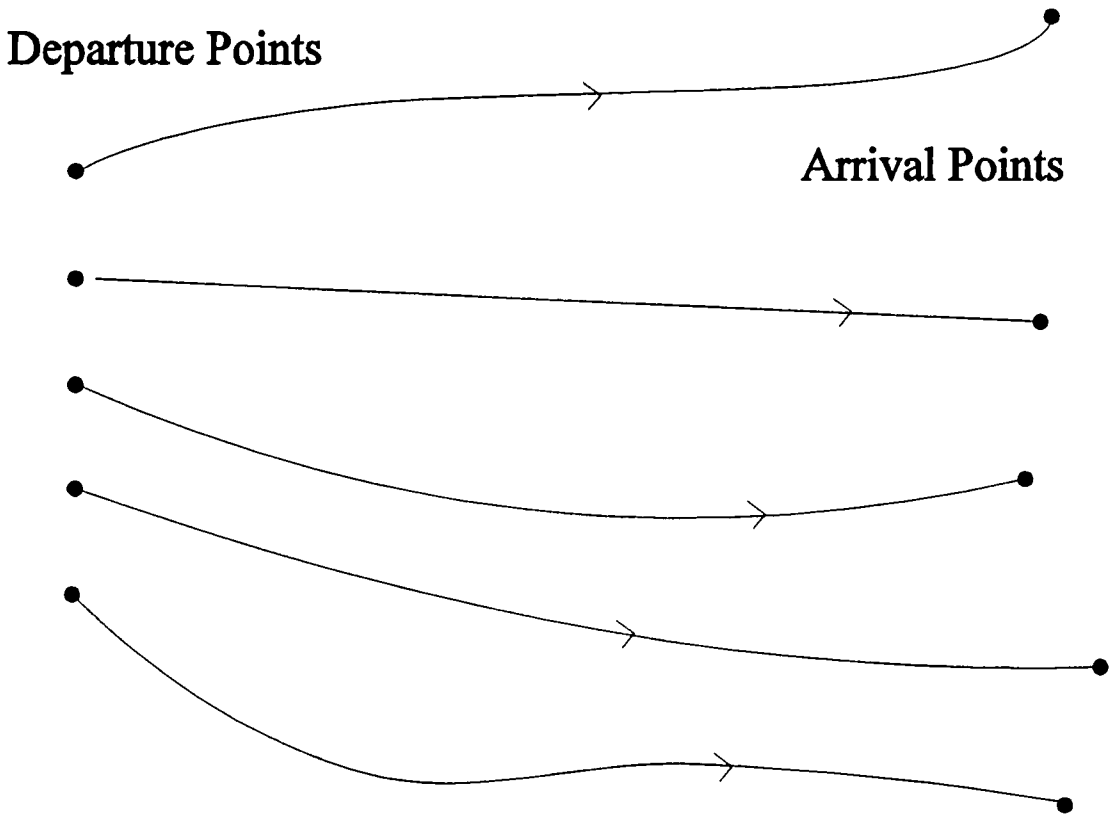

Figure 2: Lagrangian Advection

point at time $t+\Delta t$ and traces the particle's path backwards to the departure point at the t or $t-\Delta t$ time level (Figure 3).

In doing this for each grid point, a set of particles is constructed which arrive at grid points at the end of the time level. Hence, at each time level a new set of particles is chosen instead of one set of particles being followed through the entire procedure. By approaching the problem this way, the regular resolution found in the fixed grids of the Eulerian schemes is acquired while retaining the improved stability of the Lagrangian method [13].

Consider the integration of an exact differential $d\omega$ over a curve C (the differential form $d\omega$ is exact if $d\omega = 0$). The integral of $d\omega$ over a curve from point D to point A of $d\omega$ is independent of path. For example,

$$\int_C dF = \int_C [\frac{\partial F}{\partial t} + v \cdot \nabla F]dt \ = F(A) - F(D) \tag{11}$$

The implications for semi-Lagrangian advection are that any path taken back to the departure point is equivalent to the trajectory taken by the particle. Hence, there are various options for backtracking to the departure point.

For example, consider the trajectory in Figure 4 represented by the solid line where A represents the arrival point, M represents the position of the particle at the time midpoint, and D represents the departure point. The

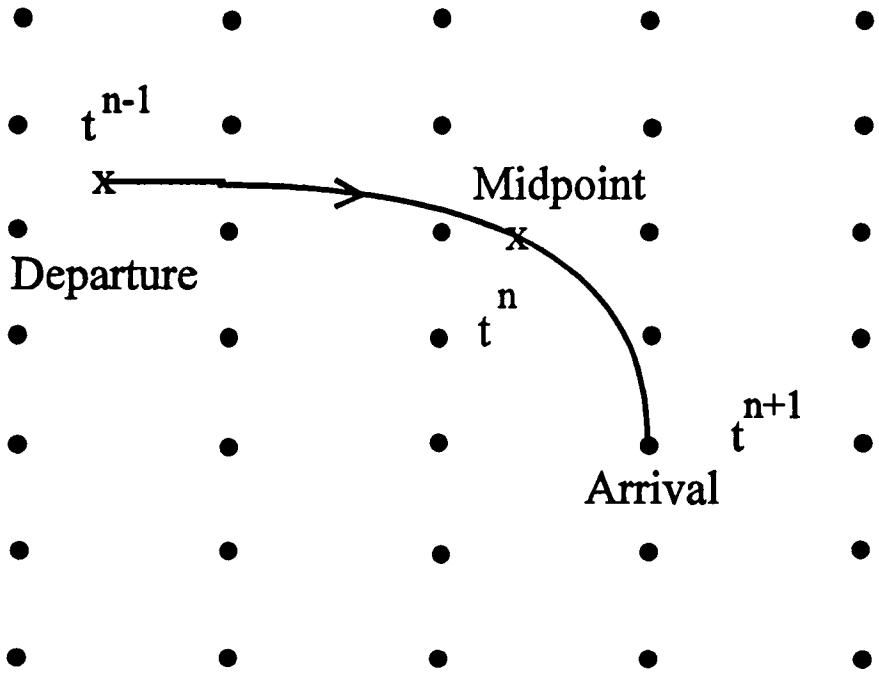

Figure 3: Three-time-level semi-Lagrangian advection.

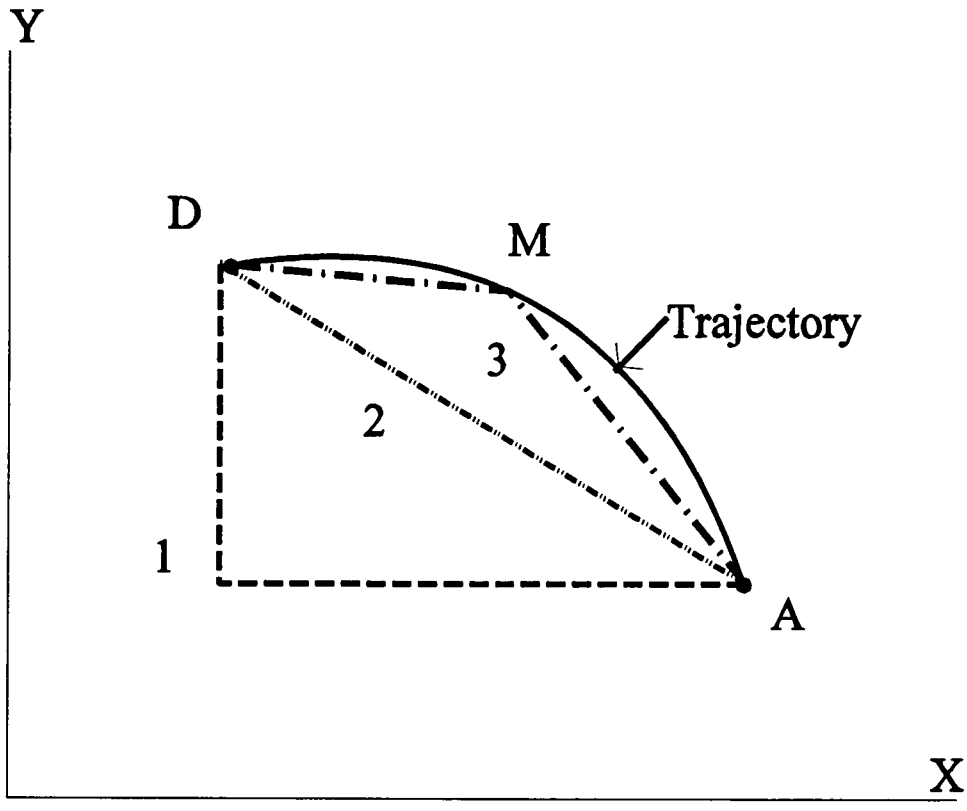

Figure 4: Example particle tracking paths.

choices of paths include tracking in the x direction first and then in the y as in line (1), tracking first to the midpoint and then to the departure point (3), or finally traveling in a straight line to the departure point (2). Two widely used paths in semi-Lagrangian advection are the *two-time-level* and *three-time-level* schemes. This paper concentrates primarily on the three-time-level scheme and also investigates an alternative recently proposed by McGregor [8] called the D_N scheme.

Since particles are tracked from a fixed grid (with information known only at grid points), it is unlikely that the departure point or interim trajectory estimations also fall on a grid point (Figure 3). It is necessary to interpolate the field value (or velocity) at those points not corresponding to grid points. In addition to increasing the overhead in the scheme, the properties of the interpolants are automatically inherited. Such properties include damping, shape deformation, loss of mass, and dispersion (discussed later). In fact, interpolation becomes the central item of concern in this method and a great deal of work in this area has occurred.

4 Numerical Approach

In the three-time-level scheme, the midpoint of the particle trajectory is estimated first. Velocities from the midpoint are then used to compute the departure point. The function F is then interpolated at the departure point x_D and that interpolated value becomes the value of F at the arrival point x_A (Figure 3).

On a two dimensional rectangular grid the scheme is numerically derived as follows. Let the discrete region be defined by points $(x_{ij}, y_{ij}) \in \Omega$. Further let F_{ij} and v_{ij}, respectively, denote the function and velocity values known only at the grid points. Given F at time level t^{n-1} and v at t^n, we seek to determine F^{n+1} at all arrival points (grid points) as a solution to the discrete advection equation.

The material derivative of F is computed using the centered-in-time approximation

$$0 = \frac{dF}{dt} = \frac{F_A^{n+1} - F_D^{n-1}}{2\Delta t} + O(\Delta t^2). \tag{12}$$

where n represents the nth time level, Δt is the time increment, $F_A^{n+1} = F(\mathbf{x_A}, t^{n+1})$, and $F_D^{n-1} = F(\mathbf{x_D}, t^{n-1})$. Ignoring the $O(\Delta t^2)$ terms gives the approximation $F_A^{n+1} = F_D^{n-1}$.

In order to compute the departure point, the midpoint of the trajectory is first computed. In R^2, the position vector is represented by $\mathbf{x(t)} = (x(t), y(t)) = (x, y)$. Writing $\mathbf{v(x}, t) = (\frac{dx}{dt}, \frac{dy}{dt}) = (u, v)$, the time derivatives of x and y at the midpoint (x_M, y_M) are estimated using the following first order accurate finite difference approximations [13]:

$$\frac{x_M - x_A}{\Delta t} = -u(x_M, y_M, t^n),$$

$$\frac{y_M - y_A}{\Delta t} = -v(x_M, y_M, t^n).$$

Estimates for the midpoint the are

$$x_M = x_A - u(x_M, y_M, t^n)\Delta t, \tag{13}$$

$$y_M = y_A - v(x_M, y_M, t^n)\Delta t. \tag{14}$$

This system must be solved iteratively, interpolating using neighboring grid values, the velocity at the newly estimated midpoint, for each iteration. It can be shown that if $\Delta t < [max(|u_x|, |u_y|, |v_x|, |v_y|)]^{-1}$, then these iterations converge [13]. An initial starting value for this iteration is (x_A, y_A) [13]. Since (x_M, y_M) most likely do not fall on a grid point, interpolation of u and v at the midpoint is necessary. Now the departure point may be computed using a centered-in-time approximation with time derivatives of x and y at the

midpoint:

$$\frac{x_A^{n+1} - x_D^{n-1}}{2\Delta t} = u(x_M, y_M, t^n),$$
$$\frac{y_A^{n+1} - y_D^{n-1}}{2\Delta t} = v(x_M, y_M, t^n).$$

It follows that

$$x_D^{n-1} = x_A^{n+1} - 2u(x_M, y_M, t^n)\Delta t, \tag{15}$$

$$y_D^{n-1} = y_A^{n+1} - 2v(x_M, y_M, t^n)\Delta t. \tag{16}$$

$F(x_D, y_D, t^{n-1})$ is then interpolated from values of F at neighboring grid points [13]. This value of F then becomes the value at the arrival point at time t^{n+1}. In doing this for each time level, the scalar function is advected.

5 Interpolation

Trajectory estimates rarely fall on grid points and interpolation is required to estimate F or v. Interpolation occurs at two steps in the three-time-level scheme: (1) in estimating the velocity field during midpoint iterations and (2) in estimating the scalar field at the departure point. In choosing an interpolant, there are two important considerations.

First, interpolation is the single most costly step of the method and overall efficiency depends on the choice of interpolant. In general, the higher the order of interpolation, the higher the computational cost. Therefore, if a lower order interpolant is sufficient, one should avoid higher order methods. Secondly, interpolations can have a direct relationship to the properties of the function advected. The choice of interpolant affects both the mass and shape preservation properties of the advective scheme [12]. The following sections discuss interpolation during midpoint iterations and scalar field estimation and address these considerations.

5.1 Midpoint Iterations

Potentially the greatest computational expense occurs during the midpoint estimation, as each iteration may require an interpolation of the velocity field. Fortunately, the accuracy of the interpolation during midpoint iterations has been found to be far less important than the interpolation of the transport scalar field function. Hence a lower order interpolant is usually sufficient. McDonald [7] has shown that an interpolatant of order one less than the interpolant used for scalar field interpolation is adequate. However, in practice, linear interpolation performs satisfactorily when using cubic interpolations for the scalar field. McDonald showed that it is not theoretically necessary to use the same order of interpolation for each iteration. For example one may alternate between quadratic and linear interpolation during iterations [13]. In this thesis, linear interpolation is used during midpoint estimations.

5.2 Scalar Field Interpolation

The second interpolation step occurs in estimating the scalar field at the departure point. The choice of interpolant is important in this step when one wishes certain properties of the field to be maintained. In practice there are

two major attributes of interest: mass conservation and shape preservation. Mass conservation is preservation of the integral of a function ($\int_\Omega F^{n+1} = \int_\Omega F^n$). Shape preservation is retention of the original form of the function. This original form is often lost to oscillations or damping introduced by the interpolant. Damping increases the smoothness of F giving the same effect as adding a damping term $\varepsilon\nabla^2 F$ to the advection equation and can create overshoots or undershoots of the scalar function.

The accuracy of an interpolatory scheme can be formally determined using Taylor series. This approach provides a bound on the maximum error. However, the accuracy of interpolants for shape preservation is often qualitative. An interpolation scheme is often judged on its visual characteristics, i.e., how well it retains the properties from a physical perspective. From this point of view, there is no best interpolant in a semi-Lagrangian scheme, as the form of interpolation should be matched with the physical problem [7].

5.2.1 Shape Preservation

Preserving the shape of the function is often an important criterion in an advection process. Many standard polynomial interpolations create oscillations in the solution. Shape-preserving interpolants are a class of interpolants that

maintain in the solution any monotonicity, and/or convexity implied by the data. Error analysis may be performed in terms of Taylor series expansions. However, many problems do not have strictly monotonic or convex data and may have discontinuities in the derivatives in their underlying form [16].

There exists a large class of shape preserving schemes. Many of these schemes come from associating interpolants with modified *derivative estimators.* Many interpolants require that the derivative(s) as well as function values be known at grid points. Derivative estimators are methods for estimating derivatives given the data values. In order to insure monotonicity or convexity in interpolation, constraints are imposed on these derivative estimates. In the work done by Williamson and Rasch in [16], this class has been reduced to a relatively smaller class of interpolants that are useful in typical fluid flow applications. In a follow up paper [17], two of the better interpolation methods, in are evaluated for two-dimensional semi-Lagrangian advection. These two are the Hermite and second version rational cubic interpolants with Hyman and Akima derivative estimators.

Consider the one dimensional grid $\{x_i\}_{i=1}^n$ and data $\{F(x_i)\}$. Let the

discrete slope on an interval be defined as

$$\Delta_i = \frac{F_{i+1} - F_i}{x_{i+1} - x_i} \tag{17}$$

The Akima derivative estimator (for evenly spaced data) is

$$d_i = \begin{cases} \frac{\alpha\Delta_{i-1}+\beta\Delta_i}{\alpha+\beta} & if\ \alpha+\beta \neq 0 \\ \frac{\Delta_{i-1}+\Delta_i}{2} & if\ \alpha+\beta = 0 \end{cases}$$

$$\alpha = |\Delta_{i+1} - \Delta_i|,\ \beta = |\Delta_{i-1} - \Delta_{i-2}|$$

and the Hyman derivative estimator is

$$d_i = \frac{\Delta_{i-2} - 7\Delta_{i-1} + 7\Delta_i - \Delta_{i+1}}{12}.$$

From Williamson and Rasch [16], the Hermite and rational cubic are defined as follows. Consider

$$p_i = \frac{P_i(\theta)}{Q_i(\theta)} \tag{18}$$

where p_i is the interpolating function in the interval $[x_i, x_{i+1}]$, $\theta = \frac{x-x_i}{h_i}$ and $h_i = x_{i+1} - x_i$. Let $\theta \in [0,1]$ or equivalently $x_i \leq x \leq x_{i+1}$, and let P_i and Q_i be polynomials. In particular, let

$$P_i(\theta) = F_{i+1}\theta^3 + (r_i F_{i+1} - h_i d_{i+1})\theta^2(1-\theta) + (r_i F_i + h_i d_i)\theta(1-\theta)^2 + F_i(1-\theta)^3$$

and

$$Q_i(\theta) = 1 + (r_i - 3)\theta(1-\theta).$$

If $r_i = 3$, then p_i is the standard Hermite cubic polynomial. If $r_i = 1 + \frac{c_{i+1}}{c_i} + \frac{c_i}{c_{i+1}}$ where $c_i = \Delta_i - d_i$ and $c_{i+1} = d_{i+1} - \Delta_i$, the interpolant is the second version rational cubic polynomial. Certain conditions are classified for preserving monotonicity or convexity. A detailed explanation of these constraints may be found in [16].

Williamson and Rasch investigated combinations of these constraints, interpolants, and derivative estimators were used in semi-Lagrangian advection on a plane and on a sphere with consistent results [16], [17]. Under these conditions, they found the best method was the Hermite cubic with the Akima estimator modified to satisfy C^0 monotonicity. The second best was the rational cubic version 2 with Hyman derivative approximations modified to satisfy C^1 monotonicity. In this thesis, Hermite cubic with Akima derivative estimators are used in scalar field estimation.

5.2.2 Mass Conservation

It is difficult to devise semi-Lagrangian schemes that strictly conserve mass [7]. However, in practice, the meteorological community has found that when using nonshape-preserving polynomial interpolation mass is satisfactorily conserved without enforcement of the conservation property [13].

