DEVELOPMENT OF IMPROVED
FINITE ELEMENT FORMULATION FOR
SHALLOW WATER EQUATIONS

by

Edward T. Woodward

September 1981

Thesis Advisors:
R.T. Williams
U.R. Kodres

Approved for public release; distribution unlimited.

Copy available to DTIC does not permit fully legible reproduction
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
The basic principles of the Galerkin finite element method are discussed and applied to two different formulations: one using different basis functions and the other using the vorticity-divergence form of the shallow water equations. Each formulation is compared to the primitive form of the equations developed by Kelley (1976). The testing involves a comparison of three finite element prediction models using variable size elements. Equilateral elements significantly improve the solution and are used in most of the comparisons.
The formulation using different basis functions produces poorer results than the primitive formulation. The vorticity-divergence formulation produces superior results while executing faster than the primitive model. However, it does require more storage and the relaxation parameters are sensitive to the domain geometry. The computer implementation for the vorticity-divergence model is discussed and the source listing is included.
DEVELOPMENT OF IMPROVED
FINITE ELEMENT FORMULATION FOR
SHALLOW-WATER EQUATIONS

by

Edward T. Woodward
Captain, United States Air Force
B.A., University of Maine, 1970

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN COMPUTER SCIENCE
and
MASTER OF SCIENCE IN METEOROLOGY

from the

NAVAL POSTGRADUATE SCHOOL
September 1981

Author: Edward T. Woodward

Approved by: Roger T. Williams
Thesis Advisor

Eino G. Evens
Thesis Co-Advisor

R.L. Case
Chairman, Department of Computer Science

R.L. Epping (Acting)
Chairman, Department of Meteorology

H. W. Wood
Dean of Information and Policy Sciences

William M. Feller
Dean of Science and Engineering
ABSTRACT

The basic principles of the Galerkin finite element method are discussed and applied to two different formulations; one using different basis functions and the other using the vorticity—divergence form of the shallow water equations. Each formulation is compared to the primitive form of the equations developed by Kelley (1976). The testing involves a comparison of three finite element prediction models using variable size elements. Equilateral elements significantly improve the solution and are used in most of the comparisons. The formulation using different basis functions produces poorer results than the primitive formulation. The vorticity—divergence formulation produces superior results while executing faster than the primitive model. However, it does require more storage and the relaxation parameters are sensitive to the domain geometry. The computer implementation for the vorticity—divergence model is discussed and the source listing is included.
# Table of Contents

I. Introduction ..................................................... 11
   A. Background .................................................. 12
   B. Objectives .................................................. 15
   C. Thesis Structure ............................................. 17

II. Finite Elements .................................................... 19
   A. Finite Element Concept .................................... 20
   B. Galerkin Application ....................................... 25
   C. Area Coordinates ........................................... 31

III. Shallow-Water Model .............................................. 35
   A. Governing Equations ....................................... 37
   B. Equation Formulation ...................................... 39
   C. Time Discretization ....................................... 41
   D. Computational Technique ................................... 43
   E. Grid Geometry .............................................. 45
   F. Initial Conditions ......................................... 47
   G. Boundary Conditions ....................................... 49

IV. Computer Implementation ......................................... 52
   A. Program Architecture ...................................... 53
      1. Main Program ............................................ 55
      2. Initialization Phase ................................... 56
      3. Forecast Phase .......................................... 64
      3. Utility Modules ......................................... 65
V. PRIMITIVE MODEL EXPERIMENT.............................................68
  A. MODEL DESCRIPTION.................................................68
  B. RESULTS..............................................................73

VI. LINEARIZED MODEL EXPERIMENT........................................76
  A. EQUATION REFORMULATION.........................................78
  B. RESULTS..............................................................81

VII. VORTICITY-DIVERGENCE MODEL EXPERIMENTS..............................84
  A. TEST DOMAINS AND INITIAL CONDITIONS..........................84
  B. TEST CASE COMPARISONS...........................................87
    1. Regular Case.....................................................87
    2. Smooth Case....................................................91
    3. Abrupt Case....................................................102
  C. COMPUTATIONAL SENSITIVITY......................................105

VIII. CONCLUSIONS..................................................................109

APPENDIX A (Source List)......................................................111
  A. DEFINITIONS............................................................111
  B. MAIN PROGRAM.........................................................114
  C. INITIALIZATION PHASE...............................................115
  D. FORECAST PHASE.......................................................141
  E. UTILITY PROGRAMS....................................................151

LIST OF REFERENCES..........................................................167

INITIAL DISTRIBUTION LIST..................................................169
LIST OF FIGURES

Figure 1. Piecewise linear basis function.................23
Figure 2. Basis and test function interaction during the piecewise integration process.........................27
Figure 3. Basis function for node 28........................30
Figure 4. Cartesian versus area coordinates..................32
Figure 5. Transformation to area coordinates................32
Figure 6. Domain divided into equilateral triangles..............46
Figure 7. 3 dimensional view of the initial fields...........................48
Figure 8. Assemble and store the coefficient matrix for element 3..................................62
Figure 9. Domain divided into right triangles....................71
Figure 10. Initial fields for the primitive model..................72
Figure 11. 48 hour forecast comparison between equilateral and right triangular elements.........................74
Figure 12. Differently shaped basis functions.................77
Figure 13. Staggering the geopotential about the velocity.....................................79
Figure 14. Initial fields for the linearized experiment..................82
Figure 15. 48 hour forecast comparing the primitive vs linearized model..........................83
Figure 16. Test domain geometries..................................85
Figure 17. Initial field for the Regular Case.....................88
Figure 18. 48 hour forecast comparison for the Regular Case.................................89
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.</td>
<td>Comparison of computational times for the 48 hour forecast, Regular Case</td>
<td>52</td>
</tr>
<tr>
<td>Table 2.</td>
<td>Comparison of computational times for the 48 hour forecast, Smooth Case</td>
<td>51</td>
</tr>
<tr>
<td>Table 3.</td>
<td>Comparison of computational times for the 48 hour forecast, Smooth Case</td>
<td>23</td>
</tr>
<tr>
<td>Table 4.</td>
<td>Comparison of computational times for the 48 hour forecast, Abrupt Case</td>
<td>105</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

The author believes that most of the credit should go to Professor R.T. Williams for the education that he gave me through his patient teaching and guidance during the entire research and writing of this paper. Acknowledgement is also due Professor U.R. Kodres for the many useful suggestions, encouragement and reading the manuscript.

Special thanks to each department chairman, Professor G.R. Bradley, Computer Science, Professor R.J. Renard, Meteorology, and Professor J.J. Saltiner, former Meteorology chairman, for allowing me to pursue the two degrees and accepting this thesis as partial fulfillment for the requirements of both degrees.

Thanks are extended to Lt. R.G. Kelley whose research was the forerunner to this investigation and Capt. M. Older for the many useful comments he offered during the implementation. Particular thanks to Dr. M.J.P. Cullen whose personal suggestions provided the direction for this work.

I give my love to my wife, Joy, for her infinite patience, for being the first guinea pig in reading and preparing this manuscript. Finally, to my family, Mr. Mark and Joy thanks for enduring and encouraging me throughout my graduate studies at the Naval Postgraduate School.
I. INTRODUCTION

Shuman (1978) claims that progress in numerical modeling of the general circulation has been to some degree dictated in the past by the rate of development in the field of computer technology. However, the limited ability to parameterize the effects of small-scale processes in terms of large scale motions has been an equally important limiting factor. Essentially, the major problem of numerical modeling of the general circulation is simply that of producing a very long range numerical weather forecast.

Certainly the equations used in the models must be more sophisticated to include those physical processes which are unimportant for a short range forecast, but may become crucial as the length of the forecast is extended. Another area where concentrated efforts have improved the forecast involves the computational techniques employed to approximate and solve the governing equations of the models.

The motivation behind this thesis is to investigate the application of a relatively new computational technique to the field of numerical weather prediction. The finite element method, long established in engineering, has been seriously considered only during the past decade in meteorology. This method has great potential for application in atmospheric prediction models.
A. BACKGROUND

The most common numerical integration procedure for weather prediction has been the finite difference method in which the derivatives in the differential equations of motion are replaced by finite difference approximations at a discrete set of points in space and time. The resulting set of equations, with appropriate restrictions, can then be solved by algebraic methods. Until recently, the finite difference method has been the workhorse in atmospheric prediction models, from their first computer implementation to the present.