5.3 Interpolation in Higher Dimensions

Various methods exist for extending interpolation schemes from one dimension to higher dimensions. Williamson and Rasch evaluate three methods for extending these interpolation schemes into higher dimensions for semi-Lagrangian advection: fractional time steps, monotonic bicubic, and a tensor product. Of these three, the tensor product was found to be satisfactory and easiest to implement. The essence of this tensor product is to perform a number of interpolations in one dimension followed by a single interpolation in the second [17]. For example, in Figure 5, to interpolate at the point denoted by **X** with a cubic interpolant, one would perform four interpolations in the x direction (solid lines) interpolating at the four different y values (denoted by *) at **X**. Then interpolate through these interpolants (dashed line) to give the interpolation at **X**.

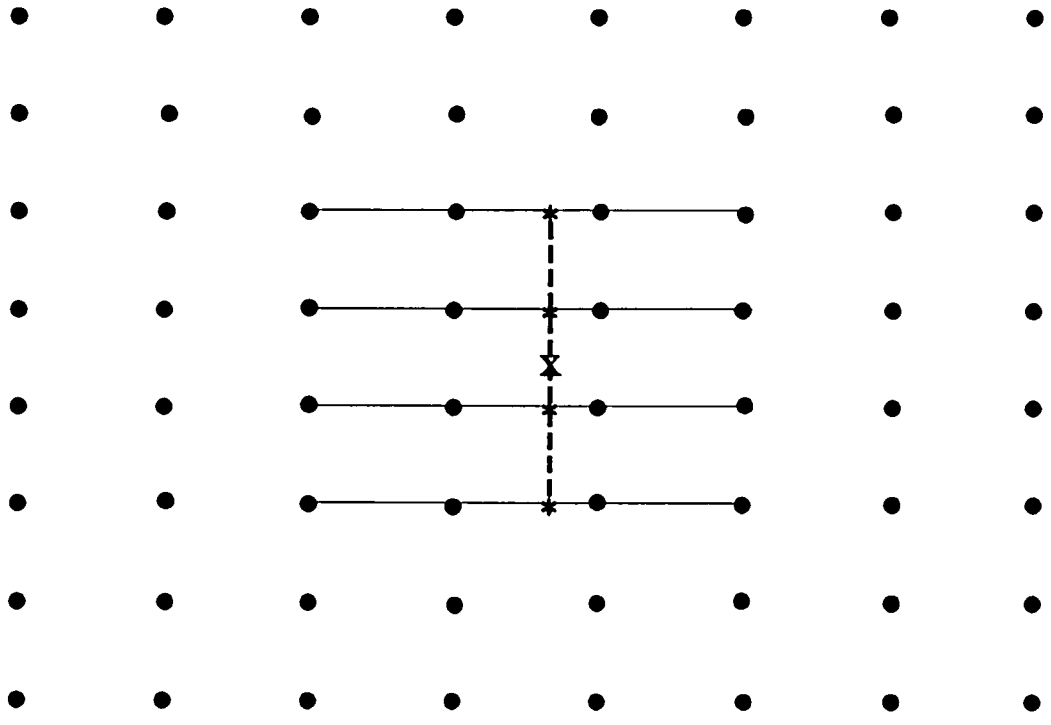

Figure 5: Two dimensional tensor product in a cubic interpolation.

6 Consistency, Stability, and Convergence

An important concept in numerical methods is convergence. Let h and k denote respectively the interval lengths in the spatial and time discretization of a region R. Furthermore let m and n be integers indexing the spatial and time discretization such that $m = n = 0$ is the origin. A finite difference method is said to be *convergent* if at a fixed point (X, T) the numerical solution uniformly approachs the theoretical solution as h,$k \rightarrow 0$ and m, $n \rightarrow \infty$ such that $mh(= X)$ and $nk(= T)$ remain fixed [9]. A method is said to be *consistent* if the finite difference approximation approaches the partial differential equation as $\Delta t, \Delta x \rightarrow 0$. We say that a method is *stable* if the difference in the estimated and theoretical solution remains bounded as the number of timesteps tends to infinity. Convergence is proved by the Lax Equivalence Theorem [6]:

> *Lax Equivalence Theorem: Given a consistent linear finite difference method, stability is necessary and sufficient for convergence.*

Hence proving convergence is reduced to proving consistency and stability of the method. Any argument formed for stability and consistency is dependent on the velocity field and the interpolation scheme used [7]. In this

section, analysis of stability, consistency, and convergence are carried out for one dimensional flow with constant velocity. An analysis for higher order interpolations is similar. The following proof closely follows that of McDonald and Bates [2], [7].

6.1 Notation

On the one dimensional grid write the estimate at the (n + 1)st time level as

$$F(I\Delta x,(n+1)\Delta t) = F(x_*,n\Delta t),$$

where I indexes the spatial discretization with $I\Delta x$ being the arrival point, $x_* = I\Delta x - u\Delta t$ the departure point, and u the constant velocity. $F(x_*,n\Delta t)$ is approximated linearly using the points $(I-p-1)\Delta x$ and $(I-p)\Delta x$, the x coordinates to the left and right respectively of the departure point x_*, where p is the number of grid intervals upstream from the arrival point (Figure 6). Furthermore let F_I^n represent $F(I\Delta x,n\Delta t)$ [7].

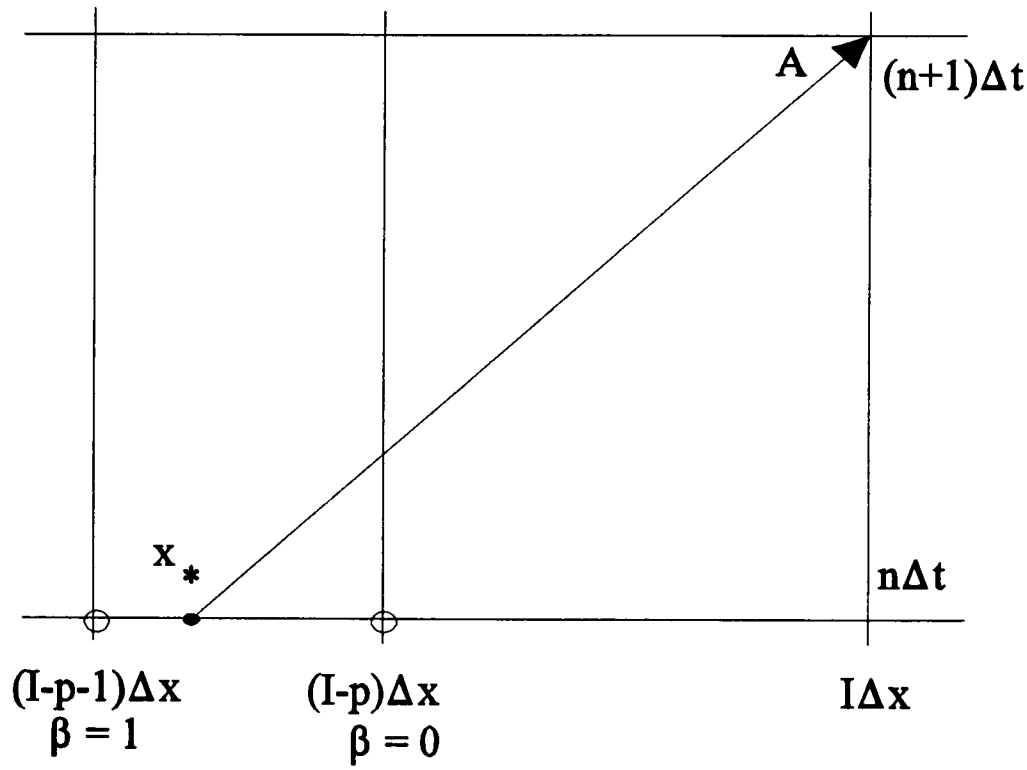

Figure 6: The two circled points are used in the interpolation of F at x_*.

6.2 Stability

The linear interpolation is given by

$$\begin{aligned} F(I\Delta x,(n+1)\Delta t) &= \left\{\frac{F((I-p-1)\Delta x,n\Delta t)-F((I-p)\Delta x,n\Delta t)}{-\Delta x}\right\} \\ &\quad \cdot(x_*-(I-p-1)\Delta x) \\ &\quad +F((I-p-1)\Delta x,n\Delta t) \\ &= F((I-p-1)\Delta x,n\Delta t)\Delta x(\frac{-u\Delta t+p\Delta x}{-\Delta x}) \\ &\quad +F((I-p)\Delta x,n\Delta t)\frac{u\Delta t-p\Delta x-\Delta x}{-\Delta x} \\ &= F((I-p-1)\Delta x,n\Delta t)(u\frac{\Delta t}{\Delta x}-p)+ \\ &\quad F((I-p)\Delta x,n\Delta t)(1+p-u\frac{\Delta t}{\Delta x}) \end{aligned}$$

Letting $\alpha = u\frac{\Delta t}{\Delta x}$ and $\beta = \alpha - p$; then

$$F(I\Delta x,(n+1)\Delta t) = F((I-p-1)\Delta x,n\Delta t)\beta+F((I-p)\Delta x,n\Delta t)(1-\beta). \quad (19)$$

The solution for constant velocity is of the form

$$F_I^n = F_I^0\lambda^n\exp(ikI\Delta x) \quad (20)$$

By the von Neumann method for stability, it follows that the method is stable if $|\lambda^n| \leq 1$ [6]. Substituting this solution into (19) and solving for λ gives

$$\lambda = (1-\beta(1-\exp(-ik\Delta x)))\exp(ipk\Delta x). \quad (21)$$

Since λ is complex, we have

$$\begin{aligned}
|\lambda|^2 &= \lambda\bar{\lambda} \\
&= 1-\beta[1-\exp(-k\Delta x))\exp(-ipk\Delta x) \\
&\quad \cdot\exp(ipk\Delta x)(1-\beta[1-\exp(k\Delta x)]) \\
&= 1-2\beta+2\beta^2-\beta^2(\exp(ik\Delta x) \\
&\quad +\exp(-ik\Delta x)+\beta(\exp(ik\Delta x)+\exp(-ik\Delta x)) \\
&= 1-2\beta+2\beta^2-\beta^2\cos(k\Delta x)+2\beta\cos(\Delta x) \\
&= 1-2\beta(1-\beta)(1-\cos(k\Delta x)) \qquad (22)
\end{aligned}$$

The scheme is stable then if $|\lambda|^2 < 1$ ($|\lambda| < 1$). The smallest value that $1-\cos(k\Delta x)$ achieves is 0 and the inequality holds. The largest value $1-\cos(k\Delta x)$ may attain is 2. In this case, we have

$$|\lambda|^2 = 1-4\beta(1-\beta);$$

then

$$\begin{aligned}
1-4\beta+4\beta^2 &\leq 1, \\
\Longleftrightarrow\ 4\beta^2-4\beta &\leq 0, \\
\Longleftrightarrow\ \beta(1-\beta) &\leq 0. \qquad (23)
\end{aligned}$$

Hence $0 \leq \beta \leq 1$ must hold for stability. This implies that

$$0 \leq \alpha - p \leq 1,$$

$$0 \leq u\frac{\Delta t}{\Delta x} - p \leq 1.$$

Subtracting I, adding p, and multiplying by $-\Delta x$ gives

$$(I - p - 1)\Delta x \leq I\Delta x - u\Delta t \leq (I - p)\Delta x$$

$$(I - p - 1)\Delta x \leq x_* \leq (I - p)\Delta x. \tag{24}$$

Therefore as long as the departure point is between the two coordinates used in the interpolation then the method is stable [2]. Note that this agrees with the CFL condition specified in terms of domain of dependence.

6.3 Consistency

In order to prove consistency, one must show that as $\Delta t, \Delta x \rightarrow 0$ the difference equation

$$\frac{\Delta F}{\Delta t} = \frac{F(I\Delta x, (n+1)\Delta t) - F(x_*, n\Delta t)}{\Delta t} = 0. \tag{25}$$

approaches the advection equation (4). The proof proceeds by writing the Taylor series expansion of the method around the arrival point at time n,

$F(I\Delta x, n\Delta t)$. Then by extracting the terms in the partial differential equation from terms in the expansion, one can show that the finite difference equation is comprised of terms from the partial differential equation and some additional terms:

$$\frac{\Delta F}{\Delta t} = \frac{\partial F}{\partial t} + u\frac{\partial F}{\partial x} + additional\ terms. \tag{26}$$

The objective is to show that as $\Delta t, \Delta x \rightarrow 0$ these additional terms go to zero, i.e., the difference equation approaches the partial differential equation.

Using terms from the linear interpolation to approximate $F(x_*, n\Delta t)$, (25) is written as

$$\begin{aligned}\frac{\Delta F}{\Delta t} &= \frac{1}{\Delta t}(F(I\Delta x,(n+1)\Delta t) - F((I-p-1)\Delta x, n\Delta t)\beta + \\ &\quad F((I-p)\Delta x, n\Delta t)(1-\beta)) = 0. \end{aligned}\tag{27}$$

Writing the Taylor series expansion of these terms around the arrival point at time n gives

$$F(I\Delta x,(n+1)\Delta t) = F_I^n + \Delta t\left\{\frac{\partial F}{\partial t}\right\}_I^n + \sum_{l=2}^{\infty}\frac{\Delta t^l}{l!}\left\{\frac{\partial^l F}{\partial t^l}\right\}_I^n, \tag{28}$$

$$\begin{aligned}F((I-p-1)\Delta x, n\Delta t) &= F_I^n - (p+1)\Delta x\left\{\frac{\partial F}{\partial x}\right\}_I^n + \\ &\quad \sum_{l=2}^{\infty}\frac{(-(p+1)\Delta x)^l}{l!}\left\{\frac{\partial^l F}{\partial x^l}\right\}_I^n, \end{aligned}\tag{29}$$

$$F((I-p)\Delta x, n\Delta t) = F_I^n - p\Delta x\left\{\frac{\partial F}{\partial x}\right\}_I^n + \sum_{l=2}^{\infty}\frac{(-p\Delta x)^l}{l!}\left\{\frac{\partial^l F}{\partial x^l}\right\}_I^n \tag{30}$$

Grouping like terms in the expansion gives

$$
\begin{aligned}
F_I^n &: 1-\beta-(1-\beta)=0 \\
\left\{\frac{\partial F}{\partial t}\right\}_I^n &: 1 \\
\left\{\frac{\partial F}{\partial x}\right\}_I^n &: \beta(p+1)\frac{\Delta x}{\Delta t}+p\frac{\Delta x}{\Delta t}(1-\beta) \\
&= \frac{\Delta x}{\Delta t}(\beta p+\beta+p-p\beta) \\
&= \frac{\Delta x}{\Delta t}(\alpha-p+p) \\
&= \frac{\Delta x}{\Delta t}u\frac{\Delta t}{\Delta x} \\
&= u \\
\sum_{l=2}^{\infty} &: \frac{\Delta t^{l-1}}{l!}\left\{\frac{\partial^l F}{\partial t^l}\right\}_I^n \\
\sum_{l=2}^{\infty} &: \frac{-1}{l!}\frac{\Delta x}{\Delta t}\left\{\frac{\partial^l F}{\partial x^l}\right\}_I^n (\beta(-p-1)^l+(1-\beta)(-p)^l)
\end{aligned}
$$

Regrouping the terms, we have

$$
\frac{\Delta F}{\Delta t}=\left\{\frac{\partial F}{\partial t}\right\}_I^n+u\left\{\frac{\partial F}{\partial x}\right\}_I^n+\sum_{l=2}^{\infty}\frac{1}{l!}\cdot\left[\Delta t^{l-1}\left\{\frac{\partial^l F}{\partial t^l}\right\}_I^n-\frac{\Delta x^l}{\Delta t}\left\{\frac{\partial^l F}{\partial x^l}\right\}_I^n W^l(\beta,p)\right] \tag{31}
$$

where $p=\alpha-\beta$ and $W(\beta,p)=(1-\beta)(-p)^l+\beta(-p-1)^l$

As $\Delta t\rightarrow 0$, the differential system seen in (26) is unaffected, and the first term in the sum $\Delta t^{l-1}\left\{\frac{\partial^l F}{\partial t^l}\right\}_I^n$ goes to zero since $\Delta t^{l-1}\rightarrow 0$. The remaining term involving $W^l(\beta,p)$ requires deeper analysis. In taking the limit as

$\Delta t, \Delta x \to 0$, there are three cases to consider.

Case 1:

The grid is refined so that $\frac{\Delta t}{\Delta x} \to 0$ as $\Delta t \to 0$. In this limit p = 0 and $\alpha = \beta$.

Here

$$\frac{\Delta x^l}{\Delta t} W^l(\alpha, 0) = \frac{\Delta x^l}{\Delta t} - u\frac{\Delta t}{\Delta x} = -u\Delta x^{l-1}.$$

Since $\Delta x \to 0$, this term goes to zero as well and leaves only the terms of the partial differential equation.

Case 2:

In this case, $\frac{\Delta t}{\Delta x} \to \gamma$ a nonzero constant. Here $\alpha \to u\gamma$ and $\beta \to u\gamma - p$. Hence

$$\frac{\Delta x^l}{\Delta t} W^l(\beta, p) \to \frac{\Delta x^l}{\Delta t} W^l(u\gamma - p, p).$$

Using the fact that $\frac{\Delta t}{\Delta x} \to \gamma$, we have $\Delta t \to \gamma \Delta x$. Hence

$$\frac{\Delta x^l}{\Delta t} \to \frac{\Delta x^{l-1}}{\gamma} W^l(u\gamma - p, p).$$

Since $\Delta t \to 0$, then $\Delta x \to 0$ and this term goes to zero. Therefore the method is consistent here.

Case 3:

Here $\frac{\Delta x}{\Delta t} \to 0$ and $\alpha, p \to \infty$, so

$$\frac{\Delta x^l}{\Delta t} W^l(\beta, p) \quad \to \quad \frac{\Delta x^l}{\Delta t}((\alpha - p)(-p-1)^l + (1 + p - \alpha)(-p)^l))$$

$$= (\Delta x^{l-1}u - \frac{\Delta x^l}{\Delta t}(I-i))(-p-1)^l +$$
$$(-u\Delta x^{l-1} + \frac{\Delta x^l}{\Delta t}(i-I))(-p)^l.$$

Since $\frac{\Delta x}{\Delta t} \to 0$ implies that $\Delta x \to 0$, the terms here approach zero and again the method is consistent.

Therefore by the Lax Equivalence Theorem, the semi-Lagrangian method is convergent for linear interpolation. The method is also convergent for higher order interpolations. The arguments are similar and may be found in [7].

7 Variations of the Semi-Lagrangian Method

Variations of the semi-Lagrangian scheme appear in the literature with each having particular advantages and limitations. The more common versions are discussed below.

7.1 Two-Time-Level Scheme

In 1984, Staniforth and Pudykiewics devised a two-time-level scheme from the three-time-level scheme that is potentially twice as fast. If the velocity field v is known both independently and simultaneously with the scalar field F, it is possible to decouple the three-time-level scheme into two independent integrations, one of which advects the scalar field at even time steps by using the velocity field at odd time steps. The other advects F at odd time steps using the velocity field at even time steps. Either of these two may be taken as a solution. It is possible then to estimate the trajectory and leapfrog the scalar field from time t_{n-1} to t_{n+1} without knowing the value of F at t_n, thus halving the computational cost associated with the three-time level scheme (Figure 7).