With the introduction of each new generation of computers, the gap between numerical forecasts and atmospheric observations has decreased. The rate at which this gap decreased has slowed down and appears to be leveling off. This would indicate that computer technology may not be the primary obstruction to better numerical forecasts. In fact, bigger and faster computers alone have demonstrated their inability to significantly improve the numerical forecast.

For example, a major limiting factor of finite difference approximations is the truncation error. The National Weather Service 7 Layer Primitive Equation Model (7LPE Model), operational from 1966 to 1980, had truncation errors which increased at a rate proportional to the square of the grid spacing. That is, the smaller the grid interval,
the smaller the truncation error. To increase its accuracy would require increasing the grid matrix density. This would require increased computer storage and computational time. State of the art computers are capable of providing these additional resources.

The problem now goes beyond numerical techniques and computer technology. Operationally, the National Weather Service is not capable (due to monetary restrictions) of providing a denser concentration of atmospheric observations. Therefore, with the present density of initial data (observations) and objective analysis techniques (getting the data for grid points by interpolating from observed data sources), reducing the grid spacing further on the 7LPE Model does not significantly increase the accuracy of the solution.

This additional computer capability cannot be utilized using finite difference methods. Therefore, new numerical integration techniques must be investigated, such that given the same density of observed data, superior solutions are produced.

Two alternative techniques, the spectral method and the finite element method, have started to gain attention. Both the spectral and finite element methods require more computational time per forecast time step than does the finite difference method. For example, the finite element method requires an equation solver to invert a larger matrix.
at each time step for each variable. In this sense, these methods were held back by computer technology, but recent advances in computer technology (i.e. larger and faster storage devices, multi-processors, etc) have made these alternative numerical techniques competitive.

For long range weather predictions, the spectral method applied over the globe or hemisphere is a natural method, due to the existence of efficient transforms for the nonlinear terms on spherical geometry. It also eliminates the truncation error for the horizontal space derivatives and the nonlinear instability (aliasing). For these reasons, global spectral models have been developed and implemented on an operational level, replacing the global finite difference models.

However, because the spectral harmonics are globally rather than locally defined, it is thought that for problems of more detailed limited area forecasting, the finite element method is more suitable. Pioneering work to adapt finite element methods to meteorological applications has been done by Cullen (1973, 1974 and 1979), Staniforth and Mitchell (1977), Hinsman (1975) and Kelley (1976). The most recent finite element meteorological model at the Naval Postgraduate School was written by Kelley (1976) with the collaboration of Dr. R.T. Williams. It is this study that will serve as a basis for this thesis. The model written by Kelley will be altered and used for comparative testing with
Improved finite element forms implemented by this author. Some of the techniques and codes developed by Kelley are also employed in this thesis. Older (1981) developed a technique to smoothly vary the grid geometry in the domain. This technique is also implemented both on Kelley's model and with the new formulation to give greater versatility when testing the model performance.

3. OBJECTIVES

The objectives for this thesis can be divided into two categories: 1) meteorology, 2) computer science. First, the meteorological objectives of developing improved finite element forms for shallow water equations are as follows:

1) - Older (1981) after collaboration with Dr. M.J.P. Cullen, showed how equilaterally shaped elements produced significantly better results than did other triangular elements. Kelley (1976) used right triangular elements in the implementation of a two-dimensional finite element model using the primitive form of the shallow water equations. A considerable amount of small-scale noise was observed in the solution. Hereafter, this model, which was developed by Kelley (1976), will be referred to as the primitive model. This first objective involves re-implementing the primitive model using equilaterally shaped elements and comparing the results to those in Kelley's thesis.
2) - Williams and Zienkiewicz (1981) presented new finite element techniques for formulations for the shallow water equations, which use differently shaped functions to approximate the different dependent variables, which in effect stagger the variables. Schoenstadt (1980) demonstrated the advantage of spatial staggering of dependent variables in finite difference models. The application of this technique to finite element models is a natural extension, and excellent results were obtained by Williams and Zienkiewicz (1981) from application of these new formulations on linearized one dimensional cases. The objective here is to implement the new forms on the primitive model and again do quantitative comparisons of the results.

3) - The major emphasis in this study deals with the implementation and comparison of the vorticity divergence form of the shallow water equations, which is described in detail in Chapter III. This formulation has the following advantages. First, the geostrophic adjustment process is treated better than in the primitive form of the equations. Secondly, the velocity and height fields are evaluated at the same grid point, where the best primitive form requires staggering these dependent variables. And thirdly, a larger time step is allowed due to the semi implicit form of the equation. Again comparisons between the results from the vorticity divergence and primitive model are presented.
The computer science aspect of this thesis was primarily devoted to the implementation of the different models and the style and architecture of the program. Finite element methods require more computational time than do finite difference methods, not only in the solution of the equations, but also in the amount of computation required to evaluate each term in the equations.

The implementations of these two dimensional models, although complex when viewed from the surface, have a lot of generality and redundancy in the operations required. Versatile modules can be written to ease the implementation and facilitate changes. The objective here is to efficiently implement these new forms and demonstrate the utility of these versatile modules for future implementations.

C. THESIS STRUCTURE

This thesis presents the results obtained from tests of the various finite element formulations. The results are compared to those from the primitive model written by Kelley (1976). Accompanying the results is a detailed discussion of the reformulation and implementation process.

Chapter II of the thesis presents a tutorial of the finite element method and the area coordinates system used in the evaluation of the element integration. The Galerkin finite element method used in all the models is developed and applied to the advection equation in one dimension.
Chapter III presents the detailed description of the vorticity-divergence shallow water model. Here the equations are shown and written using the Galerkin method. A discussion of the computational technique used is presented along with the model’s physical parameters.

Chapter IV presents a descriptive overview of the computer implementation. The chapter includes a list of options available for testing, a brief description of the matrix compaction technique and the formulations of the versatile modules used to implement the complex equations.

Chapters V through VII discuss the results obtained from the different experiments. Chapter V briefly describes the primitive model used for all comparisons and the results from changing the element shape to equilateral triangles. Chapter VI reformulates the primitive model so that the geopotential is staggered with respect to the velocity variable. For simplicity, the continuity equation is also linearized. Chapter VII compares the results from the vorticity-divergence model developed in Chapter III to those from the primitive model.

The last chapter summarizes the results from all the experiments and identifies what areas require follow on work. The source code for the vorticity-divergence model is presented in Appendix A.
II. FINITE ELEMENTS

As is often the case with an original development, it is rather difficult to quote an exact date on which the finite element method was invented, but the roots of the method can be traced back to these separate groups: applied mathematicians, physicists and engineers. Since the early developments of the finite element method, a large amount of research has been devoted to the technique. However, the finite element method obtained its real impetus by the independent developments carried out by engineers. Its essential simplicity in both physical interpretation and mathematical form has undoubtedly been as much behind its popularity as is the digital computer which today permits a realistic solution of even the most complex situations.

The name "finite element" was coined in a paper by P.W. Clough, in which the technique was presented for plane stress analysis, as discussed in Bathe (1976). While finite element methods have made a deep impact via the field of solid mechanics, where it can be said that today they represent the generally accepted method of discretizing continuum problems for computer-based solution, the same appears not to be true in fluid mechanics or atmospheric prediction.
Numerous finite element formulations are currently available. Strang (1973), Norrie (1973) and Zienkiewicz (1971) present detailed theoretical discussions of each. The Galerkin method, the most popular finite element method, is described in detail below and used in the equation formulation later.

A. FINITE ELEMENT CONCEPT

The problem of solving partial differential equations can be specified in one of two ways. In the first, finite difference methods specify the dependent variables at certain grid points in space and time, and the derivatives are evaluated using Taylor series approximations. Secondly, the calculus of variation requires the minimization of a functional over a domain, where a functional is defined as a variational integral over the domain.

The calculus of variation approach creates a purely physical model where the functional equivalent to the known differential equations are known. Its major disadvantage is that it limits the method only to those problems for which functionals exist. Finite element methods, an extension of this method, derive mathematical approximations directly from the differential equations governing the problem. The advantage here is that it extends the method to a range of problems for which a functional may not exist, or has not been discovered.
The finite element method divides the domain into subdomains or finite elements (usually of the same form). Nodes are located along the boundary of the elements, usually at the element vertices and at strategic positions (midside, centroid, etc.) in the interior and on the sides of faces of elements.

Commonly used elements are triangular, polygonal or polyhedral in form for two-dimensional problems. The choice of elements depends on the type of problem, the number of elements desired, the accuracy required and the available computing time. To begin with, the element must be able to represent derivatives of up to the order required in the solution procedure, and to guarantee continuous first derivatives across the element boundaries to avoid singularities. Triangular elements are employed in this thesis because they can be used effectively to represent irregular boundaries, and/or geometry, and also to concentrate coordinate functions in those regions of the domain where rapidly varying solutions are anticipated.

Consider the problem of solving approximately the differential equation

\[ L(u) = f(x) \quad \text{II-1} \]

where \( L \) is a differential operator, \( u \) the dependent variable, and \( f(x) \) is a specified forcing function. Suppose that II-1 is to be solved in the domain \( a \leq x \leq b \) and that
appropriate boundary conditions are provided. The residual $R$ is formed from $II-1$ as follows:

$$L(u) - f(x) = R$$ \hspace{1cm} II-2

The critical step is to select a trial family of approximate solutions (the members of a trial family are often called basis functions). The basis function is prescribed (functionally) over the domain in a piecewise fashion, element by element, and are generally a combination of low order polynomials. A one dimensional example is shown in Figure 1, wherein the domain ($x$ axis) is divided into six elements (line segments) A through F. The basis functions are linear and one is shown for node 4 only in Figure 1. The function has a value of unity over node 4, and decreases linearly to is zero at nodes 3 and 5 and zero elsewhere.

Consider a series of linearly independent basis functions $V_j(x)$, as in Figure 1. Now $u(x)$ can be approximated with a finite series as follows:

$$u(x) = \sum_j \phi_j V_j(x) = \phi_j V_j$$ \hspace{1cm} II-3

where $\phi_j$ is the coefficient of the $j$th basis function and has a value equal to $u$ at node $j$.

Substituting this approximate solution $II-3$ wherever $u$ appears in the differential equation $II-1$

$$L(\phi_j V_j) - f(x) = R$$ \hspace{1cm} II-4
Figure 1. Piecewise linear basis function.
The best solution will be one which in some sense reduces the residual $R$ to a minimum value at all points in the domain. By definition, the residual obtained using the exact differential equation is identically zero everywhere. The residual $R$, formed in equation II-4, is minimized when multiplied with a weighting function, integrated over the domain and set equal to zero. This process is known as the weighted residual method

\[ \int_{a}^{b} RW \, dx = 0 \quad \text{II-5} \]

where $W$ is the weighting function and is referred to as the test function in the following development. The weighted residual method minimizes the errors of the residuals, such that the summation of all the positive and negative errors add to zero.

The Galerkin method, the most popular finite element method, is more general in application and is a special case of the method of weighted residuals, as discussed by Pinder and Gray (1977). The requirement imposed on the weighted residual method forming the Galerkin method is:

* the test (weighting) function be equal to the basis (trial) function $W = V$. This process leads in general to the best approximation of the solution.
The final Galerkin form is obtained by substituting II-4 into II-5, yielding

\[ \int_{a}^{b} \hat{W}_i L(\phi_j \gamma_j) dx - \int_{a}^{b} W_i f(x) dx = 2 \]  

If this procedure is repeated for \( N \) points in the domain, a system with \( N \) equations and \( N \) unknowns will be generated.

3. GALERKIN APPLICATION

The following example taken in part from Haltiner and Williams (1980) applies the Galerkin method to the advection equation with linear elements

\[ \frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 2 \]  

This equation is dependent in both time and space. The treatment of time variation is important for most meteorological prediction problems. The Galerkin method is not applied to the time dependence because it is more convenient to use finite differences in time, as is done with this example later. The same treatment is applied to the prognostic equations later, where two finite differencing methods are employed to do the time integration.

The Galerkin procedure represents the dependent variable \( u(x,t) \) with a sum of functions that have the prescribed
spatial structure as in Figure 1. Approximate \( u(x,t) \) with the finite series as follows

\[
u(x,t) = \sum_{j=1}^{J} \phi_j(t) v_j(x) = \phi_j v_j
\]

where the coefficient \( \phi_j(t) \), a function of time, is the scalar value of \( u \) at node \( j \). The basis functions, \( v_j(x) \), are functions of space only and \( j \) equals 1 to 7 for the example in Figure 1. The repeated subscript in this form implies a sum over the repeated subscript.

The Galerkin equation for the advection equation II-7 is obtained by setting \( L = c(\partial u/\partial x) \) and substituting in the approximate solution II-8 wherever \( u \) is found.

\[
\sum_{j=1}^{N} \frac{\partial \phi_j}{\partial t} \int_{a}^{b} v_j v_i \, dx + c \sum_{j=1}^{N} \phi_j \int_{a}^{b} \frac{\partial v_j}{\partial x} v_i \, dx = 0
\]

where \( i = 1 \) to \( N \), \( v_i \) the test function and \( v_j \) the basis function. The domain of integration is given by \( a \leq x \leq b \), and the integration is done in a piecewise fashion, element by element.

In this one-dimensional case, an equation like II-9 is written for each node. Considering node 4, what are the possible non-zero contributions from equation II-9? Figure 2 illustrates the basis and the test function interaction during the piecewise integration process. From the
Figure 2. Basis and test function interaction during the piecewise integration process.
definition of the basis and the test function, locally
defined as unity at node \( j \) and linearly decreasing to zero
at \( j \pm 1 \) and zero elsewhere, the only non-zero
contributions are made when \( j = 3 \) over element \( C \), \( j = 4 \) over
elements \( C \) and \( D \), and \( j = 5 \) over element \( D \).

The evaluation of II-9 for \( i = m \), which is given in
Haltiner and Williams (1981), leads to the equation:

\[
\frac{1}{c} \frac{d}{dt} \left( u_{m+1} + 4u_m + u_{m-1} \right) + \frac{c}{2\Delta x} \left( u_{m+1} - u_{m-1} \right) = 0 \quad \text{II-10}
\]

The boundary points, which in this example are nodes 1
and 7, are evaluated in the same way as the interior nodes,
with the exception that cyclic conditions are imposed.

The time discretization of II-10 is done using a finite
difference scheme. Applying leapfrog time differencing gives
the following equation

\[
\frac{1}{12\Delta t} \left( u^{n+1}_{m+1} - u^{n-1}_{m+1} + 4(u^{n+1}_m - u^{n-1}_m) + u^{n+1}_{m-1} - u^{n-1}_{m-1} \right) \\
+ \frac{c}{2\Delta x} (u^n_{m+1} - u^n_m) = 0 \quad \text{II-11}
\]

The resultant equation set, in matrix form, contains an
\( NxN \) matrix where \( N \) is the number of nodes.

The transition from one-dimension to two is
mathematically identical. The domain is now subdivided into
finite areas, which are triangles in this implementation and
the basis functions are linear. However, now they are pyramid shaped with value unity at the center and decrease to zero at the surrounding nodes, and are zero elsewhere. Figure 3 shows this basis function for node 28 outlined in heavy black. The value at any node again can be approximated by \( \ii-3 \), where \( j \) ranges over all nodes connected to node \( i \) including \( i \) itself. The connectivity for node \( i = 28 \) in Figure 3 is \( j = 15, 16, 27, 28, 29, 39 \) and 40.