The important assumption here is that v must be known at the same time

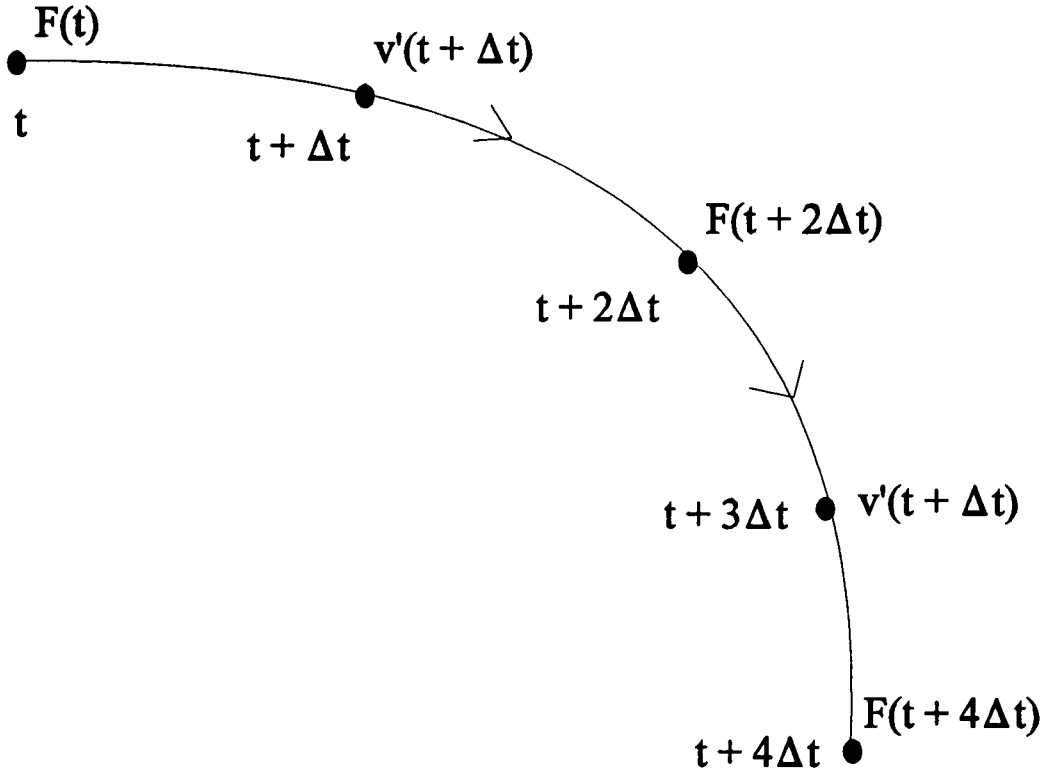

Figure 7: The two-time-level SLT scheme. Note that if $\mathbf{v}$ is known at odd time steps, $\mathbf{v}'$ may be substituted by $\mathbf{v}$

and independently of F. The method breaks down if F = **v**, i.e., the quantity is advecting itself, e.g., momentum equations in fluid-dynamic problems. This deficiency is corrected by extrapolating the velocity to the $t+\Delta t/2$ time level using known winds at t and $t-\Delta t$ using an $O(\Delta t^2)$ extrapolator. Thus the two-time-level algorithm [13] is

$$\frac{F_A - F_D}{\Delta t} = 0, \tag{32}$$

$$\alpha = \mathbf{v}^*(\mathbf{x} - \alpha/2, t + \Delta t/2)\Delta t, \tag{33}$$

where α is the particle displacement and $\mathbf{v}^*$ is the $O(\Delta t^2)$ extrapolator

$$\mathbf{v}^*(\mathbf{x}, t + \Delta t/2) = \frac{3}{2}\mathbf{v}(\mathbf{x}, t) - \frac{1}{2}\mathbf{v}(\mathbf{x}, t - \Delta t) + O(\Delta t^2). \tag{34}$$

7.2 Non-interpolating Method

In an effort to overcome the inefficiency associated with interpolation of the velocity field, Ritchie proposed a non-interpolating version of the semi-Lagrangian method [11]. In this new scheme, the trajectory is considered as the sum of two vectors. The first vector begins at the arrival point (grid point) and ends at the grid point nearest the departure point. This vector is treated in a Lagrangian fashion in section (3.2) The second *residual* vector

would then extend from this grid point to the departure point and is treated in a Eulerian fashion. This approach can be illustrated by considering the one dimensional advection problem illustrated in Figure 8.

In the non-interpolating scheme, v is decomposed into two parts by adding and subtracting $\frac{p\Delta x}{2\Delta t}$, where Δx is the grid length, to obtain

$$\frac{\partial F}{\partial t} + \frac{p\Delta x}{2\Delta t}\frac{\partial F}{\partial x} = -v'\frac{\partial F}{\partial x}, \tag{35}$$

where Δx and Δt are space and time discretizations, and p is the integer nearest $\frac{2v\Delta t}{\Delta x}$ and $v' = v - \frac{p\Delta x}{2\Delta t}$. Applying the centered semi-Lagrangian approach gives

$$\frac{F(x_i, t+\Delta t) - F(x_i - p\Delta x, t-\Delta t)}{2\Delta t} = -v'\frac{\partial F}{\partial x}(x_i - \frac{p\Delta x}{2}, t). \tag{36}$$

Since $x_i - p\Delta x$ is a grid point, no interpolation is required to evaluate the left-hand side. The right-hand side is evaluated at grid points if p is even. If p is not even, then one may evaluate midway between grid points by using a second order space differencing (Eulerian). This method was found to be unconditionally stable and competitive in accuracy with the three-time-level scheme [11].

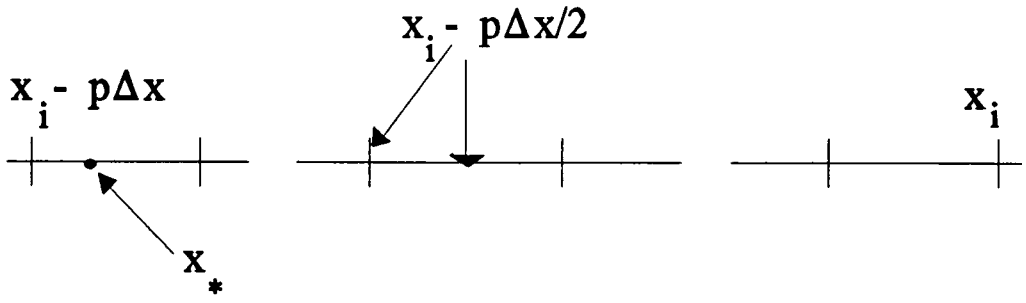

Figure 8: One dimensional non-interpolating semi-Lagrangian scheme.

7.3 D_N Scheme

This scheme, proposed by McGregor in 1993, entirely removes the trajectory estimation from the semi-Lagrangian scheme. The departure point is found by a Taylor series expansion about the arrival point with accuracy depending on the number of terms retained in the series.

In this scheme, a set of position vectors of particles is set up at each arrival grid point at each time level. Let $\mathbf{r}(t + \Delta t)$ denote a member of this set of vectors. The departure point is then expanded in a Taylor series about the arrival point as follows:

$$\mathbf{r}(t) \approx \mathbf{r}(t + \Delta t) + \sum_{n=1}^{N} \frac{(-\Delta t)^n}{n!} \frac{d^n \mathbf{r}}{dt^n}(t + \Delta t) \tag{37}$$

where

$$\frac{d^n \mathbf{r}(t)}{dt^n} = \frac{d}{dt}\left[\frac{d^{n-1}\mathbf{r}(t)}{dt^{n-1}}\right] \quad n = 2, 3, ..., N. \tag{38}$$

The total time derivative (6) is applied to $\mathbf{r}$ giving

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} \cdot \nabla \mathbf{r}.$$

Note that $\frac{\partial \mathbf{r}}{\partial t} = 0$ due to independence of space and time coordinates for partial derivatives. Successively applying the total time derivative to $\mathbf{r}$ gives the set of derivatives in (38).

McGregor argues that if the Eulerian velocities change with time, it becomes difficult to evaluate the partial time derivatives for higher - order terms of $t + \Delta t$. The proposed scheme would approximate the total derivative by

$$\frac{d}{dt} \approx \mathbf{v}' \cdot \nabla \tag{39}$$

where $\mathbf{v}'$ is determined by an extrapolation in time from known velocities at previous steps. The recommended extrapolation [11] is

$$\mathbf{v}' = \frac{1}{8}[15\mathbf{v}(t) - 10\mathbf{v}(t - \Delta t) + 3\mathbf{v}(t - 2\Delta t)] + O(\Delta t^3). \tag{40}$$

Determining departure points by equations (37), (38), and (39) and terms up to the Nth total time derivative is called the D_N scheme. Hence the derivatives of $\mathbf{r}$ at the arrival point can be approximated in the Taylor series giving the position of $\mathbf{r}$ at time t. In this thesis, all derivatives are estimated using Eulerian centered in space approximations. Evaluation of F at the departure point still requires interpolation.

8 Applications

The semi-Lagrangian scheme is applicable to any advection equation that takes the form $\frac{dF}{dt} = R(\mathbf{x}, t, F)$ and an enormous number of illustrations may be found in the literature. A few illustrative areas of applications are now presented.

8.1 Meteorology

Robert is generally regarded as the originator of the SLT scheme. His interest was in its application to weather forecasting. Meteorology continues to be the major area of application. One problem of interest in meteorology is the problem of moisture advection. Moisture fronts are difficult to treat as they often have sharp discontinuities. Ritchie proposed a method for applying the semi-Lagrangian scheme to the moisture equation in a regional forecast model (DRPN, Division de Prevision Numerique) [10].

In the DRPN model, the moisture variable specific humidity, is controlled by advection and horizontal diffusion during each time step. The governing equation for this application is

$$\frac{dF}{dt} = \frac{\partial F}{\partial t} + V \cdot \nabla F = vm\nabla^2 F, \tag{41}$$

where

$$\frac{d}{dt} = \frac{\partial}{\partial t} + m^2(U\frac{\partial}{\partial X} + V\frac{\partial}{\partial Y}) + \sigma'\frac{\partial}{\partial \sigma}, \tag{42}$$

$$(U, V) = (\frac{u}{m}, \frac{v}{m}), \tag{43}$$

where ν is the diffusion coefficient, m is the map-scale factor, (u,v) represents the wind vector and (U,V) is the corresponding wind vector in the model coordinates with its components along the Cartesian coordinates (X,Y) of the projection onto a sphere. The vertical wind component σ' is in the vertical pressure coordinate σ.

The semi-Lagrangian application found in [10] is close to the form (4). The differences are in the addition of a vertical coordinate system and a diffusion term. Ritchie used a Lagrangian cubic interpolation and tested the method against the Eulerian approach by producing a forty -eight hour forecast. In this comparison, he found that the semi-Lagrangian scheme produced improved forecasts with far less dispersion and recommended that all advective terms in the DRPN model would benefit from a SLT approach [10].

8.2 Groundwater

Groundwater flow through a porous medium has received increased attention in recent years due primarily to applications in environmental remediation efforts. The objective here is to model the movement of a contaminant through a groundwater system. The fundamental relationship was established by D'Arcy in the following experiment. Suppose there is a column of sand with water being added at the top. He found that the difference in pressure between the top and the bottom is proportional to the velocity mass flow. For the velocity at a point, D'Arcy proposed

$$\mathbf{v} = -k\nabla p \tag{44}$$

where k is a measure of the porosity and permeability of the medium, and p is pressure. Since there are two unknowns, this equation is usually coupled with an appropriate equation such as the conservation of mass for incompressible fluids (water) [3]:

$$\nabla \cdot v = 0. \tag{45}$$

Applying this to (44) with constant k gives

$$-k\nabla^2 p = 0 \tag{46}$$

A third equation modeling the transport of a contaminant by a flow is $\frac{dF}{dt} = 0$ where F is the concentration of the contaminant. Using the semi-Lagrangian particle tracking approach, one first solves (46) for p and uses the solution to obtain **v**. The velocity is then used to obtain particle positions. One may estimate the total time derivative of F as before as in (12).

8.3 Shallow Water

The equations describing incompressible inviscid flow on a sphere with a free surface are called the shallow water equations:

$$\frac{dU}{dt} + \Phi_x - fV = 0 \tag{47}$$

$$\frac{dV}{dt} + \Phi_y - fU = 0 \tag{48}$$

$$\frac{d\Phi}{dt} + U_x + V_y = 0 \tag{49}$$

where U and V are the velocity components, g is the acceleration due to gravity, $\Phi(= gz)$ is the geopotential height of the free surface of the fluid above a flat bottom, and f is the Coriolis parameter.

These equations are important in atmospheric and oceanic modeling as kernels of general circulation models. Semi-Lagrangian advection has been applied to these equations and a detailed description may be found in [13].

8.4 A survey of particular applications

The following remarks provide a quick overview of some of the applications and successes of SLT in the field and serve to point out literature where detailed explanations may be found.

- A 3D pollutant transport model set in a two-time-level scheme successfully simulated nuclear debris from the Chernobyl reactor accident [13].

- This scheme serves now in Canada's Environmental Emergency Response Model [13].

- Semi-Lagrangian advection was coupled with a split-explicit scheme to integrate a multiple primitive equations model currently used by Ireland to produce weather forecasts on a routine basis [13].

- Robert, Yee, and Ritchie in 1985 proposed a method of integrating the semi-Lagrangian technique into multi-level atmospheric models. The semi-Lagrangian method served to provide an increased amount of stability and the primitive equations where integrated at a high level of resolution [12].

- Sawyer (1963) applied this scheme to the barotropic vorticity equation and found similar results obtained by Eulerian methods without the instability associated with the CFL criterion [10].

9 Parallelization

Parallelization is the process of dividing a large task into smaller independent tasks which can be performed simultaneously. Not all tasks are parallelizable as they are not composed of independent sub-tasks. Parallelization of computational tasks provides two important advantages over sequential ordering: (1) Potential for linear increase in computational speed and (2) increase in memory capacity.

If a process may be divided uniformly among N processors, ideally the process could be completed N times faster than on a single processor. This is referred to as a linear increase in speed. Unfortunately, for many applications this is an unobtainable speed. Communication requirements among processors are the source of inefficiency not present in sequential execution. This problem is attributed to communication requirements among processors not needed on a single CPU. As one would expect, N processors can hold N times more information than a single processor. This is potentially a tremendous advantage especially when sub-tasks do not require global information. When parallelizing a task, issues in efficient interprocessor communication and memory utilization must be addressed in order to reap the full benefit

of parallelization.

Semi-Lagrangian advection lends itself well to parallelization. Since trajectory estimations are independent, separate processors may carry out advection for each arrival point simultaneously. One may divide the global domain among processors, with each programmed to carry out advection for every grid point in its local domain (Figure 9). At the end of every time step, each processor's field is updated to the new time level and the calculation is ready for the next timestep. Given a field of velocities of sufficiently large magnitude, the trajectory estimations may go outside of a processor's local domain. If an estimate goes out of a processors range the point is said to be *off processor.* When each processor knows the velocity and scalar field over the global range (*universal or shared memory*) the calculation of departure points may proceed. But for dense grids, the amount of memory available may be insufficient for each processor to have a copy. Instead processors communicate between each other in order to complete the task. This communication is important as it can lead to less than a linear speed-up in computational speed.

Communication between processors is performed by message passing. A message is simply a data stream. Sending and receiving messages has an as-

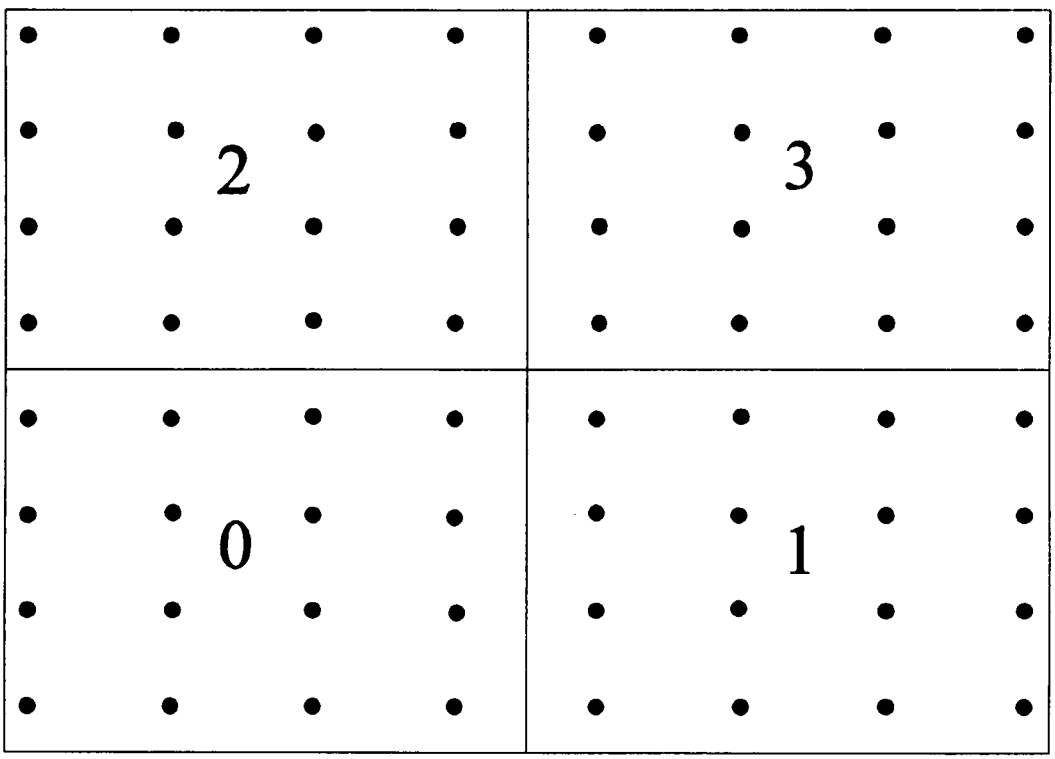

Figure 9: Parallelization of the Euclidean grid.

sociated cost as each message requires a connection and delivery wait time. When a processor sends a message to another processor, it must first inquire of the receiving unit whether it can accept the message. The receiving unit must respond before the message is passed. This period of time before the information is passed is referred to as the *latency period* and can potentially be a large overhead in computation time. The information transfer requires time as well. A communication algorithm may be inefficient if it has a large number of messages with very little information in each pass. Here latency overhead becomes dominant. Few message passes with enormous and potentially unnecessary blocks of information is another source of inefficiecy in parallel algorithms. In this case, the data transfer may become dominant. Two communication algorithms are studied in this thesis for semi-Lagrangian transport which are motivated by these considerations: *process migration* and *data migration.*

9.1 Process Migration

In process migration, when an estimate goes off processor, it is sent to the processor capable of continuing the task (Figure 10). When a processor fin-

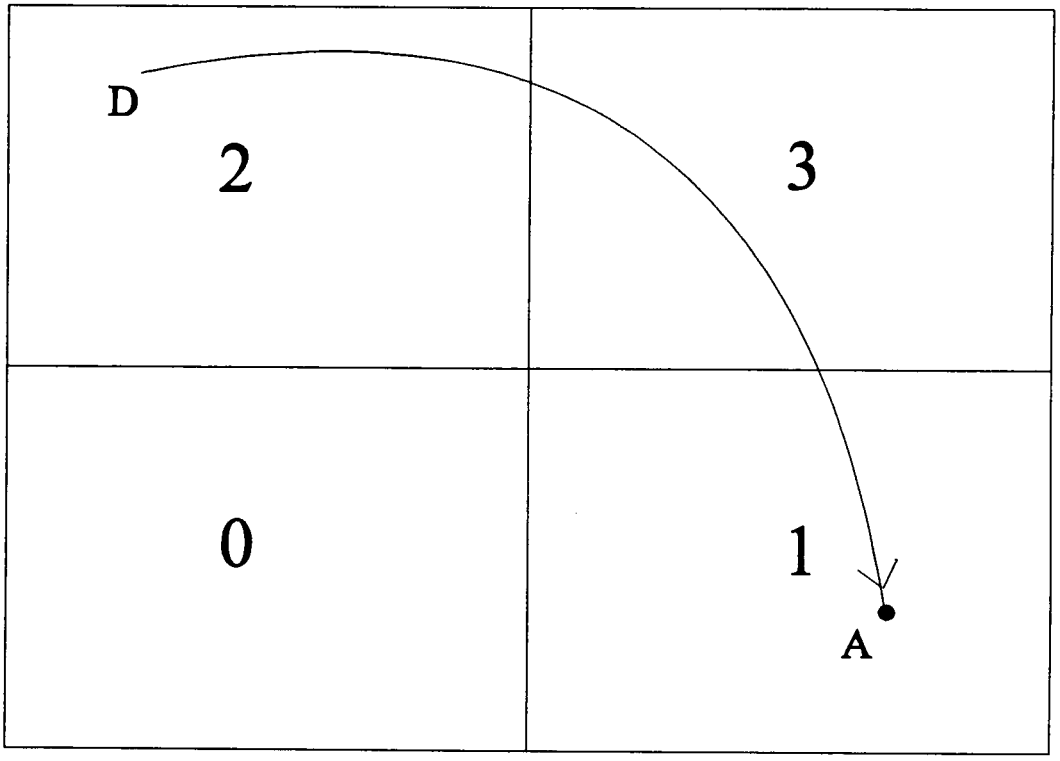

Figure 10: A particle trajectory which traverses three local processor domains.

ishes another's task it returns the result to the originating processor. A general structure must be formed for sending and receiving tasks. For the three-time-level scheme the proposed method is to recognize a timestep as occurring in two parts: (1) each processor performs the three-time-level algorithm as far as possible and if unable to complete, it then passes the task to the appropriate processor and (2) enters a working and listening stage where tasks requested by other processors are continued. Within this general framework, there are several options for use of the stages.