The integration is still over the entire domain. With both the basis and the test function zero over the domain, except locally over each element, the global integration can be performed by integrating locally over each element. By definition, this integration can be expressed as an inner product of both functions (i.e. basis, test) as follows:

\[
\langle v_j, v_1 \rangle = \ii-12
\]

Using this definition and the repeated subscript notation equation \( \ii-9 \) becomes

\[
\partial_j \langle v_j, v_1 \rangle + c \partial_j \langle v_j, v_1 \rangle = 0 \quad \ii-13
\]

where the dot implies differentiation with respect to time, and the second subscript implies differentiation with respect to the second subscript. The local integration may be calculated directly from exact expressions derived from area coordinates described in detail in the next subsection.
Figure 3. Basis function for node 28. The shaded area is the complete basis function and the $v_j$, where $j = 15, 16, 27, 28, 29, 39, 40$ are $j$th node basis functions for node 28. The dashed line at node 28 has length unity.
In summary, the Galerkin procedure involves subdividing the domain into finite elements, approximating the dependent variables by a linear combination of low order polynomials and substituting them into the equations. The equation is multiplied by a test function, integrated over the entire domain and finally the resulting system of equations is solved.

C. AREA COORDINATES

While the Cartesian coordinate system is the natural choice of coordinates for most two dimensional problems, it is not convenient when working with triangularly shaped elements. It is therefore necessary to define a special set of normalized coordinates for a triangle. Area, or natural coordinates as they are commonly called, reduce the formidable task of integrating products between the basis and test functions and their derivatives over a triangular element and result in easily computable and exact expressions.

The following development is taken in part from the formulation by Zienkiewics (1971). Consider the triangular element illustrated in Figure 4. There is a one-to-one correspondence between the Cartesian coordinates (X,Y) and the area coordinates \((L_1,L_2,L_3)\) for the element. Let \(A\) denote the area of the triangle and \(A_1\), \(A_2\) and \(A_3\) the areas of the subtriangles in Figure 4 such that \(A = A_1 + A_2 + A_3\).
Fig 4. Cartesian vs. area coordinates

\[ \begin{align*}
& (0,0,1) \\
& (x_1, y_1) \\
& (1,0,0) \\
& (x_2, y_2) \\
& (0,1,0) \\
& A_1 \\
& A_2 \\
& A_3 \\
& P(X,Y) \text{ or } P(L_1, L_2, L_3)
\end{align*} \]

Fig. 5. Transformation to area coordinates

\[ \begin{align*}
& a_1 = x_3 - x_2 \\
& a_2 = x_1 - x_3 \\
& b_1 = y_2 - y_3 \\
& b_2 = y_3 - y_1 \\
& b_3 = y_1 - y_2 \\
& a_3 = x_2 - x_1
\end{align*} \]
The relationship between a point \( P(X,Y) \) in Cartesian coordinates and \( P(L_1, L_2, L_3) \) in area coordinates can be seen by the following transformations

\[
\begin{align*}
X &= L_1X + L_2X + L_3X \\
Y &= L_1Y + L_2Y + L_3Y \\
1 &= L_1 + L_2 + L_3 \\
\end{align*}
\]

where

\[
\begin{align*}
L_1 &= \frac{A_1}{A}, \quad L_2 = \frac{A_2}{A} \quad \text{and} \quad L_3 = \frac{A_3}{A} \\
\end{align*}
\]

and

\[
\begin{align*}
L_1 &= \frac{(2A + b_1X + a_1Y)}{2A} \\
L_2 &= \frac{(2A + b_2X + a_2Y)}{2A} \\
L_3 &= \frac{(2A + b_3X + a_3Y)}{2A} \\
\end{align*}
\]

where \( 2A \) is twice the area of the triangle and the \( a \)'s and \( b \)'s are defined as in Figure 5.

It is worth noting that every tuple \( (L_1, L_2, L_3) \) corresponds to a unique pair \( (X,Y) \) of Cartesian coordinates.

In Figure 4, \( L_1 = 1 \) at vertex 1 and \( 0 \) at vertices 2 and 3. A linear relation exists between the area and Cartesian coordinates which implies that values for \( L_1 \) vary linearly over the triangle with a value one at vertex 1 and a value of zero at vertices 2 and 3; and similarly for \( L_2 \) and \( L_3 \).

This demonstrates how each component in the tuple \( (L_1, L_2, L_3) \) behaves over the triangle as do the linear basis and test functions over the element, as was seen in Figure 4. Clearly

\[
L_1 = v_1 \\
\]

II-16
where $V_1$ is a linear function of the Cartesian coordinates (i.e. basis, test).

Zienkiewicz (1971) shows that it is possible to integrate any polynomial in area coordinates using the simple relationship

$$\int \int \frac{m! \ n! \ p!}{(m + n + p + 2)!} 2A$$

where $m$, $n$, and $p$ are positive integers and $A$ is the elementary area. For an example of this integration technique using inner product notation, equation II-12 is evaluated as follows

$$\langle V_j, V_i \rangle = \left\{ \begin{array}{ll}
\int \int_{A} V_i^2 \, dxdy = \frac{2! \ 0! \ 0!}{(2 + 0 + 0 + 2)!} 2A = \frac{A}{6} & i = j \\
\int \int_{A} V_j V_i \, dxdy = \frac{1! \ 1! \ 0!}{(1 + 1 + 0 + 2)!} 2A = \frac{A}{12} & i \neq j
\end{array} \right.$$

The differential operations in area coordinates follow directly from the differentiation of (II-15) where

$$\frac{\partial}{\partial x} = \sum_{i=1}^{3} \frac{1}{2A} \frac{\partial}{\partial L_i}$$

and

$$\frac{\partial}{\partial y} = \sum_{i=1}^{3} \frac{1}{2A} \frac{\partial}{\partial L_i}$$
As explained earlier (see Equation II-16), \( V_1 \) is a linear function (i.e. basis, test) which equals a component \( L_1 \) of the area coordinate tuple. Therefore

\[
\frac{\partial V_j}{\partial L_1} = \begin{cases} 
0 & \text{if } i \neq j \\
1 & \text{if } i = j 
\end{cases}
\]  

II-21

Consequently \( \partial V_j \cdot \partial x \) for \( j = 1 \) is

\[
\frac{\partial V_j}{\partial x} = \frac{b_1}{2A} \frac{\partial V_1}{\partial L_1} + \frac{b_2}{2A} \frac{\partial V_1}{\partial L_2} + \frac{b_3}{2A} \frac{\partial V_1}{\partial L_3} = \frac{b_1}{2A} 
\]

II-22

As an example, consider the inner product \( \langle V_{jx}, V_1 \rangle \) at vertices \( j = 2, i = 1 \). This integration is evaluated as

\[
\langle V_{2x}, V_1 \rangle = \int \int_\Delta \frac{b_2}{2A} V_1 \ dx \ dy 
\]

II-23

\[
= \frac{b_2}{2A} \frac{1! \ 0! \ 0!}{(1 + 0 + 0 + 2)!} 2A = \frac{b_2}{6} 
\]

Therefore any inner product in the formulation can be readily evaluated using area coordinates. Another benefit of using this coordinate system is that all of the inner products are functions of space only and need be computed only once.
III. SHALLOW WATER MODEL

The governing equations for this model are derived by making several simplifying assumptions on the primitive equations of motion, which then give the barotropic shallow water equations. However, as mentioned previously, the shallow water equations describe many significant features of the large-scale motion of the atmosphere, and therefore have been used in numerous experiments over the years.

The vorticity-divergence form of the equations has several advantages. Williams (1981) has shown that the geostrophic adjustment process is treated much better with the vorticity divergence formulation than with a direct treatment of the primitive form of the shallow water equations, such as was used by Kelley (1976). This formulation also allows the velocity components and the height to be carried at the same nodal points, whereas the best scheme for the primitive form of the equations requires staggering of the fields, as seen in Schoenstadt (1980). The vorticity divergence form of the equations is also convenient for the application of semi-implicit differencing, which saves considerable computer time.
A. GOVERNING EQUATIONS

The primitive form of the shallow water equations in Cartesian coordinates is

\[
\frac{\partial \phi}{\partial t} + D \phi = - \frac{\partial}{\partial x} (\phi u) - \frac{\partial}{\partial y} (\phi v) \quad \text{III-1}
\]

\[
\frac{\partial u}{\partial t} = - \frac{\partial \phi}{\partial x} + \gamma v - \frac{\partial K}{\partial x} \quad \text{III-2}
\]

\[
\frac{\partial v}{\partial t} = - \frac{\partial \phi}{\partial y} + \gamma u - \frac{\partial K}{\partial y} \quad \text{III-3}
\]

Equation (III-1) is the continuity equation and the III-2 and III-3 are the momentum equations, respectively. The variables are defined as follows:

- \(x, y\) - the spatial coordinates of the domain
- \(u, v\) - components of the wind vector
- \(\phi\) - geopotential = (gravity x free surface height)
- \(\bar{\phi}\) - mean geopotential = 49,000 meters\(^2\)/seconds\(^2\)
- \(t\) - time
- \(K\) - kinetic energy
- \(\zeta\) - absolute vorticity = (\(\Omega + f_0\))
- \(\Omega\) - relative vorticity
- \(f_0\) - coriolis force (mid-channel f-plane)
- \(D\) - divergence

The shallow water equations can be written in vorticity divergence form as follows:
\[ \frac{\partial \phi}{\partial t} + D\phi = -\frac{\partial}{\partial x}(\beta u) - \frac{\partial}{\partial y}(\beta v) \quad \text{III-4} \]

\[ \frac{\partial \xi}{\partial t} = -\frac{\partial}{\partial x}(\nu C) - \frac{\partial}{\partial y}(-\nu C) \quad \text{III-5} \]

\[ \frac{\partial \phi}{\partial t} + \frac{\partial}{\partial y} \phi = -\frac{\partial}{\partial x}(\nu C) - \frac{\partial}{\partial y}(u C) - v^2 \eta \quad \text{III-6} \]

where III-4 is the same continuity equation as III-1, III-5 is the vorticity equation and III-6 is the divergence equation.

Because of the vorticity divergence form of the equations, it becomes necessary to solve the time dependent variables \( \xi \) and \( \eta \) in terms of \( \psi \), the stream function (rotational part of the wind), and \( \zeta \), the velocity potential (divergent part of the wind). The initial fields for the model will be in terms of \( \psi, \zeta \) and \( \phi \).

The following diagnostic relationships are defined and used later in the solution of the equation set.

\[ u = -\psi_y + \zeta_x \quad \text{III-7} \]

\[ v = \psi_x + \zeta_y \quad \text{III-8} \]

where the subscript implies differentiation,

\[ K = \frac{u^2 + v^2}{2} \quad \text{kinetic energy, III-9} \]

\[ uC = u(\zeta + \xi) \quad \text{III-10} \]

\[ vC = v(\zeta + \xi) \quad \text{III-11} \]

\[ \alpha = \phi u \quad \text{III-12} \]

\[ \beta = \phi v \quad \text{III-13} \]
\[ s = r^2 \psi, \quad \text{III-14} \]
\[ d = r^2 \chi, \quad \text{III-15} \]

3. EQUATION FORMULATION

The Galerkin method described in Chapter II is now applied to equations III-4 through III-15. For ease of comprehension, the shorthand inner product notation as in II-12 will be used to simplify the equations. The detailed Galerkin formulation will be shown for equation III-7, the \( u \) component of motion. The method follows directly from the example in Chapter II of this thesis, which in turn follows in part from Kelley (1976) and Saltiner and Williams (1981).

Consider equation III-7 and assume that each variable \( u \), \( \psi \) and \( \chi \) is approximated by

\[
u = u_j v_j, \quad \psi = \psi_j v_j, \quad \chi = \chi_j v_j, \quad \text{III-16}
\]

where the repeated subscripts indicate summation over the range of the subscript. Substituting these approximate solutions into III-7 yields

\[
u_j v_j = \frac{\partial}{\partial y} (\psi_j v_j) + \frac{\partial}{\partial x} (\chi_j v_j) \quad \text{III-17}
\]

Since only the basis function \( v_j \) is a function of space, III-17 may be further simplified by factoring out the time dependent coefficients.
The next step requires multiplying by a test function \( v_1 \) as discussed in Chapter II, and integrating over the area domain

\[
\begin{align*}
\int_A u_j \int_A v_j v_1 \, dA &= - \int_A \int_A \frac{\partial v_j}{\partial y} v_1 \, dA \\
&+ \int_A \int_A \frac{\partial v_j}{\partial x} v_1 \, dA \\
&= III-12
\end{align*}
\]

The final form in inner product notation is

\[
\langle u_j v_j, v_1 \rangle = - \langle v_j v_{jy}, v_1 \rangle + \langle x_j v_{jx}, v_1 \rangle III-19
\]

where the double subscript implies differentiating with respect to the second subscript.

The three prognostic equations (III-4, III-5 and III-6) are similarly advanced using the Galerkin technique to become, respectively:

\[
\begin{align*}
\langle \dot{\phi}_j v_j, v_1 \rangle + \dot{\phi} \langle \Sigma_j v_j, v_1 \rangle &= - \langle \alpha_j v_{jx}, v_1 \rangle - \langle \beta_j v_{jy}, v_1 \rangle III-20 \\
\langle \dot{\gamma}_j v_j, v_1 \rangle &= - \langle (uQ)_j v_{jx}, v_1 \rangle - \langle (vQ)_j v_{jy}, v_1 \rangle III-21 \\
\langle \dot{\Sigma}_j v_j, v_1 \rangle - \langle \phi j v^2 j, v_1 \rangle &= \langle (vQ)_j v_{jx}, v_1 \rangle - \langle (uQ)_j v_{jy}, v_1 \rangle \\
&+ \langle x_j v^2 j, v_1 \rangle III-22
\end{align*}
\]

where \( v^2 \) is the Laplacian operator and the dot implies differentiation with respect to the time dependence in III-4, III-5 and III-6.

Similarly, Galerkin equations are formulated for equations III-7 through III-15.
C. TIME DISCRETIZATION

The equation set III-20, III-21 and III-22 is arranged so that all the terms on the left hand side can be treated implicitly, and all the terms on the right hand side can be treated explicitly. The explicit time integration will be done by the leapfrog difference method. To start the time integration, two forward half steps are taken, after which the full leapfrog scheme is used for the remainder of the forecast period.

The vorticity equation III-21 is solved independently from III-20 or III-22. However, III-20 and III-22 (continuity and divergence equations, respectively) are coupled. To explicitly solve either, decoupling of the equations is necessary. In this thesis this is done through algebraic substitution of III-22 (solved for $D(n+1)$) into III-20. Once the time integration is performed on III-20, III-22 can be solved for $D(n+1)$ using the $\phi(n+1)$ value.

The final prediction equations are

$$
\phi^*_j = \frac{1}{\rho} \left[ <V_{jx}, V_{ix}> + <V_{jy}, V_{iy}> + C <V_{jx}, V_{iy}> - [\Omega_j]_{n+1} - [\Omega_j]_n - 1 \right. \\
+ Cg_j^{n-1} <V_{jx}, V_{ix}> - A_j^{n-1} <V_{jy}, V_{iy}> \\
- \phi_j^{n-1} [ <V_{jx}, V_{ix}> + <V_{jy}, V_{iy}> ] \\
- 2[(\nabla J)^R_j <V_{jx}, V_{ix}> - (u\Omega_j)^R_j <V_{jy}, V_{iy}> ] \\
- 2k_j^R [ <V_{jx}, V_{ix}> + <V_{jy}, V_{iy}> ] \\
- 3[(\phi u)^R_j <V_{jx}, V_{ix}> + (\phi v)^R_j <V_{jy}, V_{iy}> ] \right)
$$

III-23
where \( A = \frac{4}{(2at)} \), \( B = \frac{A}{\dot{\Phi}} \), \( C = \frac{B}{(2at)} \) and \([BRRT]\) is the geostrophic boundary contribution, see Section 3.

\[
\begin{align*}
S^{n+1}_j \langle v_j, v_1 \rangle &= S^{n-1}_j \langle v_j, v_1 \rangle \\
&\quad - 2at [(u\zeta)_j^n v_{jx} v_1^* + (v\zeta)_j^n v_{jy} v_1^*] \quad \text{III-24}
\end{align*}
\]

\[
\begin{align*}
D^{n+1}_j \langle v_j, v_1 \rangle &= D^{n-1}_j \langle v_j, v_1 \rangle + (at/2) [\zeta_j^{n+1} v_{jx}^2 v_1 - \zeta_j^{n-1} v_{jx}^2 v_1 + 2(v\zeta)_j^n v_{jy} v_1^* - 2(u\zeta)_j^n v_{jy} v_1^* + 2c_j^n v_{jx} v_1^*] \quad \text{III-25}
\end{align*}
\]

After these three elliptic equations are solved, the history of the variables III-7 through III-15 is updated.