One option is to implement the two stages for midpoint calculations only and then *synchronize* the processors before continuing the departure point calculations. Synchronization is a waiting point where nothing else occurs until all processors have midpoint estimations. This is then repeated for departure point and scalar field estimation. Because of the synchronization this method is easier to code but can potentially be inefficient as some processors may finish all tasks quickly and be forced to wait on others before proceeding to the next step.

A more efficient but complex algorithm employs a prioritization scheme. The particular scheme proposed here is as follows. Each processor calculates a trajectory as far as possible and then passes the information to the appro-

priate processor with a flag denoting what stage the estimation has achieved (midpoint, departure point, or function estimation). After processing its local work, each processor goes into a listening and working mode where it probes for tasks sent by other processors. It accepts tasks in this order: midpoint estimates, departure point estimates, scalar field interpolation, and recovery of its own trajectory results. By processing in this manner, the first actions of estimation are given priority so that work on other processors is not blocked and all estimations continue in a loosely synchronized manner. This loose synchronization reduces wait times as the flow of tasks usually leaves work for any given processor to complete. A synchronization occurs at the end of every time step just after the updating of the scalar fields. For the D_N scheme process migration is simplified as no trajectories are being estimated. The D_N scheme estimates departure points with limited overlapping required for derivative estimation. The D_N scheme does not use process migration until estimating F at the departure points. Hence no prioritization scheme is needed other than to accept function evaluation tasks before recovering results.

Another level of sophistication in process migration is complete task deferral for overworked processors to prevent *load imbalance.* Here load imbal-

ance refers to the situation where certain processors are performing a greater number of trajectory or field estimations while others do nothing. A worst case scenario occurs if a source and sink velocity field exists (Figure 11). The source originates in processor 0 with all trajectories ending in processor 3. In this case, processor 0 is required to estimate nearly all components of the other processor trajectories. Hence the process approaches a sequential algorithm as only one processor is performing nearly all the work.

A task deferral scheme would identify those processors which haven't worked in two or three steps and send unfinished work from overloaded processors. This process can be closely akin to data migration (discussed next) as large blocks of redundant information exist. In addition, while the concept is intuitive, the coding is more complex and lends itself to a code optimization problem. Hence, only the first prioritization approach is used, as the others are only modifications with the same general concept.

9.2 Data Migration

In data migration, the data necessary to continue estimation for those points off processor is made available to the processor by its neighbors. An effi-

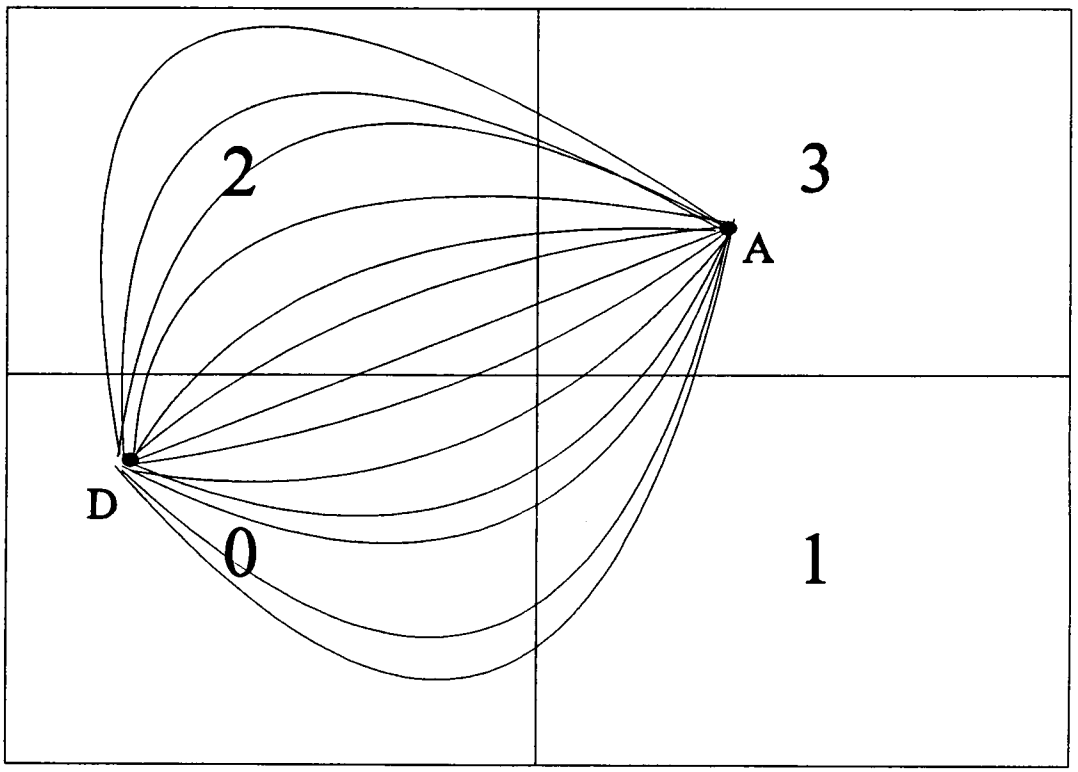

Figure 11: Source and sink flow pattern. The source is at the departure point D and the sink is at arrival point A.

cient form of data migration, is to send all the information each processor needs at the beginning of each time step. For each processor, this is accomplished as overlapping or information extensions to its local domain sufficient to complete the trajectory estimation for each local grid point (Figure 12). This approach has the potential to produce few messages but with excessive amounts of information. An example of excessive data transfer occurs when the processor only needs additional data to its left but receives information on all sides. It is however intuitive, easy to code and may be the clear choice for several situations. The problem is often dependent on the velocity field. If the field is fast moving with a single trajectory estimate passing over multiple processor domains, data migration may not be viable as excessively large blocks of information would be passed and stored redundantly in memory. However, if the velocity field is relatively slow and the amount of memory is sufficient to hold extra information, it could be the better choice.

Another form of data migration that does not use this overlapping approach. Instead of overlapping information at the beginning of the process, each processor only inquires about the precise information needed to complete a particular trajectory estimation and only at the time it is needed. Hence each time an estimate goes off processor, that processor requests only

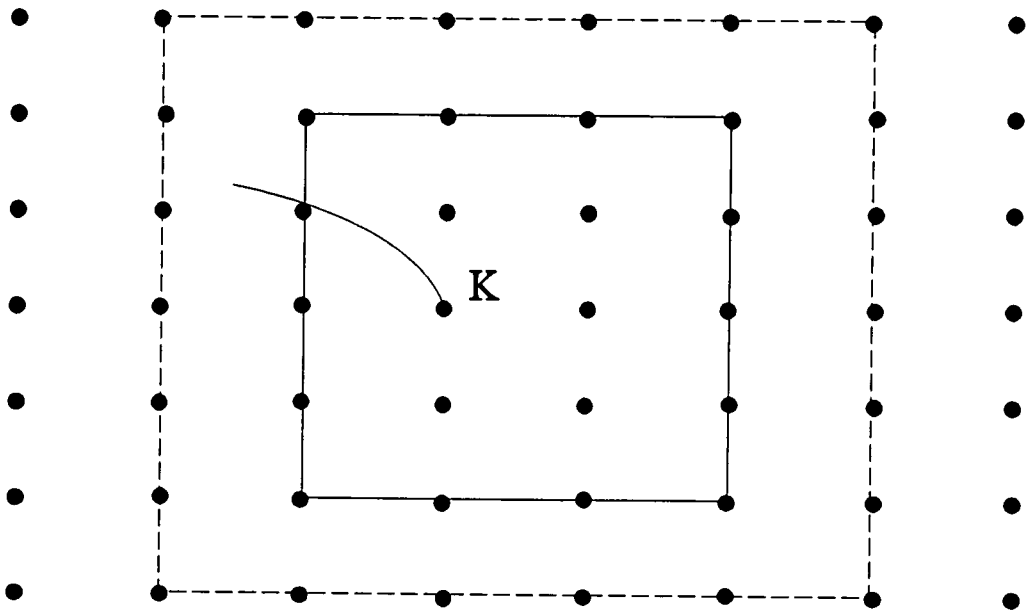

Figure 12: The solid boundary represents the boundary defined in the parallelization. The dashed boundary represents a single overlap of additional information. With the additional overlap, this trajectory can be fully evaluated by processor K.

the information needed to continue estimation. This method is equivalent to process migration in terms of the number of messages sent but usually has much longer messages. It is therefore not as efficient and is not considered further.

9.3 Hybrid Migration

A hybrid form of data and process migration schemes is also considered. In this form, process migration occurs on an extended informational domain. By extending the information a little further, it may be possible to capture a large number of departure points without passing any further messages. Those that do go off the extended domain are continued in process migration (Figure 13). This form would be expected to be ideal for situations where a few departure points are significantly farther away while most are near the processor domain. The number of overlaps used depends on the velocity. Ideally, one would stop at the number of overlaps that produce a minimum in the message passing network. This is *a priori* problem dependent and is often difficult to estimate. In this thesis, all three forms (data, process, hybrid) are investigated and results are compared.

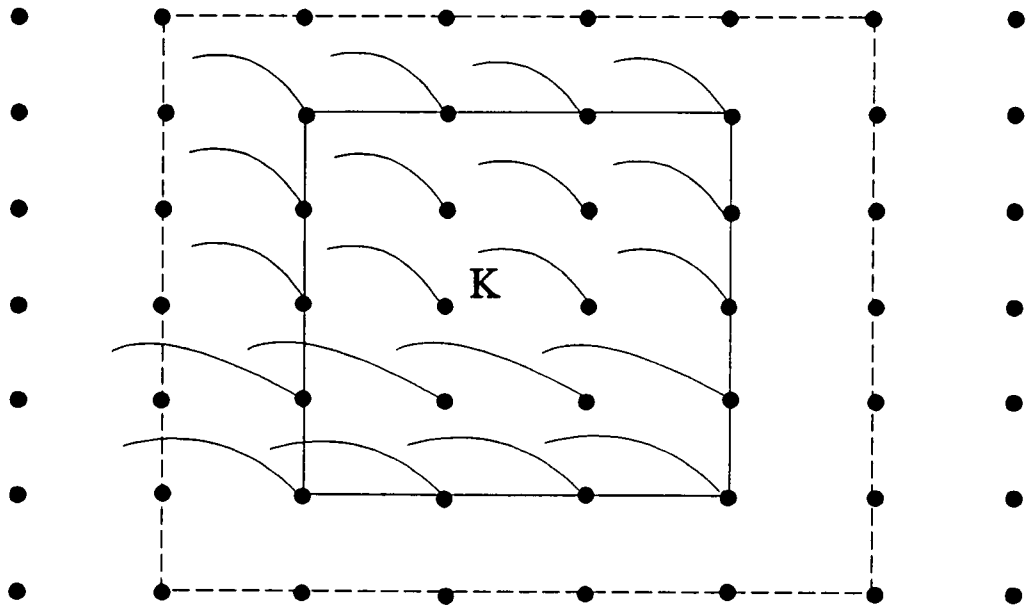

Figure 13: In hybrid migration, informational extensions capture more trajectories that go off processor, while process migration controls those that extend beyond the overlaps.

10 Details of the Parallel Code

10.1 Definitions and Initial Setup

The code was developed to implement SLT in numerically solving (4) on a doubly periodic Euclidean grid. The three-time-level SLT was developed first and was later modified to implement the D_N scheme.

The user specifies the domain division by specifying the X number of processors (number of processors in the horizontal direction), the Y number of processors, the global number of x data nodes, and the global number of y data nodes. The range in both directions is always considered to be one. The domain is constructed so that the first row of y nodes and first column of x nodes are at the zero position. Since the grid is doubly periodic, 1 and 0 are considered to be the same coordinate. Hence the last row of y nodes and column of x nodes are always the last discretization before one (Figure 14). The global grid discretization is the only shared memory between the processors. This arrangement is necessary in order to determine which processor should finish the task or from where the data should come. The user also specifies the initial function and velocity values.

The processors are assigned integer identification numbers (id) beginning

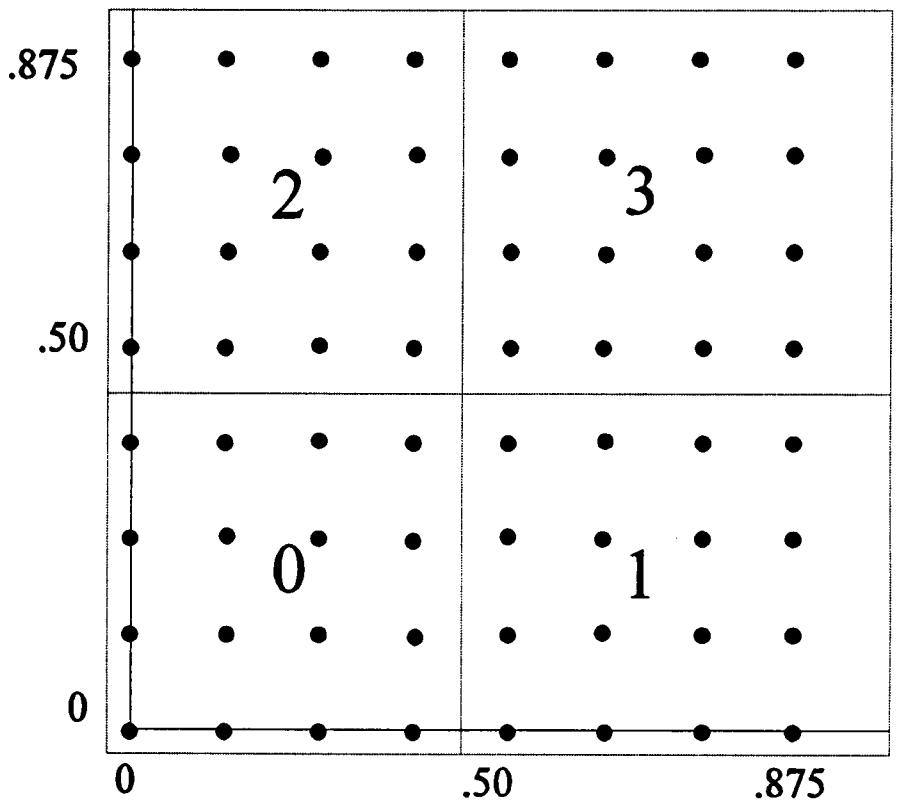

Figure 14: Periodic Euclidean grid and processor assignments. Fine dashed lines represent local processor domains. Solid lines represent x and y axes.

at zero and increasing by one until all processors are labeled. The processors are assigned local domains by increasing id, beginning in the lower left hand corner and increasing first to the right then up. Each processor knows its neighbors by their id number. Figure 14 illustrates these discretizations. Before presenting an in depth overview of the code procedures, a few of the issues that are technically important are discussed: necessary overlap, interpolation overlap, and periodicity.

Local domains are based on grid points. In particular, a local domain in the x direction is specified by assigning the processor the jth column of nodes through the (j+k)th column of nodes. Regardless of the stencil chosen, a small amount of overlap is necessary. In particular, an overlap of one is necessary to prevent gapping between processors. Suppose that this overlap is not used as in Figure 15. Furthermore suppose that an estimate (midpoint or departure) c, is located in one of the gap regions. The responsible processor, say the one to the left, determines that the estimate is off processor and attempts to send it to a right-hand neighbor. It then scans all of its right-hand neighbors' domains and discover that it is not within any of their domains; this situation results in a breakdown in the algorithm. Hence, an overlap of one on all sides of a processor is necessary. The processor is not

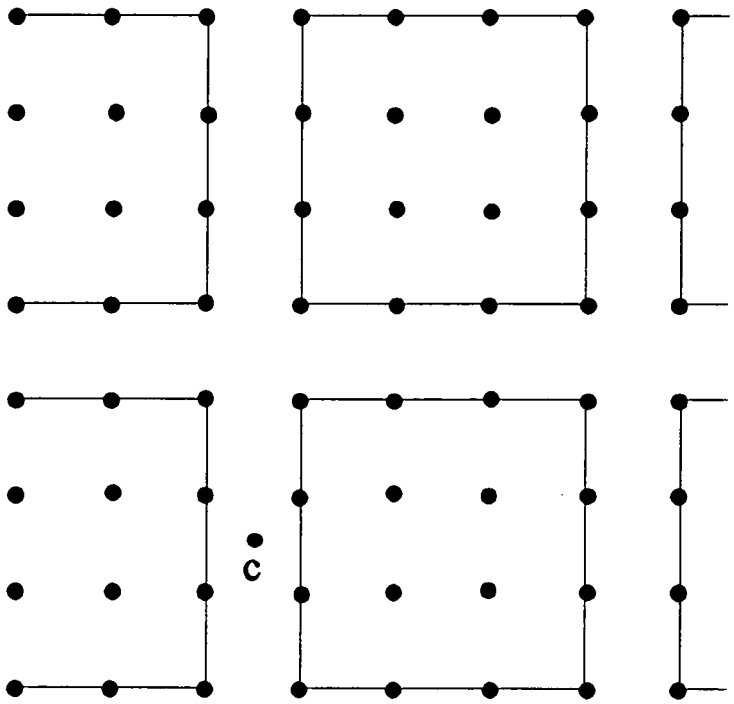

Figure 15: Parallelization without an overlap of one. Point c is an estimate in the semi-Lagrangian algorithm.

responsible for trajectories associated with those additional nodes, but continues to process estimates that fall in the area just short of them. When using the hybrid approach this boundary may be larger than one and is the component equivalent of the data migration scheme.