A large time step can be applied to this form of the shallow water equations due to the semi-implicit nature of the equations. This is very important since finite element methods generally require more computer time per time step. The vorticity-divergence formulation acts as a filter, which slows down the high frequency waves in the solution. The two-dimensional advective stability criterion for a linear element, derived by Cullen (1973), was used to determine the correct time step.

\[
\Delta t = \frac{4x}{|c| \sqrt{\epsilon}} \quad \text{III-26}
\]

where \( \Delta t \) is the time step in seconds, \( 4x \) the shortest grid spacing in meters and \( c \) the fastest phase velocity.
D. COMPUTATIONAL TECHNIQUES

The final prognostic equation set requires the solution of a Helmholtz equation for $\varphi$ and Poisson equations for $Z$ and $\Psi$. The most common method of solution used by meteorologists has been the successive over relaxation method (SOR) in which an initial guess of the solution is made and then progressively improved until an acceptable level of accuracy is reached. SOR is employed in the solution of the equations, where III-23 can be represented by

$$v^2[M]{x} - C[M]{x} = \{b\} \quad \text{III-27}$$

and III 24, III 25 by

$$v^2[M]{x} = \{b\} \quad \text{III-28}$$

where $v^2$ the Laplacian operator, $[M] = \langle V_j, V_i \rangle$ matrix, $\{x\}$ the dependent variable in vector notation, $C$ - constant as in III-23 and $\{b\}$ the right hand side of the equation or the forcing function.

The mass matrix $[M]$, dimensioned $(n \times n)$, is a matrix of coefficients whose rows are the equations of the system to be solved. There exists a one to one correspondence between the rows of the mass matrix and the nodes of the domain. Each equation has a term (column) for each node, where a non-zero term represents connectivity. Nodes are connected if they are both vertices of the same element. Obviously $[M]$ is a sparse matrix containing the inner products for the
left hand side. Chapter 4 of this thesis will describe the matrix compaction procedure.

The forcing function \( \{b\} \), dimensioned \((n \times 1)\), involves only variables at the current time step and is easily computed using four very versatile subroutines described in detail in the next chapter.

The initial guess to start SOR is the previous time step solution. An average of 30 passes per equation are needed for each time step. The solution is considered to have converged to its final value when the residual for each node has been reduced to some acceptably small value.

The diagnostic equations III-7 through III-15 must also be solved every time step. However, the same technique is not used for these equations. Dr. "J.P. Cullen suggested an under relaxation scheme for which three passes over the domain should produce a solution of acceptable accuracy, since the coefficient matrix is so strongly diagonally dominant. Mass lumping of the coefficient matrix is used for the first guess. This technique requires replacing the mass matrix \([m]\) by the identity matrix \([I]\). A first guess of this type is able to describe most of the large scale features, which in turn reduces the number of iterative passes over the field. Successive passes converge to solutions which describe smaller scale motion, approximately to the same order of magnitude as introduced by computational error, so that further iterations are not needed.
I. GRID GEOMETRY

The domain of this model is a cylindrical channel, with total length of 4045 Km and width of 3523 Km. The channel simulates a belt around the earth and it proves to be an excellent test bed for comparing with the finite element formulations used by Kelley (1976) and Older (1981).

The domain is subdivided into equilateral triangles as shown in Figure 6. Most of the test runs for this thesis use a 12x12 mesh which has 156 nodes and 289 elements. This implementation is not restricted to one grid pattern. The technique developed by Older (1981) to vary the nodal geometry smoothly to achieve areas of denser and coarser resolution is also implemented, as in a third grid pattern that varies the nodal geometry abruptly. A short discussion of these nodal geometries with accompanying illustrations of each is presented in Chapter VII, where the different test cases are described.

Cyclic continuity is assured in the x direction by wrapping the domain around the earth to form a cylindrical domain. This has the advantage of eliminating the east-west boundaries and it simulates the flow around the earth. The only boundaries on this domain are the north-south walls and their treatment will be discussed shortly.
F. INITIAL CONDITIONS

As mentioned previously, the reformulation of the governing equations into the vorticity-divergence shallow water equation set requires solving the time dependent variables in terms of the stream function and velocity potential. The continuity equation is not altered, so that its solution is expressed in terms of $\phi$.

For the basic testing of the model's performance, simple analytic sinusoidal initial conditions are used to insure the most accurate analysis possible and to simplify the comparisons.

The sinusoidal initial fields are graphically shown in Figure 7 as 3-dimensional surfaces. The geopotential field $\phi$ consist of a half sine wave in the $y$ direction and a single cosine wave in the $x$ direction. The stream function $\Psi$, calculated by dividing the geopotential field by the coriolis force, has the same physical structure as $\phi$. The velocity potential $\Gamma$ has a single sine wave in the $x$ direction and a half sine wave in the $y$ direction.

These initial conditions are computed as follows

$$\phi = f_{0} A \sin \alpha_{1} \cos \alpha_{2} - f_{0} \tilde{\Psi}(y - y_{a}) + \frac{A}{2}$$

$$\Psi = \phi / f_{0}$$

$$\Gamma = C \sin \alpha_{1} \sin \alpha_{2} \quad \text{quasi-geostrophic divergence}$$

where

$A$ - arbitrary amplitude

$\phi$ - coriolis value for mid-channel latitude
a) $\phi$ and $\psi$ initial fields.

b) $\pi$ initial field.

Figure 3-dimensional view of the initial fields.
\( \bar{U} \)  mean flow

\( y_m \)  mid-latitude value of \( y \)

\( \bar{\eta} \)  mean free geopotential height

\[ = 49,000 \text{ m}^2/\text{s}^2 \]

\( \alpha_1 \)  \( \pi y/4 \)

\( \alpha_2 \)  \( 2\pi r/L \)

\( r \)  wave number

\( W \)  channel width

\( L \)  channel length

\( C \)  \( -(\bar{f}_0 \bar{\omega}_2 BA)/(f_0^2 + \phi B) \)

\( 3 \)  \( \alpha_1^2 + \alpha_2^2 \)

3. BOUNDARY CONDITIONS

Boundary conditions are only required on the north and south walls of the grid domain. Due to cyclic continuity, the domain is wrapped around creating a cylinder eliminating the east and west boundaries. However, careful attention to detail is needed during the implementation to assure this continuity. Separate boundary conditions are applied to each of the predictor equations III-23, III-24 and III-25. These conditions are computed for the wall nodes only and are applied during each pass through the relaxation scheme.

The vorticity equation III-24, the most sensitive of the predictor equations to solve, requires \( \Psi \) on the north-south boundaries to remain constant for the entire forecast period. Since this equation is solved in terms of \( \Psi \), the initial north-south \( \Psi \) values are saved and assigned to the
boundary points after each pass through the relaxation subroutine.

The proper boundary condition for the divergence equation III-25 would be $\partial \phi / \partial n = \gamma$. However, for the purpose of this study, there is more interest in the sinusoidal variation in the $y$ direction and not in the region of the walls. Therefore $\gamma = 2$ is appropriate.