When using an interpolation method, a neighborhood of data points around the interpolation point is necessary to produce the estimate. The extent of this neighborhood depends on the interpolant chosen. For those estimates that fall sufficiently close to the local domain borders, additional data off processor is required to produce the interpolation. Hence beforehand, the necessary overlap to produce an interpolation for any point in the responsible domain is acquired by each processor. For example, an overlap of three nodes on all sides is required to use a cubic Hermite interpolation for the scalar field.

In summary, there are three boundaries to consider. The inner boundary bounds the number of nodes the processor is responsible for advecting. The secondary boundary exists to define the region in which trajectory estimates are continued to be processed (must be at least one overlap to prevent gapping). The last boundary is specified by the requirements of the interpolation scheme. These three boundaries are illustrated for a single processor in Fig-

ure 16. For the D_N scheme an additional boundary is required to specify the overlaps required in estimating the derivatives for arrival points along domain boundaries.

The global domain is periodic in both the horizontal and vertical directions. Hence when an estimate is greater than or equal to one, the true estimate is found by subtracting the whole number value from the estimate to return the value to the global domain of [0,1). For example, if the departure point x estimate is calculated as 2.3, then subtracting 2 returns it to its proper value of .3. An additional level of coding is necessary as the mathematics of the iteration and interpolation schemes has nothing to do with this periodicity. Consider the midpoint iterations. Suppose that before the midpoint can successfully converge, the estimate goes off the local domain. If the modified estimate is used in the estimation, then convergence may never be obtained as the difference in the previous estimate and the new estimate is at least greater than one. Hence, there is a switch in the information about the velocity field. The code uses the non-periodic estimate in each of its computations but must modify the estimate when determining which processor to send it to, what the appropriate scalar field values are, and in reporting any final estimate.

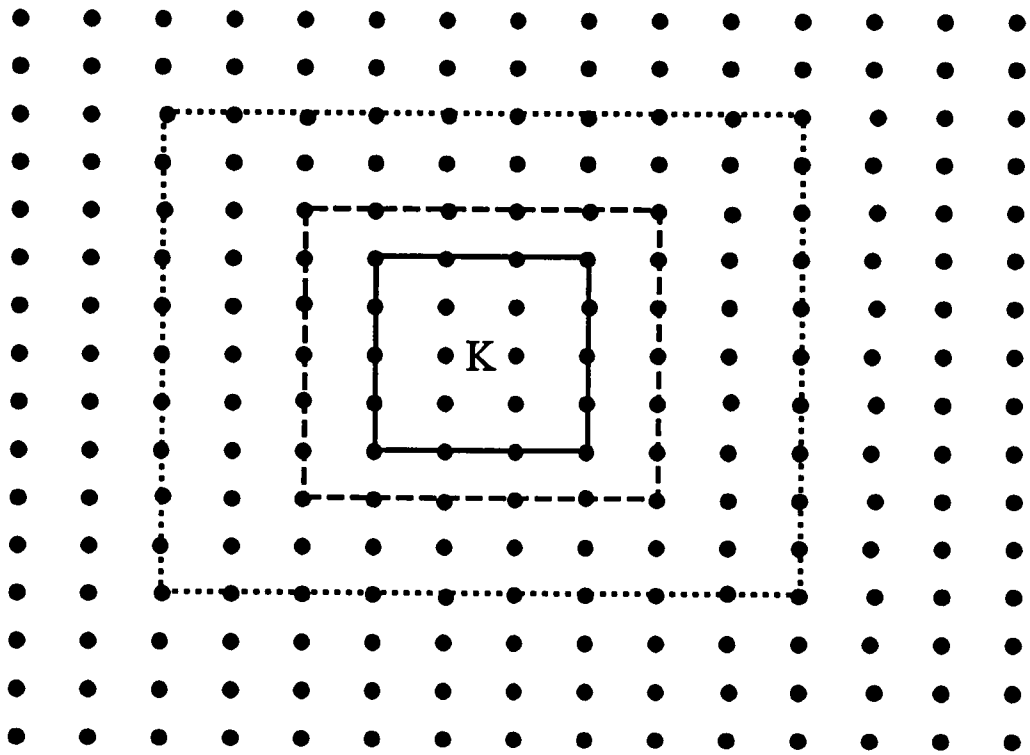

Figure 16: Three forms of overlap: The *responsible* boundary is represented by the solid lines, the *informational overlap* boundary by the dashed lines, and the *interpolation overlap* boundary by the finely dashed lines.

In summary, the first step of the code obtains any initial information needed. The user specifies the parameters of the discretization, time step, number of steps, velocity field, scalar field, number of processors, and communication stencil. In the D_N scheme one must also specify N. After these inputs, the code sets initial variables dependent on this information such as processor identification numbers.

10.2 Process Migration

In process migration, message passing is more intense and a clean criteria for sending, receiving, and processing information is needed. Appropriate synchronization of the processors becomes important as well as reducing wait time.

In this code, process migration is conducted as a prioritization scheme. The process is considered as two parts: (1) evaluation of a processor's own trajectories, and (2) working on external tasks and recovery of the processor's own estimates. The driver for this method is subroutine *Work*. Subroutine *Work* accepts a task along with a flag denoting where the task is in the algorithm. It then proceeds until completion of advection or until the estimate

has gone off processor. It then determines where to send it and passes it with an updated flag.

In the first stage, every processor passes each of its arrival points to *Work* to begin the SLT algorithm. If particle tracking is successful within a processor's domain, *Work* returns the result to the main code with a flag to this effect. If an estimate goes off processor, *Work* determines which processor should finish the task, passes the task, and returns control to the main code.

After processing each grid point this way, the processor goes into the second mode. The processor enters a loop where it checks for the following in this order:

1. Determines if each processor is done with the current timestep,
2. Probes for midpoint work,
3. Probes for departure point work,
4. Probes for scalar field estimation work,
5. Probes for the return of work it sent out,

6. If all of its own trajectories are completed, sends a message to processor zero to that effect,

7. Returns to 1.

With the exception of 1, if a given checkpoint is true, the processor performs associated tasks and returns to 1. If 1 has occurred, the processors write out any information requested about that timestep, update fields, obey a synchronization command and then return to the first stage to begin the next timestep.

Processor zero has special duties. It serves as the organizing processor to the other nodes by collecting information on when timesteps are done, broadcasting when the next timestep has begun, and recording any test information produced. Hence processor zero has an additional check point before number one above: 0. Determines whether each processor, including itself, is finished and if so, broadcasts the message to each processor.

10.3 Data Migration

Data migration is the more intuitive of the two opposing methods to code. In fact, data migration can be viewed as a special case of process migration

from a programming perspective. If the user selects data migration, a sufficient overlapping occurs and the second stage is never entered. All tasks are completed within stage 1 with an error message occurring if an estimate goes off processor. In the hybrid approach, the user chooses process migration but specifies a larger overlap than one.

10.4 D_N

The D_N scheme was programmed by modifying the three-time-level scheme. Since D_N estimates departure points via Taylor series expansions, this task is implemented first for every arrival point in each processor domain. This task requires no communication other than overlaps needed to estimate the higher order derivatives of $\mathbf{r}$. Then, stages one and two of the three-time-level scheme are implemented for scalar field estimations only.

11 Test Results

11.1 Test Goals

The primary goal in the following tests is to investigate computational efficiency within the parallel environment for the three-time-level scheme and to compare with the D_N scheme. In particular, the following issues are addressed:

- Evaluate the computational speed-up as the number of processors is increased.
- Determine behavior as informational overlapping increases from process migration to complete overlap and ascertain the relative success of the three migration schemes.
- Investigate optimal overlapping as the magnitude of the velocity field increases.
- Compare the three-time-level scheme with the D_N scheme for N = 1 and 2 using optimal overlapping for varying velocity fields.
- Determine a computational performance model for each method.

- Determine the scalability of each method.

11.2 Test Problem

The following defines the test problem used throughout. The function F is a cylinder centered initially at (.2,.5) with radius and height .1 (Figure 17). The global domain is discretized as a 32 x 32 mesh (unless otherwise specified). Furthermore to produce clearer more interpretable results the velocity is always constant across the domain. Using non-constant velocities adds further complexity to the results and is briefly discussed at the end. Since the velocity is constant, any value of N greater than one in the D_N scheme does not produce a more accurate result, as higher order derivatives are zero. However, larger values of N are used here to determine computation times related to the additional derivative estimates. All testing was conducted on the iPSC/860 parallel machine.

11.3 Error

A great deal of literature that addresses errors can be found [8]. Included for illustration only is a graph of the cylinder with velocity $(u, v) = (.1, 0)$ which

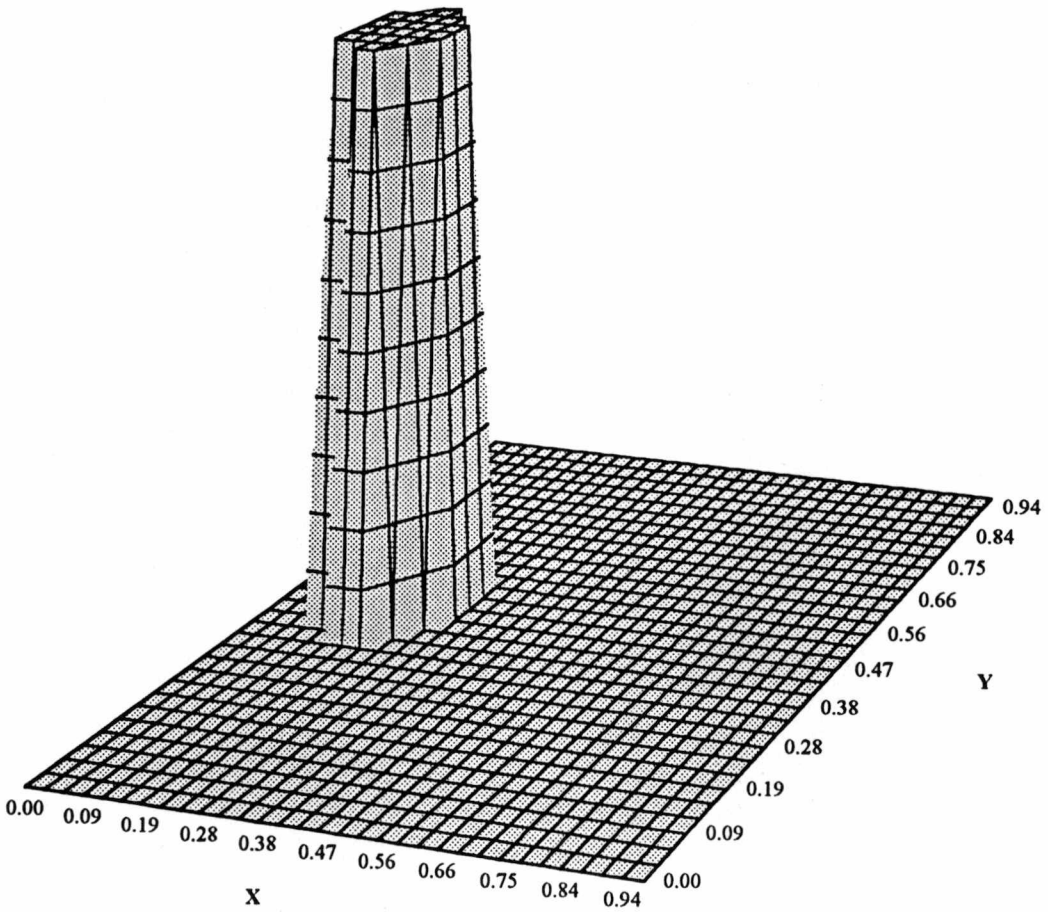

Figure 17: Cylinder position and shape at t = 0.

returned it to its original position after 50 time steps (Figure 18). This result was the same for the D_N scheme (true only for constant velocity). The associated smoothing and overshooting associated with polynomial interpolation is consistent with the results in the literature [13], [16] for Hermite cubic interpolation.

11.4 Parallelization and Computational Efficiency

A driving motivation for parallelization of any process is to decrease execution time. It would be ideal if the speed-up in computational efficiency were equal to the number of processors involved. However, due to communications between processors, this goal is usually unobtainable. In the semi-Lagrangian advection scheme, communication between processors is controlled primarily by the velocity field. Faster velocity fields can move particles over multiple processor domains during a single time step increasing communication requirements. Therefore in addressing the issue of computational speed-up it is necessary to consider improvements in light of the velocity field. In order to illustrate this property, a 2×2 and a 4×4 processor grid advected the cylinder for various velocities in the positive x direction for a single time step

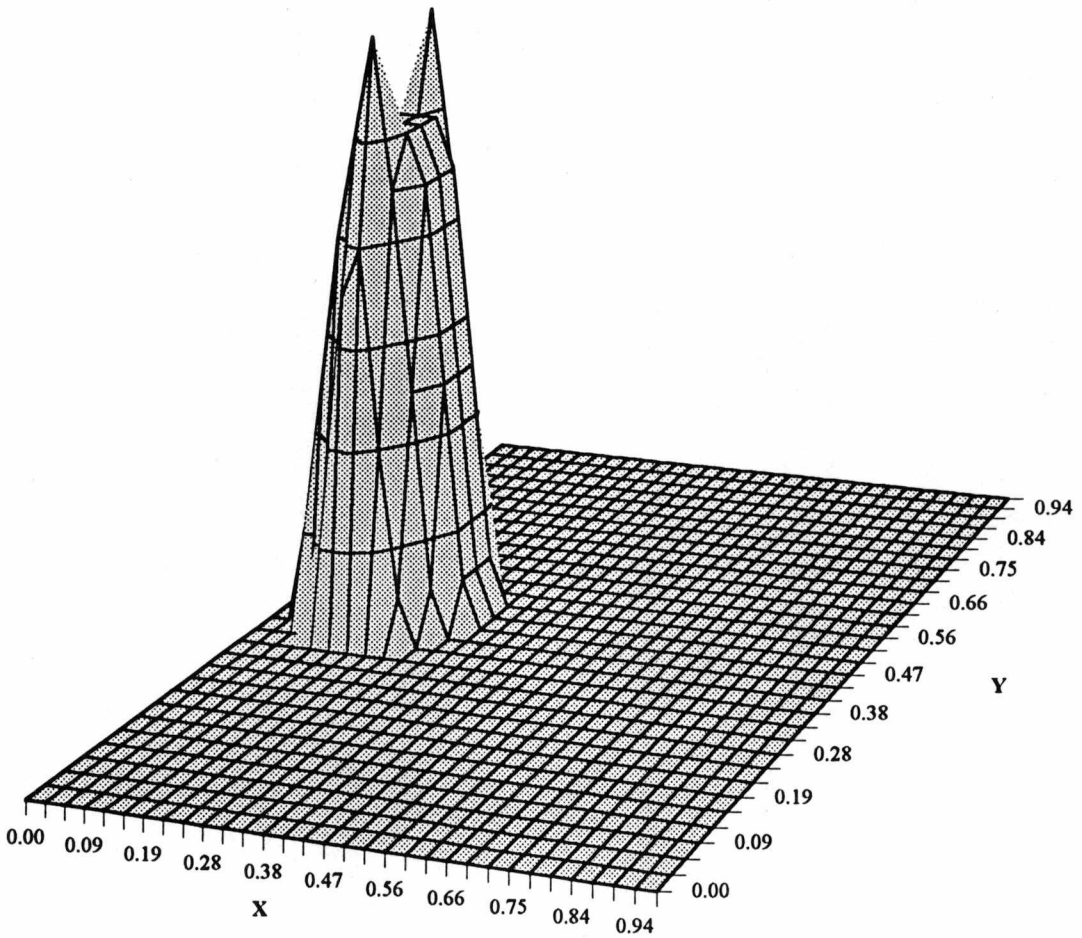

Figure 18: Advection with constant velocity.

using process migration in the three-time-level scheme. Times were recorded at each velocity and the speed-up factor, $\frac{T_{2\times2}}{T_{4\times4}}$, was computed. Ideally, the 4×4 configuration should be able to complete the task four times faster than the 2×2. However as the velocity increases, the speed-up factor is reduced (Figure 19). For small velocities (u = .00625), no trajectories go off processor and only communications related to necessary overlap an interpolatory overlap are required. The computational speed up however was not four, as communication was required to pass the necessary overlap of one and overlaps required in interpolation. One would expect that this degradation would diminish and finally cease as all estimates move off processor. This is interpreted in Figure 19 as the asymptotic behavior of the CPU time as the velocity increases. It is possible that different processor configurations could improve the loss in efficiency. In this example, since the wind is in the positive x direction only, one could use a 1 x 16 configuration that would require virtually no communication at all. Such configurations are problem dependent and require a consistent nature in the velocity field over time.

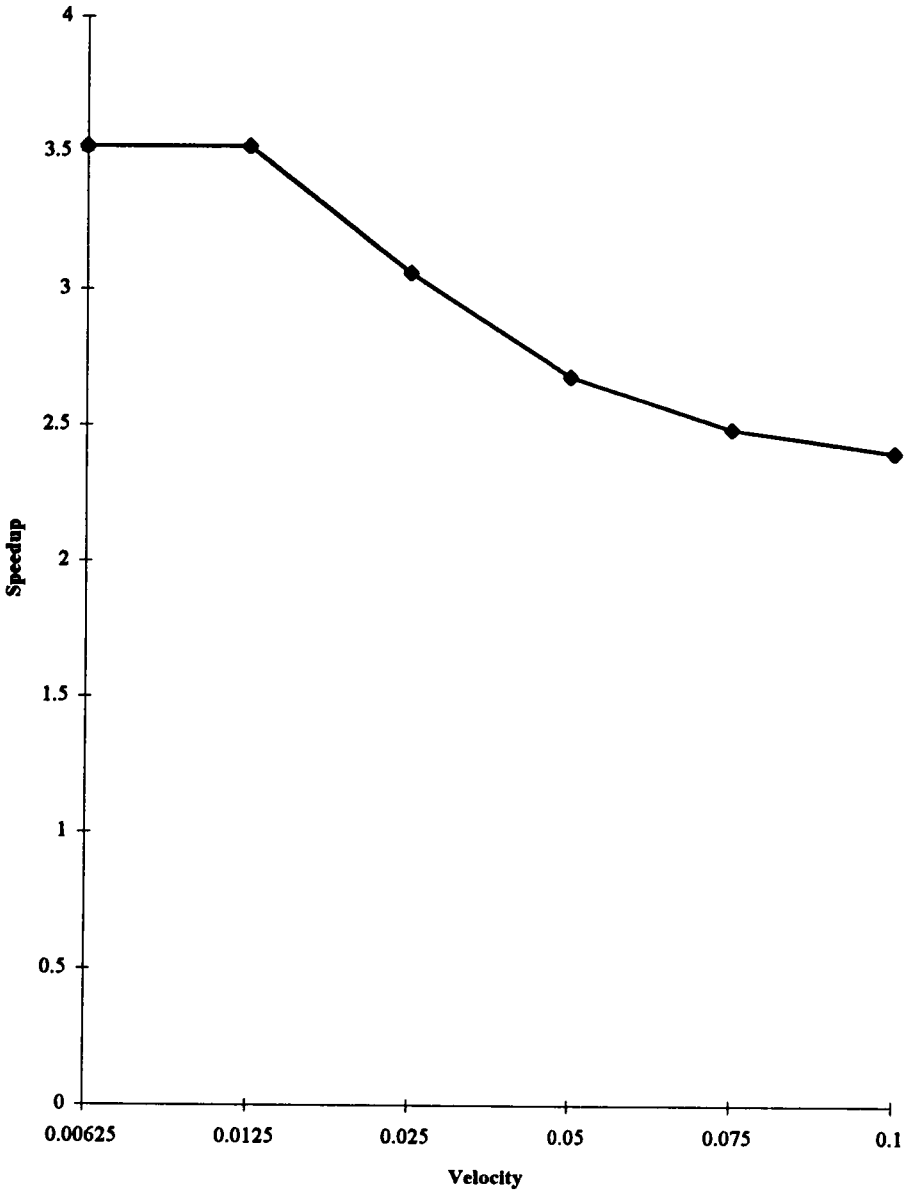

Figure 19: Decrease in parallel computational speed up as velocity field increases.

11.5 Comparison of Communication Algorithms

The results of the first exercise demonstrate the need for efficient communication algorithms. The next step was to evaluate the communication algorithms proposed: process, data, and hybrid migration. Moreover the computational performance as the number of overlaps increase from process migration to data migration and finally to complete overlapping was investigated. The three-time-level and the D_1 scheme were tested at a constant velocity of .075 in the positive x direction while increasing the number of overlaps. For this velocity, data migration requires 7 overlaps and process migration requires 3 when using cubic interpolation to estimate F.

Figure 20 shows the behavior as the overlaps increased. The more important result to notice is the presence of a optimal overlap. More importantly, this optimal overlap occurs with data migration. Process migration performed poorly in comparison, taking almost twice as long to complete the timestep. This suggests that the latency period is the dominant computational overhead. Even with 7 overlaps, with unnecessary overlapping above, below, and to the right of each processor (velocity is to the right only), the overall time was cut in half, since the number of messages was fewer. The

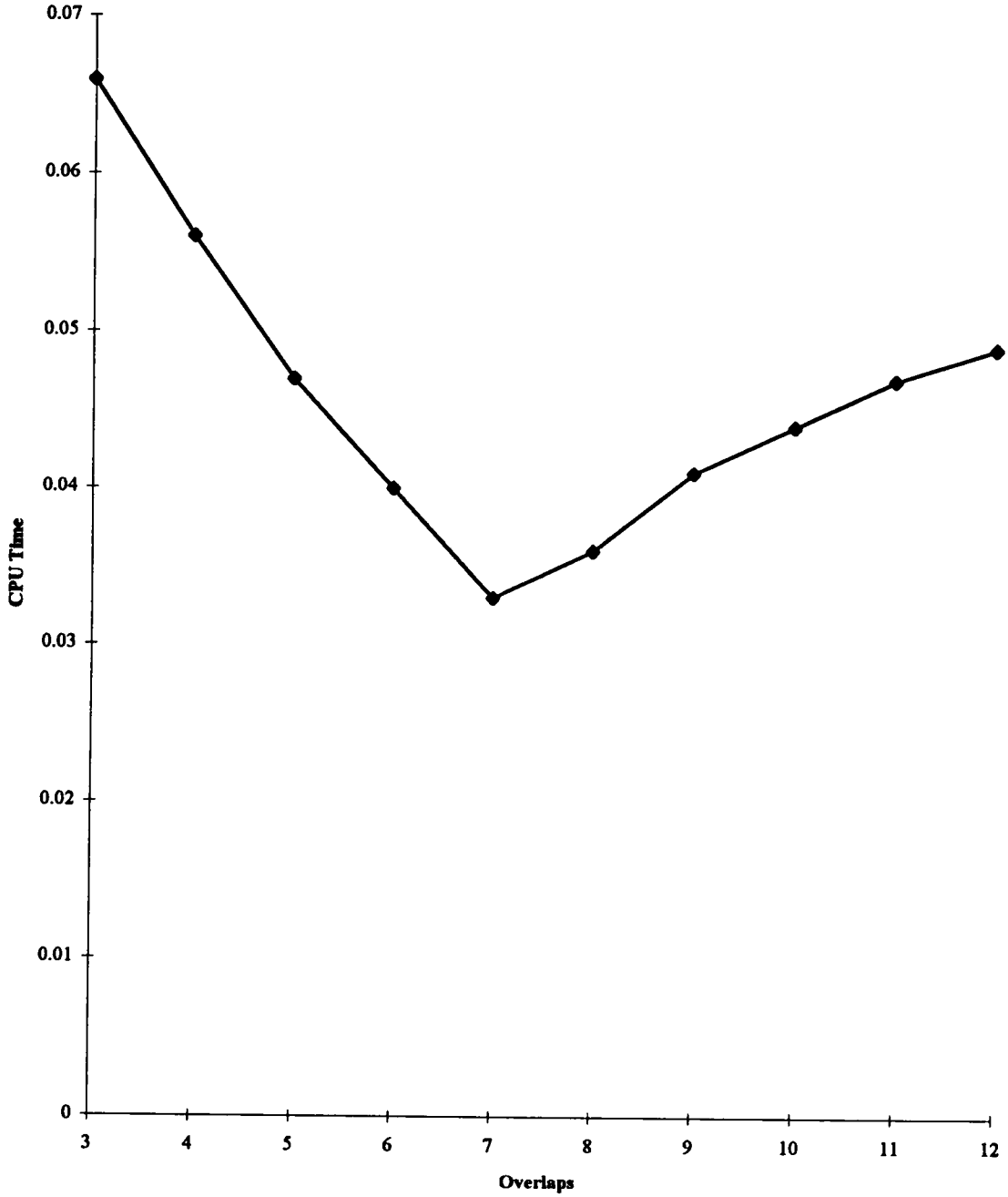

Figure 20: Computation times for three-time-level SLT.

same trend was seen in the D_1 scheme and was not included. While this is a positive result for data migration, it remains to be seen how this relationship evolves as velocity increases. As the velocity increases, data migration is forced to pass larger and larger blocks of information. One would expect that eventually, the information transfer time for larger blocks would supersede the latency period associated with numerous messages.

This relationship was the subject of the next test, which measured CPU times as the velocity increased for data and process migration as well as recording the optimal number of overlaps (Figures 21 and 22). For the three-time-level scheme, data migration and optimal overlapping were synonymous for speeds up to .25. Speeds above this were not tested (at .25, particles are moving half way across the global domain in a single time step). The D_1 scheme exhibited a different behavior with optimal timings switching from data to process migration between .2 and .225. This is the result one would expect eventually to observe. This occured earlier for D_1 than the three-time-level due to the absence of midpoint iterations and any associated processor communications. In the D_N scheme, for each departure point that goes off processor, only two message passes are required to determine F (sending the task and data recovery). In the three-time-level scheme, if only

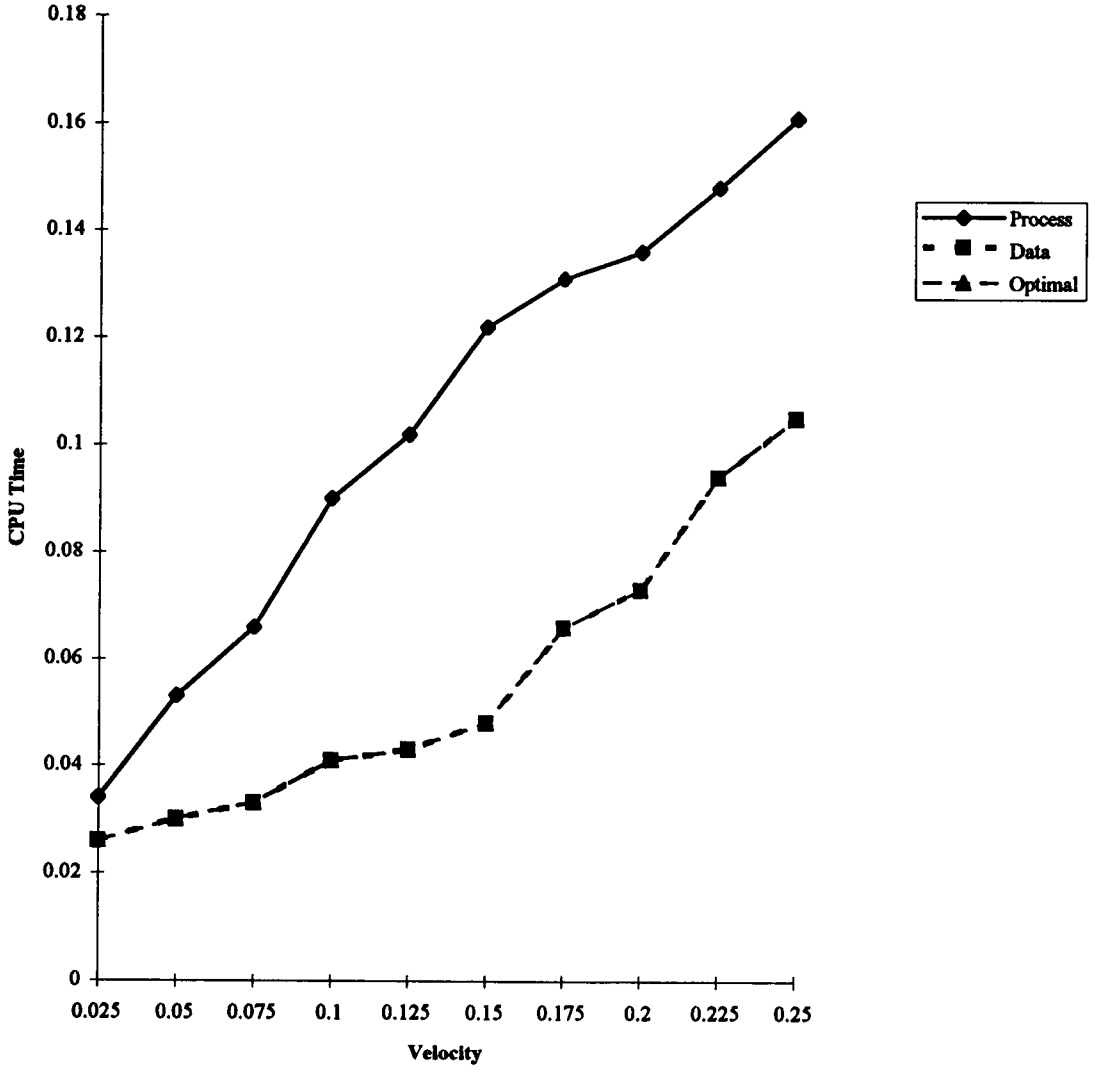

Figure 21: Process migration, data migration, and the measured optimals for three-time-level SLT.

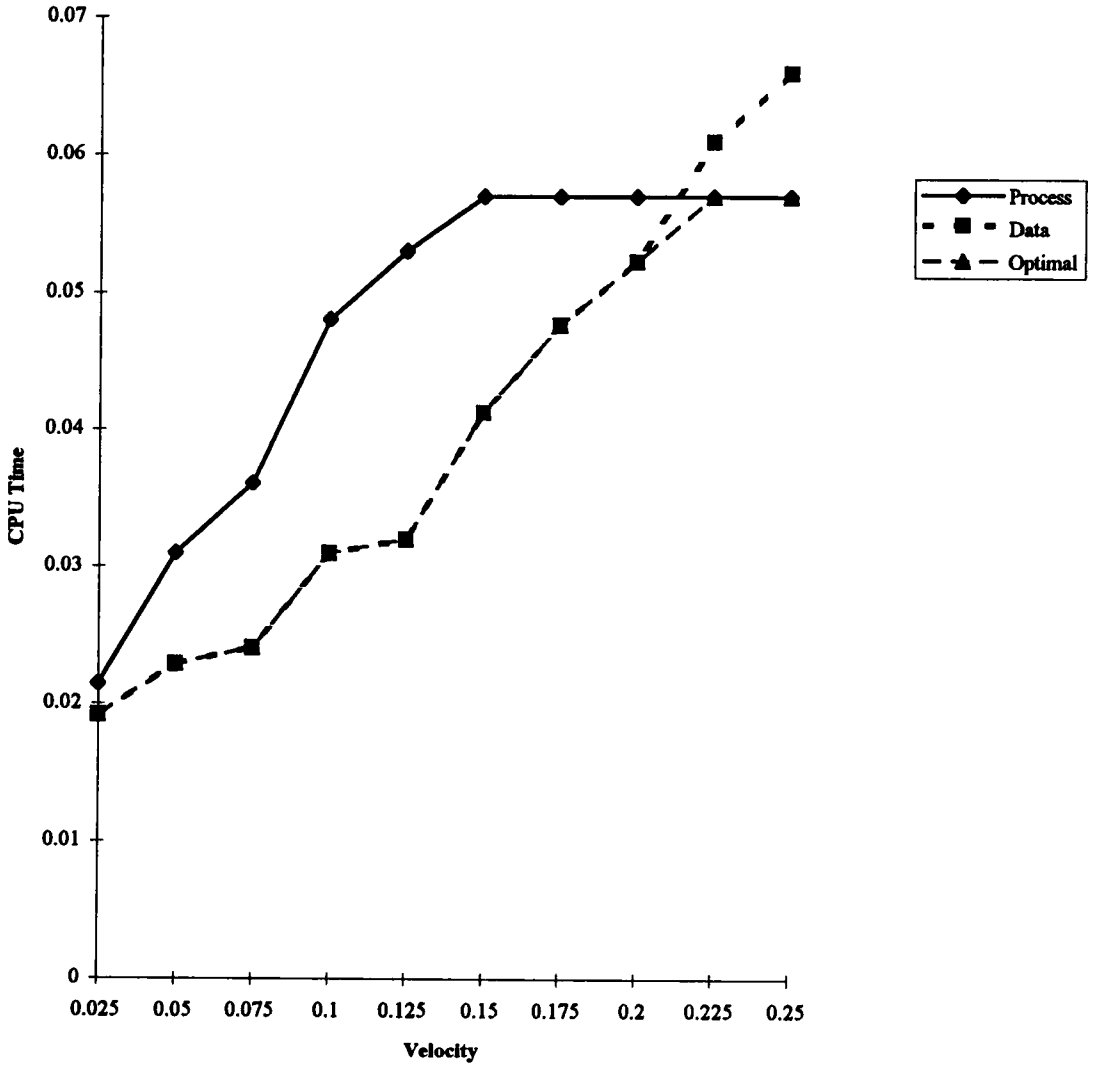

Figure 22: Process migration, data migration, and the measured optimals for D_1.

the departure point goes off processor, the same is true. However, for each midpoint iteration that goes off processor at least two more message passes are required. In short, when every trajectory moves off processor, the D_N scheme passes at most 2n messages per processor (n is the number of local grid points) independently of how far they move. The three-time-level scheme may have a significantly larger number of messages due to these midpoint iterations as several estimates in a single trajectory may fall into different domains. Hence, process migration continues to be less effective than D_N. One would expect that eventually as all midpoints move off processor, and further away, the role of optimality would switch as well in the three-time-level scheme. This event would require denser grids than were used in this study.

From observing these plots it might appear that hybrid migration performance always lies between process and data migration timings. This conclusion is wrong. Consider the case shown in Figure 23 for D_3. At a speed of .225 not only have all the departure points moved off processor, but the nearest departure point is a distance of more than 8 overlaps away on the right side. Hence hybrid migration was the least efficient when passing fewer than 9 overlaps. When passing fewer than 9, large blocks of information were

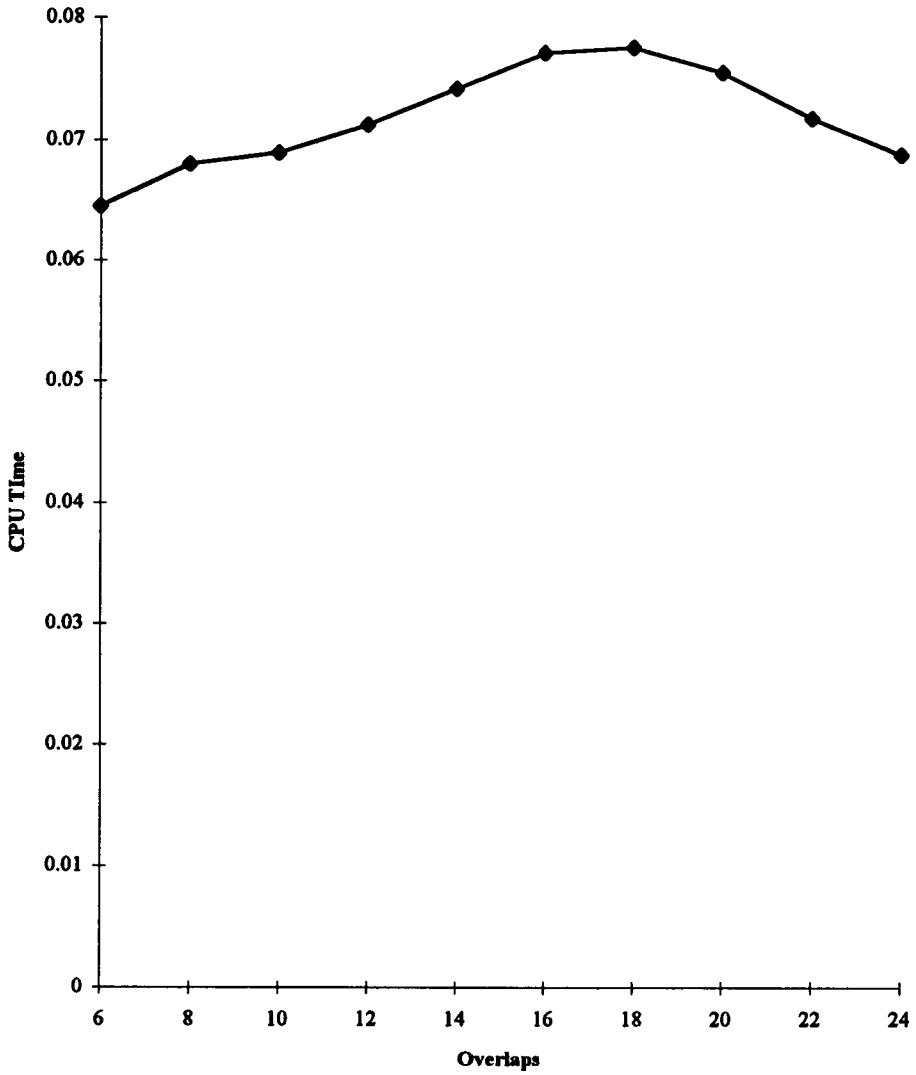

Figure 23: Computation times for D_3.

sent without capturing any of the departure points and thus required process migration to estimate every departure point.

11.6 Comparison of D_N and Three-Time-Level SLT

In order to directly compare the two advection schemes, optimal times for the three-time-level, D_1, and D_2 schemes are presented in Figure 24. For a constant velocity field, D_2 provides no benefit over D_1. It is included here to illustrate the differences in computational efficiency associated with the extra derivative estimations in D_2. While these results seem to be a negative for the three-time-level, it does not imply any global conclusions about its performance, particularly in light the non-constant velocity case (investigated later).