The continuity equation III-23, the most complex predictor equation, requires that there be no mass flux through the north-south walls. The geostrophic boundary condition

$$\frac{\partial \phi}{\partial y} = -\mu_f,$$

is applied to the north-south boundary nodes for the terms [BDMT] in equation III-23. Integrating the inner product $\langle \phi_j, \nabla \phi \cdot \nabla \rangle$ by parts produces the boundary terms

$$\int \int \phi_j \nabla \phi \cdot \nabla \phi \, dxdy = \int \int \nabla \phi \cdot \nabla \phi \, dxdy$$

$$= \int \int \left[ \phi_j \nabla \phi \cdot \nabla \phi \right] - \mu_f \phi_j \nabla \phi \cdot \nabla \phi \, dxdy$$

$$= \int \int \phi_j \nabla \phi \cdot \nabla \phi \, dxdy$$

$$= [BDMT] - \phi_j [\langle \nabla \phi_j, \nabla \phi \rangle + \langle \nabla \phi_j, \nabla \phi \rangle]$$

III-31

where $\hat{n}$ is a unit vector normal to the domain and $dr$ is the differential distance along the path of integration on the perimeter of the domain.
The geostrophic boundary condition (32) is substituted into the contour integral in equation (31) and put into Galerkin form, in the same way as in the one-dimensional advective equation in Chapter II. The resulting term is derived as follows

\[ \int \mathbf{v}_1 \mathbf{v} \, d \mathbf{r} = \int \frac{\partial}{\partial y} \left( \phi_j v_j \right) \mathbf{v}_1 \, d \mathbf{x} \]

\[ = \frac{5 \Delta x}{3} \left( u_{j+1} + 2u_j + u_{j-1} \right) \quad (32) \]

Equation (32) appears twice in the continuity equation (23), for time levels \( n+1 \) and \( n-1 \). All values of \( u \) are known for time \( n-1 \), since they are saved from the previous calculations. However, \( u(n+1) \) has not been computed. To solve for \( u(n+1) \), both \( \Psi(n+1) \) and \( \chi(n+1) \) are needed. \( \Psi(n+1) \) is solved first from the vorticity equation. \( \chi(n+1) \) needs \( \phi(n+1) \) as part of its solution and \( \phi(n+1) \) needs \( u(n+1) \) in its solution. To avoid this problem, it is assumed that \( \chi(n+1) \) has a negligible contribution to the solution of \( u(n+1) \) and only \( \Psi(n+1) \) is used.
IV. COMPUTER IMPLEMENTATION

The formulation and general theory of the finite element method was presented in the previous chapters. The objective in this chapter is to discuss some important computational aspects pertaining to the implementation of the finite element prediction system.

The main advantage that the finite element method has over other prediction techniques is its generality. Conceptually, it seems possible by using many elements, to approximate virtually any surface with complex boundaries and initial conditions to such a degree that an accurate solution can be obtained. In practice, however, obvious engineering limitations arise, a most important one being the cost of the computation. As the number of elements is increased, a larger amount of computer time is required for a forecast. Furthermore, the limitations of the program and the computer may prevent the use of a large number of elements. These limitations may be due to the computer speed and storage availability, or round-off errors propagated in the computations because of finite precision arithmetic. Also, the malfunction of a hardware component, if the prediction is carried out using many computer hours to execute, can be a serious problem. It is therefore desirable to use efficient finite element programs.
The effectiveness of a program depends essentially on the following factors. Firstly, the use of efficient finite element techniques is important. Secondly, efficient programming methods and sophisticated use of the available computer hardware and software are important. The third very important aspect in the development of a finite element program is the use of appropriate numerical techniques.

The vorticity-divergence model described in the previous chapter is implemented on the I3M 3033 computer located at the Naval Postgraduate School. Some notable features of its architecture are the three trillion bytes of virtual mass storage, of which eight mega bytes are available to each user, and the 57 nanosecond machine cycle time. The model is executed mostly using a 12x12 element domain requiring 420k bytes of storage and 30 seconds of CPU time to execute. Exceeding execution time and/or available storage is not a problem, in fact the system allowed a lot of flexibility during the implementation phase of the model.

The source code is written using FORTRAN IV and compiled on an optimizing Fortran II compiler. Appendix A contains the source code listing, which is divided into five subdivisions delineating the logical structure of the program.

A. PROGRAM ARCHITECTURE

Program features incorporated in the model are:

1) Modularity. With only a few exceptions, each module is limited to one page of FORTRAN code. This makes it
easier to comprehend the program. Each module performs only one task. For example, subroutine CONTIC computes the value of the forcing terms for the continuity equation. Likewise there is also one module for the divergence and vorticity equations. To implement a new set of equations, only these modules would have to be altered.

2) Easily controllable switches. Switches may be set to either print, plot or tabulate harmonic analysis data for most of the available fields. The ability to display intermediate results allows each portion of the algorithm to be monitored for computational adjustments. This also makes it easier for unfamiliar users to become acquainted with the computational model.

3) Forcing term subroutines. In previous implementations, each forcing term was calculated by a special subroutine. In this implementation, the calculations are accomplished by general purpose routines which simplify the implementation of the complex prognostic equations. This allows implementation of different equation sets (i.e. baroclinic model) over the same domain with minimal effort.

4) Documentation. Each variable is defined by a short phrase (Appendix A, A.). The function of each module is described in an introductory paragraph. Shared data is placed in named common blocks and identified with each subroutine which uses them. A subroutine index is given.
1. Main Program

The main program is short, calling only six modules which reflect the basic sequential flow of the model. It starts with initialization of all model parameters (i.e. model options, domain, finite element arrays, inner products). It then initializes the input fields (i.e. geopotential heights, stream function and velocity potential) and is followed by initialization of all remaining dependent variables. At this time the model is totally initialized and time integration begins. As mentioned previously two techniques are employed for time integration, each having its own module. Upon completion of the last forecast, the program terminates.

Arrays are the only data structures used and are grouped using 19 different common blocks. Several arrays are used as static link lists, as described in detail later, which simplified the algorithms. The common block format has the advantage of reducing the overall execution time of the program. Most of the arguments passed during a call to a subroutine are contained in common. This requires less time to execute since no parameter passing is required for the arguments. Another benefit of this format is that the code becomes less cumbersome and more readable. Each variable and array is defined in the first subsection of Appendix A along with a page index for the subroutines.
2. Initialization Phase

Appendix A. Section C contains all the subroutines used during the initialization phase of the program. From the user point of view, the most important subroutine is INITG2, the first subroutine called, which contains all the global variables that control the different options available per run. This is the only subroutine that is changed to run the different experiments, assuming that no new computational technique is introduced. The selection of options are:

1) - channel location - the channel may be shifted north or south by presetting the north/south latitude limits in INITG3.

2) - variable geometry - the node positions may be grouped for more dense node patterns to yield higher resolution. Two variables R1 and R2 set the ratio used to vary the spacing along the x and y axes, respectively.

3) - initial field wave length and amplitude can be altered to produce various effects.

4) - change the initial mean flow.

5) - diffusion can be entered for any of the three prognostic equations.

6) - maximum length of forecast period may be changed and a print, plot or harmonic
analysis of any dependent variable may be requested for any time interval.

Once the experiment is determined, the options listed above are set. The program is ready to be executed.

The largest part of the initialization phase consists of establishing the domain and producing all the finite element computational vectors that remain constant throughout the experiment.

The first several steps in setting up the domain are concerned with indexing. Subroutine CORES is called first, where all the nodes (grid points) and elements (triangular areas) are numbered consecutively starting at the southwest corner of the domain and moving eastward across each row or latitude. For each element, a record of all of its nodes (vertices) are stored in array ELEMNT (M,3), where M is the total number of elements. To facilitate the inner product evaluation later, a local numbering system is required for each element. That is, for each element, its nodes are stored counterclockwise in a positive sense. The first node however, is arbitrary.

With the domain divided and numbered, a connectivity list (the correspondence between each node and the neighbor nodes) is constructed for each node by subroutine CORREL. Each node is adjacent to four or six other nodes depending on whether it is a boundary or interior node, respectively. These adjacent nodes, plus itself, make up the connectivity
list for one node. The connectivity lists are then concatenated sequentially starting with the first nodes connectivity list into the vector \( \text{NAME} (NN) \), where \( NN \) is the sum of the nodes in each connectivity list. (i.e. for a 12x12 domain with 156 nodes, and equilateral elements \( NN = 1044 \)). For the first time during the initialization phase, special attention is given to cyclic continuity. As discussed earlier, cyclic continuity is the joining of the east and west boundaries to create a cylindrical channel. The connectivity list for these east/west boundary nodes must be complete to insure proper continuity for the calculations later.

The connectivity vector \( \text{NAME} \) is frequently used during most computations. Two utility vectors \( \text{ISTART} \) (containing the starting location in \( \text{NAME} \) for a particular node) and \( \text{NUM} \) (containing the number of nodes in its connectivity list) are used to locate and index through the vector \( \text{NAME} \), as will be seen shortly. This same technique is used to index through the coefficient matrices and used during most of the node interaction computations.