11.7 Computational Models

The preceding results suggest that latency plays a larger role than information transfer time in reducing parallel efficiency. In order to better quantify the relationship between the number of messages passed (NM), the number of bytes passed in those messages (NB), and the computation time (T), it is

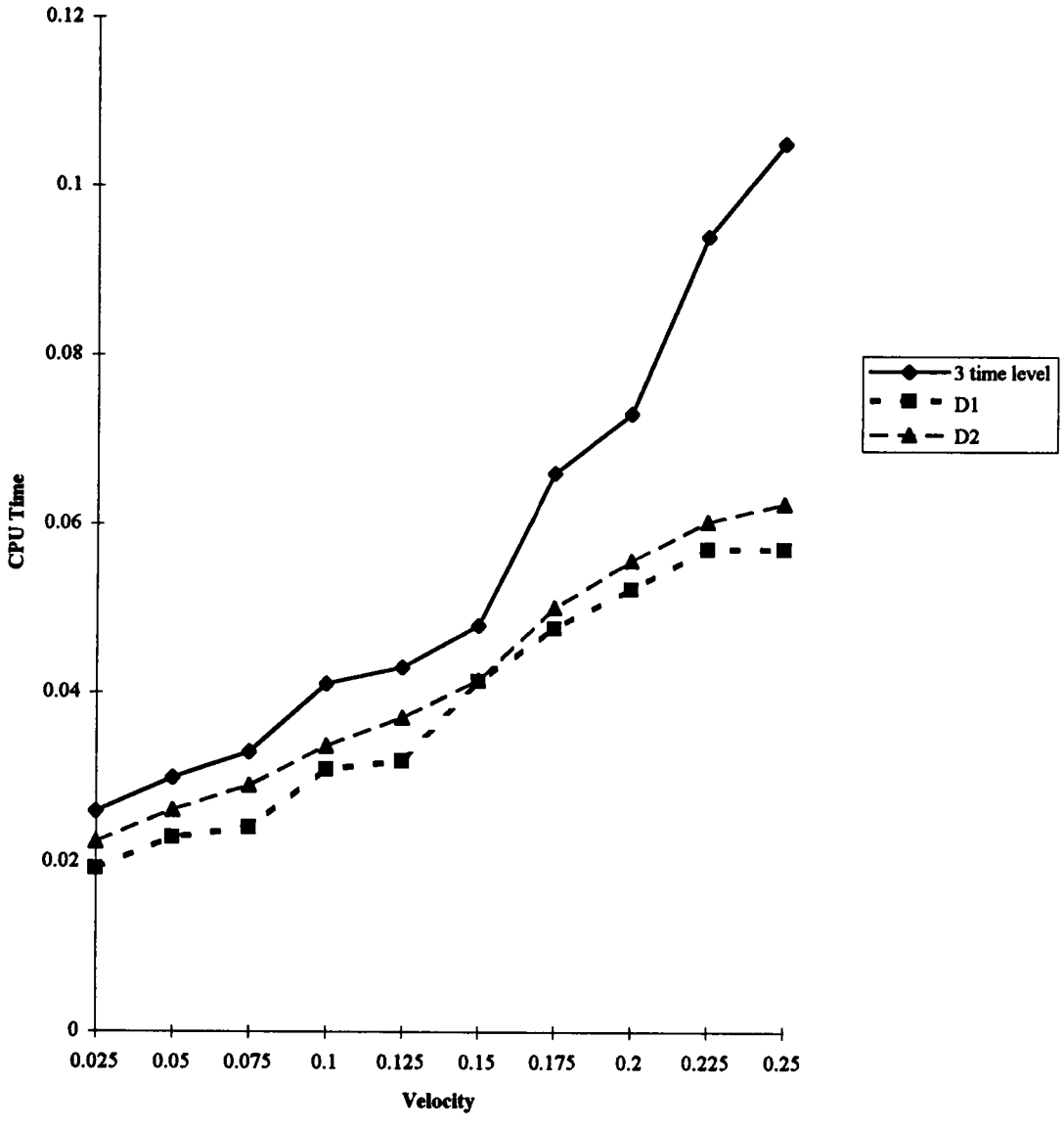

Figure 24: Optimal times for D_1, D_2, and three-time-level SLT.

beneficial to develop a performance model.

The performance model used here is

$$T = C + \gamma(NM) + \nu(NB) \tag{50}$$

where the constant C represents the time required for computation. In this model γ may be interpreted as a measure of the latency period and ν as a measure of the time required to pass a single byte. Estimates for these values exist for the iPSC/860 but were found to perform poorly in modeling the data. Therefore multiple linear regression was used to estimate the parameters from the data. This model was fit for both the three-time-level and D_N (N=3) schemes. Furthermore, a statistical analysis provides a means for estimating each variable's contribution to the CPU time.

Fitting the model for the three-time-level scheme produces

$$T = .000027(NM) + .0000008(NB) + .023 \tag{51}$$

To ascertain which variable contributes more, Type II sum of squares were determined [14]. This total was .0525 for NM and .0074 for NB. One may conclude that NM is contributing significantly more than NB to overall variability as was expected. Fitting the model for the D_3 scheme produced

similar results:

$$T = .000014(NM) + .003327(NB) + .020 \tag{52}$$

Type two sum of squares were .013 for NM and .003 for NB. Again NM is the dominant factor. By superimposing the constant C in the three-time-level performance model onto the graph of optimal times in Figure 25, one can observe the dominant role of communications.

11.8 Parallel Scalability

Scalability in parallelization refers to how effectively a parallel algorithm deals with problems on a larger scale. Two tests are typically conducted: (1) the number of grid points and processors increases uniformly and (2) the number of grid points remains constant as the number of processors increases. In the first test, the number of grid points per processor remains constant while the overall task increases. For a scalable algorithm, one should see a relatively constant computation time as the work load per processor is constant. The implication here is that given additional processors, larger problems may be completed within the same time as smaller scale problems. For the second test, a scalable algorithm would produce a significant decline in CPU times

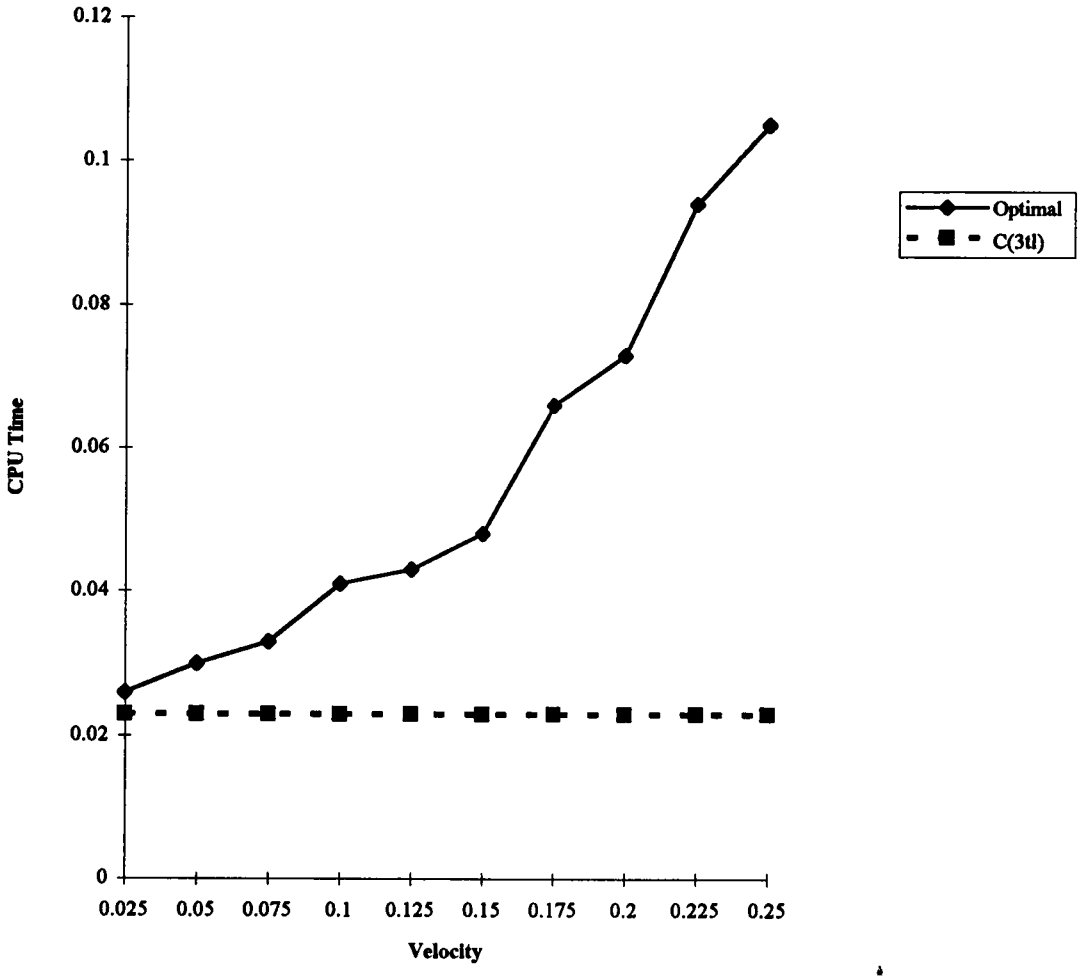

Figure 25: A comparison of the optimal timings and the estimated time for calculations only for the three-time-level scheme.

as the number of processors increased.

To address scalability, these tests were conducted for both the D_3 scheme and the three-time-level scheme at a constant velocity of .05 in the positive x direction. Comparisons were also made for data and process migration within each scheme. For the first test, the mesh began as a 16 x 16 solved by a 2 x 2 processor grid. This was eventually increased to a 64 x 32 mesh with a 8 x 4 processor grid. In the second test, the mesh was held constant at 64 x 64 as the number of processors increased.

The results for the three-time-level scheme are seen in Figures 26 and 27 and those for D_N are in Figures 28 and 29. For the first test it would appear that neither is very scalable. One would expect to see a constant CPU time of about .025 or .03 for the three-time-level scheme and similarly for D_3. Within each scheme, data migration seemed to scale better as it increased more slowly. The second test produced better results. For the three-time-level scheme using process migration, computation times were decreased by a factor of only 3 in moving from 4 to 32 processors. Data migration did better with a speed factor of over 5. In the D_3 test, data and process migration had speed-up factors of about 4 and 3 respectively. Speed up factors of 8 were not experienced due to communication times (see section 11.3).

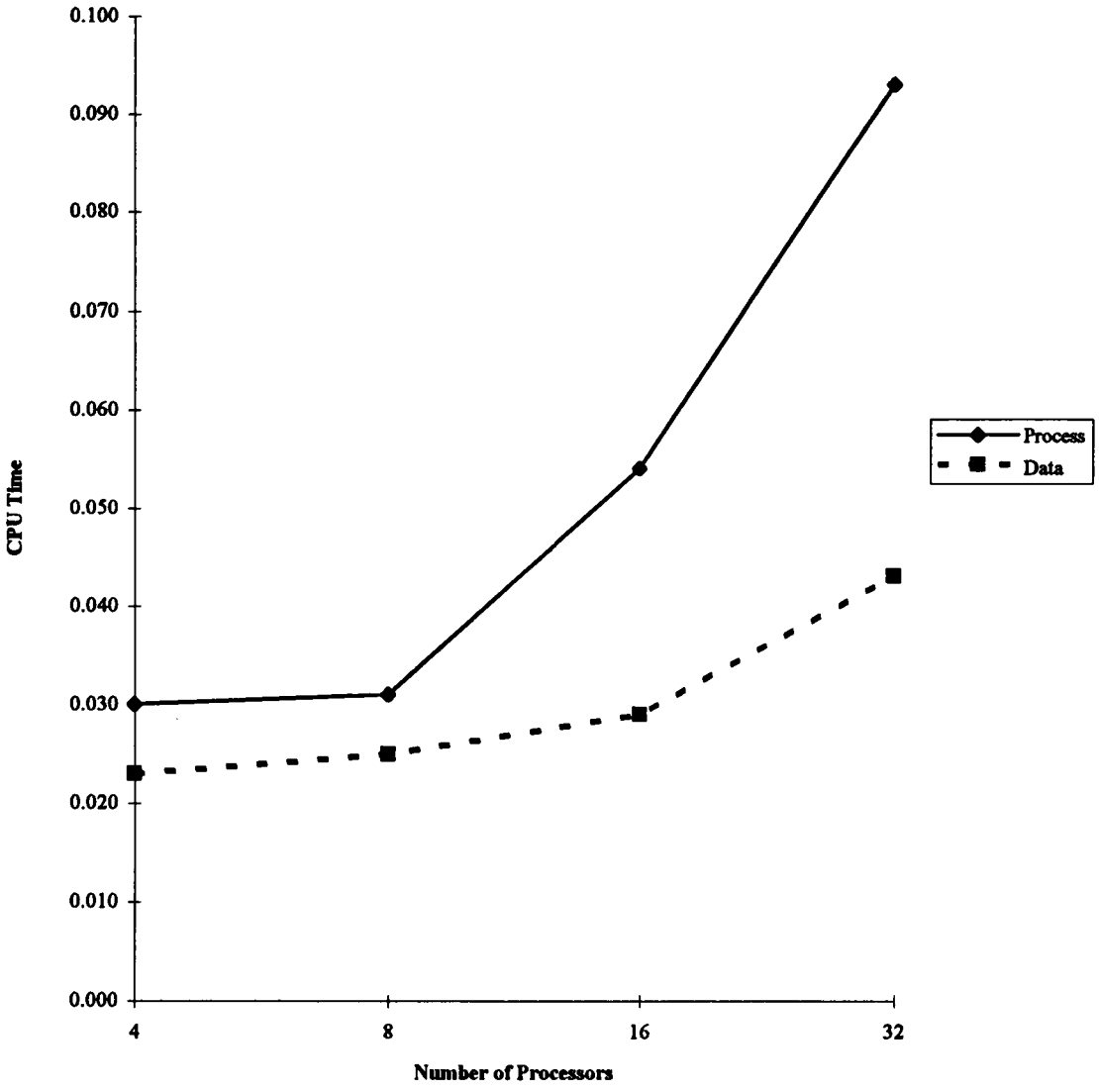

Figure 26: Number of points per processor held constant in the three-time-level scheme.

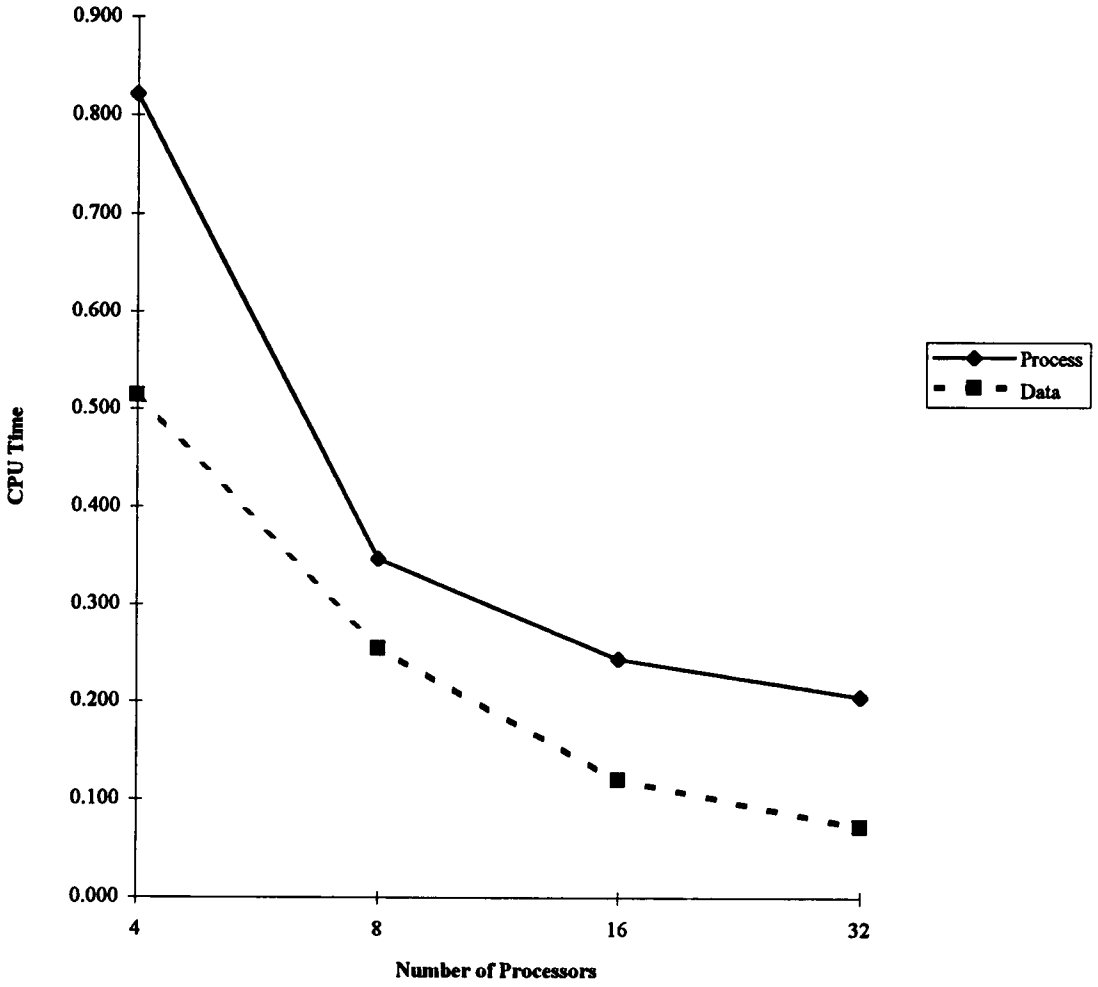

Figure 27: Number of grid points held constant in the three-time-level scheme.

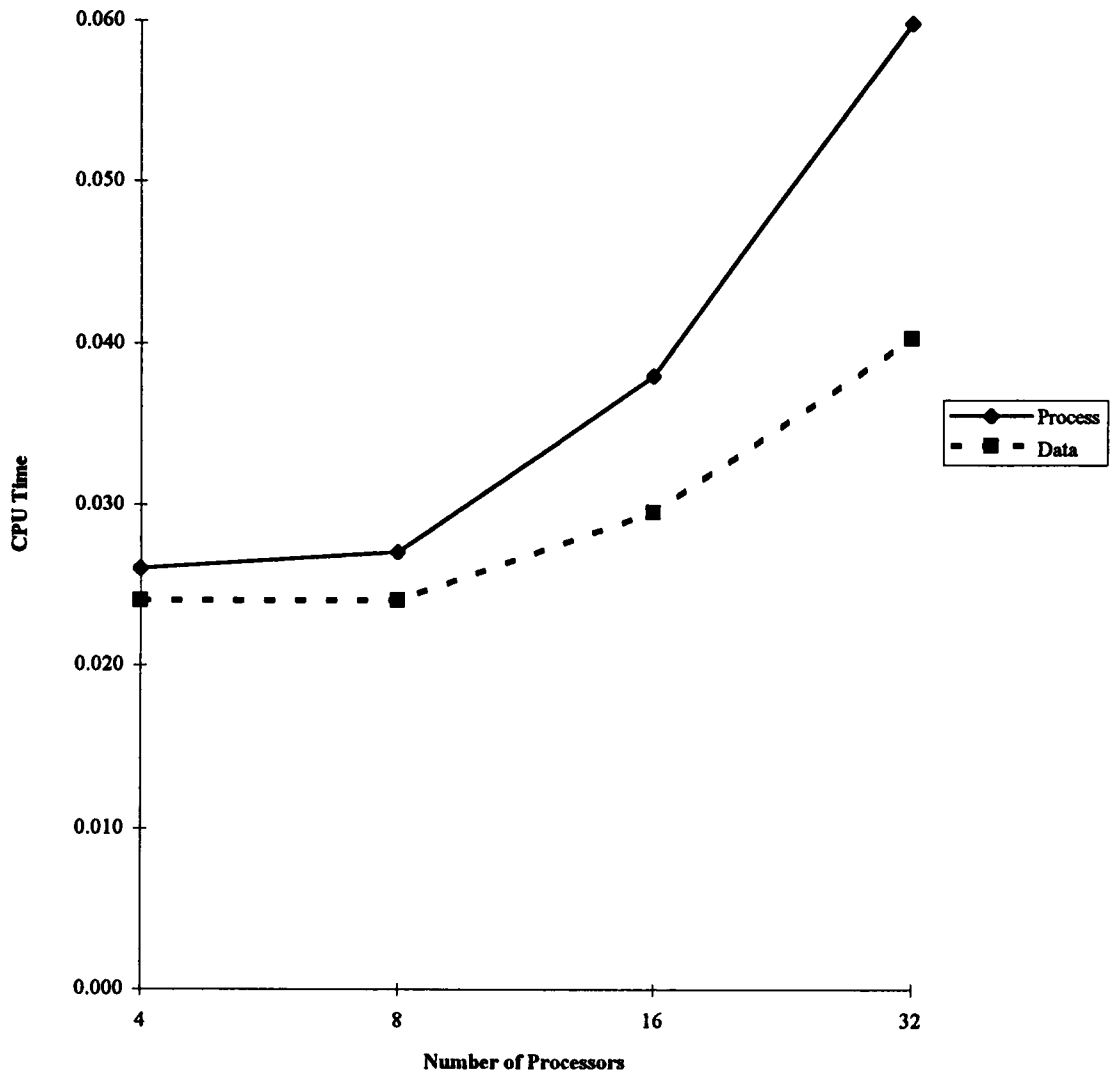

Figure 28: Number of points per processor held constant in the D_3 scheme.

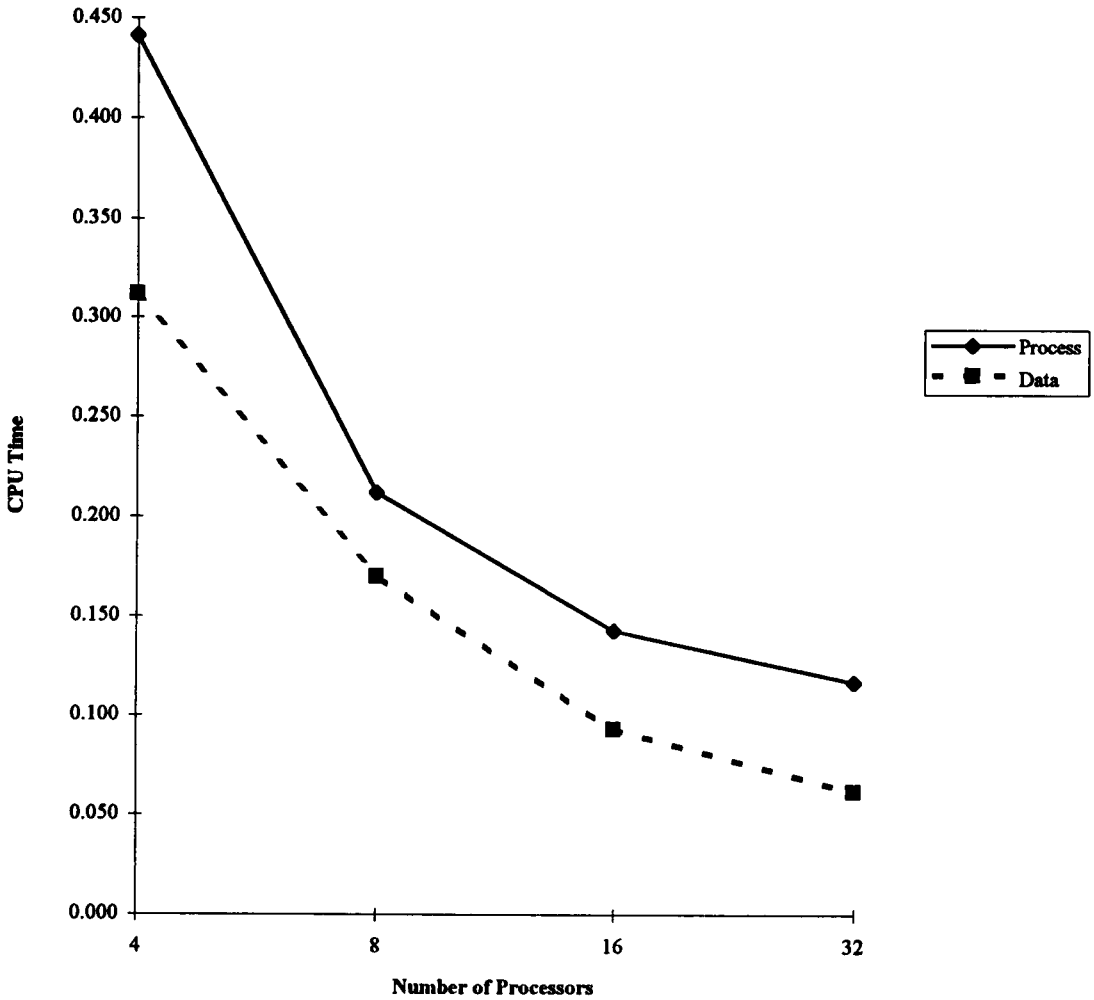

Figure 29: Number of grid points held constant in the D_3 scheme.

11.9 Non-Constant Velocity

This topic is briefly discussed to illustrate how conclusions drawn from the constant velocity case cannot be extended to non-constant situations. To examine these differences, D_1 was used to advect the cylinder in a velocity field defined by

$$u(x) = .1 \sin(\pi(i\Delta y)), \tag{53}$$

$$v(y) = 0 \tag{54}$$

where i is the grid row index and Δy is the vertical distance between grid points. Hence on the interval from [0,1), u(x) is the positive half of the sine wave with a maximum of .1 occuring at $i\Delta y = .5$ In this situation, particles near the vertical center of the domain move faster. From a semi-Lagrangian viewpoint, this corresponds to more trajectories going off processor in the center than at the edges. Since the maximum velocity is .1, then data migration would require an overlap of 9 and for a constant velocity field this is also the optimal overlap. Here an overlap of 8 (hybrid migration) was measured as optimal. In Figure 30, the solid line corresponds to measured CPU times at each overlap. The superimposed dashed line corresponds to the data migration time at 9 overlaps. Figure 31 shows the effect on the cylinder (seen

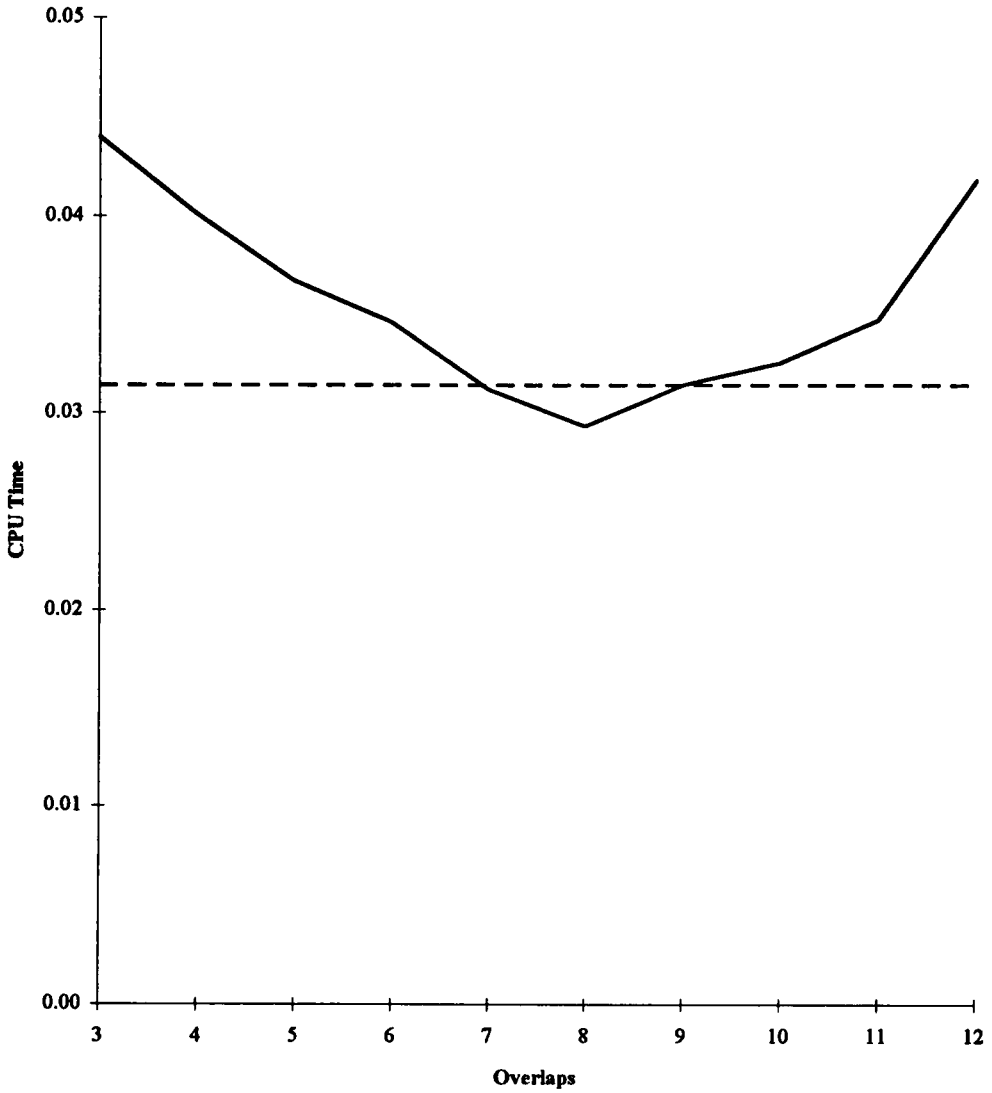

Figure 30: Non-constant velocity times are represented by the solid line. The superimposed dashed line is the data migration time.

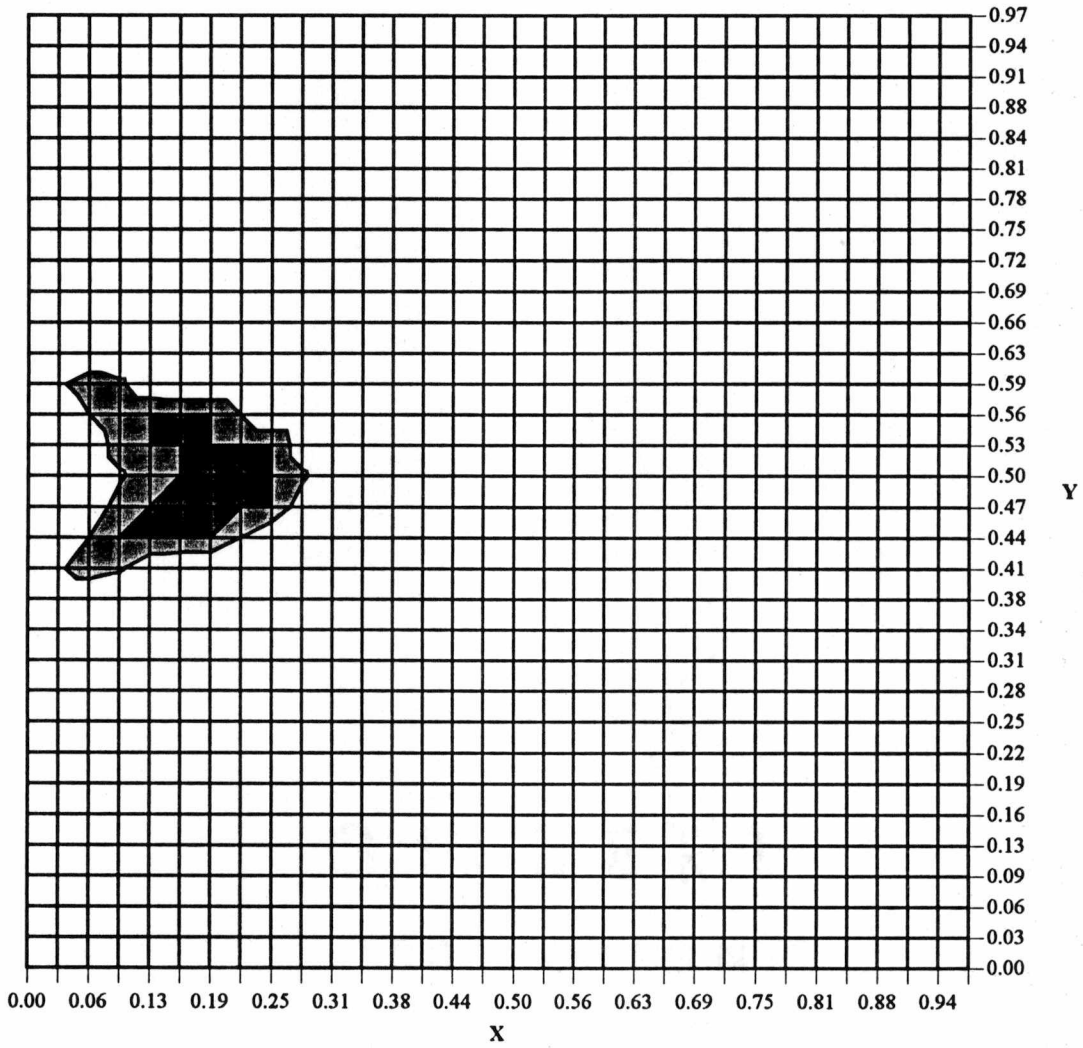

Figure 31: Advection with non-constant velocity.

from above) after only 5 time steps. As expected the center of the cylinder is moving faster than the high and low edges.

12 Summary

A parallel algorithm was developed for the three-time-level semi-Lagrangian advection scheme. Depending on the problem, interprocessor communications may be significant in this method. Therefore two primary communication algorithms within this framework were proposed (data and process migration) and used in understanding how communication requirements influence computational efficiency. Furthermore, this parallelization method was modified to produce a parallelization of the alternative semi-Lagrangian scheme D_N. The three-time-level SLT was then compared with this alternative in terms of parallel efficiency and scalability for constant velocity fields.

There was strong indication that the dominant factor in reducing parallel efficiency is the latency time associated with message passing, as process migration was consistenly less efficient than data migration. Furthermore data migration was found to produce minimum computation times. However in the D_N scheme, process migration became more beneficial as the velocity increased. This difference can be attributed to the absence of midpoint iterations in D_N. The multiple linear regression model (50) was fit to the data and a statistical analysis confirmed the larger role of latency for both

schemes. These conclusions cannot be recklessly extended to non-constant velocity, as illustrated in section 11.9. While the findings for constant velocity suggest that process migration performs poorly for constant velocity using this discretization, it may perform better for finer discretizations (data migration must send even larger blocks of information) or for non-constant velocity fields.

Scalability was investigated for each scheme. Two tests were conducted to investigate the performances as the size of the problem increased. The first test showed that given a larger problem, a larger set of processors does not maintain the computation times seen in smaller problems operated on by fewer processors. The second test showed better results. There the size of the problem was held constant and the number of processor increased. The effect was a less than linear speed-up in computation time. However the reduced CPU time could be considered an improvement. Furthermore, scalability was found to be dependent on the velocity expected. The increase in efficiency of a 4 x 4 over a 2 x 2 processor grid was found to decrease as the velocity field, and subsequently the communication times increased.

Certain measures in optimization could improve the parallel efficiency of the algorithm. Some of these are briefly mentioned but not addressed. The

emphasis here is to develop the parallel method and gain an understanding of the impact of message passing on computational efficiency. These results may vary from machine to machine as architecture and system parameters such as latency time may vary.

BIBLIOGRAPHY

References

[1] Andrews, L. (1986). Elementary Partial Differential Equations. Academic Press, Inc.

[2] Bates, J. and McDonald, A. (1982). Multiply-Upstream, Semi-Lagrangian Advective Schemes: Analysis and Application to a Multi-Level Primitive Equation Model. *Monthly Weather Review*, **110**, 1831 - 1842.

[3] Bear, J. (1972). Dynamics of Fluids in Porous Media, Dover Publications, Inc., NY.

[4] Hughes, T. and Marsden, J. (1976) A Short Course in Fluid Mechanics, Publish or Perish, Inc.

[5] Kincaid, D. and Cheney, W. (1991) Numerical Analysis, Brooks Cole Publishing Company.

[6] LeVeque, R. (1990). Numerical Methods for Conservation Laws, Birkhauser Verlag.

[7] McDonald, A. (1984). Accuracy of Multiply-Upstream, Semi-Lagrangian Advective Schemes. *Monthly Weather Review*, **112**, 1267 - 1275.

[8] McGregor, John. (1993). Economical Determination of Departure Points for Semi-Lagrangian Models. *Monthly Weather Review*, **121(1)**, 221 - 228.

[9] Mitchell, A. and Griffiths, D. (1980). The Finite Difference Method in Partial Differential Equations, John Wiley & Sons.

[10] Ritchie, Harold. Application of a Semi-Lagrangain Integration Scheme to the Moisture Equation in a Regional Forecast Model, *Monthly Weather Review*. **113**, 424 - 435.

[11] Ritchie, Harold. (1986). Eliminating the Interpolation Associated with the Semi-Lagrangian Scheme. *Monthly Weather Review*, **114**, 135 - 146.

[12] Robert, Yee, Ritchie. (1985). A Semi-Lagrangian and Semi-Implicit Numerical Integration Scheme for Multilevel Atmospheric Models. *Monthly Weather Review*, **113**, 388 - 394.

[13] Saniforth and Cote. (1991). Semi-Lagrangian Integration Schemes for Atmospheric Models – A Review. *Monthly Weather Review*, **119**, 2206 - 2223

[14] Sas Institute Inc. (1989). SAS/STAT User's Guide Version 6, Fourth Edition, **2**.

[15] Swokowski, Earl. (1984). Calculus with Analytic Geometry. Prindel, Weber, and Schmidt.

[16] Williamson, D. and Rasch, P. (1990). On Shape-Preserving Interpolation and Semi-Lagrangian Transport. *Siam J. Sci. Stat Comput.*, **11(4)**, 656-687.

[17] Williamson, D. and Rasch, P. (1989).Two-Dimensional Semi-Lagrangian Transport with Shape Preserving Interpolation. *Monthly Weather Review*, **117**, 102 - 219.

VITA

Robert Stewart was born in Oak Ridge, TN in 1970. He has lived since in Wartburg, TN where he attended both elementary school and high school. In August of 1992, he graduated from the University of Tennessee with a Bachelor of Science degree, double majoring in Mathematics and Statistics. In that same month, he entered graduate school at the University of Tennessee working on a Master of Science degree in Mathematics with a concentration in numerical mathematics. He graduated with a MS degree in May of 1995. Currently, he is employed by the University of Tennessee, working as a Senior Research Assistant at Oak Ridge National Laboratories.