The physical properties of the channel are calculated next in subroutine \( \text{CHANAL} \). Here the north and south latitude boundaries, which were pre-set in \( \text{INITGE} \) by the user, are used to compute the grid spacing along the \( x \) and \( y \) axis. Since this channel simulates a belt around the earth, the magnitudes of both \( \text{DELTAX} \) and \( \text{DELTAY} \) (meters) are
proportional to the width of the channel divided by the number of grid points in the y axis.

The Cartesian coordinates for each node are computed by subroutine LOCATE using the DELTAX and DELTAY calculated in CEANAL. If the option to use varying grid geometry is desired, subroutine TRANS transforms the grid geometry. TRANS also computes the corresponding new Cartesian coordinate values for each grid point and calculates the minimum DELTAX and DELTAY within the domain. When the geometry is changed to create a smaller DELTAX or DELTAY, the two dimensional advective stability criteria is also changed. A new time step DT has to be computed using equation III-26. Since TRANS transforms the geometry, it also computes the new DT.

Another transformation is required as discussed in Chapter II. The transformation from Cartesian coordinates to area coordinates is needed to perform the area integration of the inner products. Subroutine AREA computes these transformations exactly as outlined in Chapter II, Section C. Again cyclic continuity is very important and special care is needed to insure proper transformation.

Following the area transformation is the computation of all the inner products that are required to solve the equations. The advantage of using area coordinates is that the inner products (function of space coordinates only) are computed and stored once and used repeatedly without
recalculation. Subroutine INAER computes and stores these products using the formulas derived in Chapter II.

The coefficient matrix, dimensioned N x N, where N is the total number of nodes, is a matrix of coefficients whose rows are the equations of the system to be solved. As discussed in the computational technique section, the members of this sparse matrix are the inner products for the left hand sides of the equations. Three coefficient matrices are used in the solution of the equations. The diagnostic equations (III-7 through III-15) use a coefficient matrix with the inner product \( \langle V_j, V_i \rangle \) which is constructed by subroutine AMTRI1 and stored in compacted form in vector G(NN) by subroutine ASEM3L. However, when solving the prognostic equations, these coefficient matrices have a DT (time step in seconds) term, so that these matrices are not assembled until the time integration begins. The vorticity and divergence equations (III-24, III-25) use the coefficient matrix \( F(NN) \) with inner products \( \langle V_j \cdot V_i \rangle \) and \( \langle V_j \cdot V_i \rangle \) in solving the Poisson equations for the stream function and velocity potential, respectively. The continuity equation (III-23) uses a combination of inner products in its coefficient matrix \( F(NN) \) as follows

\[
\langle V_j \cdot V_i \rangle + \langle V_j \cdot V_i \rangle + \frac{4}{\phi(at)^2} \langle V_j, V_i \rangle
\]  

IV-1
to solve the Helmholtz equation. At the start of each time integration module, subroutine AMTRX2 is called to construct the two coefficient matrices $H$ and $F$.

These banded and sparse matrices are compacted into vectors to save storage during their assemblage by ASIM3L. The vectors are dimensioned $NN$, as is $NAME$, the connectivity vector, and both use $ISTART$ and $NUM$ to index through them. This compaction routine was used by Kelley (1976) and Older (1981) in their models, but was developed by Hinsman (1975).

To illustrate matrix assemblage using an element-by-element technique, consider Figure 8. Note that this illustration is for element number 3, but all elements are treated in a similar manner. The computational technique required that for each point (node) describing element 3, namely nodes 2, 3 and 14 stored in array $ELEMENT$, the inner product $\langle V_j, V_1 \rangle$ between those points be distributed to their proper location in the coefficient matrix.

Subroutine AMTRX1 builds the inner product nodal interaction and stores it in matrix 3, dimensioned $3x3$. Figure 8 illustrates the $B$ matrix for element 3, where the inner product $\langle V_j, V_1 \rangle$ is the multiplicand of the corresponding basis and test functions, respectively.

The local dispensing of interactions is done in ASIM3L. Consider the second row of $[B]$ in Figure 8. These are the interactions between node 3 of element 3 to the test

61
ANRXX - builds [B] for one element, passes [B] and the element number associated with [B] to ASEML. This process continues till all element node interactions are assembled in the coefficient matrix.

\[ [B]_3 = \begin{bmatrix}
  V_2 V_2 & V_2 V_3 & V_2 V_{14} \\
  V_3 V_2 & V_3 V_3 & V_3 V_{14} \\
  V_{14} V_2 & V_{14} V_3 & V_{14} V_{14}
\end{bmatrix} \]

ASEMRL - assembles the node interactions into the coefficient matrix [C], which has the same structure as NAME. The following diagram assembles inner product \( V_j V_{14} \) into the coefficient matrix [C], for element 3.

**Figure 8.** Assembling and storing the coefficient matrix for element 3.
functions. ASMEMBL locates nodes 3's connectivity list in NAME using ISTART and NUM. In Figure 8, this list is delineated by START(3) and LAST(3). Now ASMEMBL steps through the connectivity list for three iterations. During each pass, ASMEMBL is searching for one of the three node numbers for element 3. When a match is found with one of element 3's nodes (i.e. 2,3 or 14) and node 3's connectivity list (i.e. 3,2,14,15 or 4) the proper position, to which this interaction is to be added has been found in the coefficient matrix. Since NAME and vector C, the compacted coefficient matrix, are dimensioned identically, the same pointer (i.e. J in Figure 8) is used to index through both arrays. This procedure is repeated for all elements in the domain to assemble the coefficient matrix of the equations. The pseudo-code for ASMEMBL is shown in Figure 8 to facilitate stepping through this example.

The domain and all finite element work vectors are initialized at this point. Subroutine ERMSET is called later to compute interpolation points for the harmonic analysis subroutines.

The last phase of the initialization process is the initialization of the dependent variables. The three input fields geopotential heights, stream function and velocity potential are computed in subroutine IC using the equation set III-30. However, the variables calculated from the diagnostic equations have to be computed using the input
fields. These variables are used during each time step while solving the prognostic equations.

The diagnostic equations are solved in subroutine DFVAR, first during the initialization phase and later during the time integration phase. Each diagnostic equation calls its own module to compute the value of the forcing function and stores the computed values in the vector RES. These equations all use the same coefficient matrix when solving the diagnostic equations. Subroutine SOLVER is sufficiently general to solve each equation. SOLVER uses vector RES and coefficient matrix G to under-relax towards the solution. As mentioned previously, the coefficient matrix is strongly diagonally dominant so that three passes over the domain are sufficient. At the end of DFVAR, output is generated depending on what print, plot, or harmonic analysis controls were preset.

This completes the initialization phase of the model and the program for the forecast phase will be described next.

3. Forecast Phase

The forecast phase is accomplished in two steps. The first time step is made using two half steps by subroutine MATZNO. Here the prognostic coefficient matrices are constructed using half the DT value by calling AMTRX2. AMTRX2 uses the same computational technique to construct the coefficient matrices as described for AMTRX1.
Each of the prognostic equations III-23, III-24 and III-25 calls its own subroutine (CONTEC, VORTEQ and DIVEC respectively) to compute all the terms on the right hand side, which are stored in the vector RHS. After computations for RHS are completed, subroutine RELAX solves the equations by over-relaxation as described in the computational technique in Chapter 3. Once the solutions for the \((n+1)\) time step are completed, DEPAR is called to update the variables from the diagnostic equations. Two passes through MATZNO advances the solution fields one time step.

The remainder of the forecast period is integrated using the leapfrog scheme. Subroutine LEAPFR performs this integration using the identical format as MATZNO, except that DT equals two DT. At preset times as specified in INITGE, the different fields are saved for printing. This process continues until the final forecast time is reached.

2. UTILITY MODULES

Once the equation formulation is completed, as in Chapter III, all the inner products and types of integrations are known. Versatile modules can be written to perform these computations. Consider a term of general form \(\langle A_j \, V_j, V_i \rangle\) where \(i\) is the node about which the term is evaluated and the \(j\)'s are the nodes connected to node \(i\), or the surrounding nodes. The inner product values \(\langle V_j, V_i \rangle\) are already computed and stored for all the nodes, during the initialization phase of the model.
The remaining computation to complete the evaluation of this term is the multiplication of the scalar coefficient of \( A \) at node \( i \) with the corresponding inner product \( \langle V_j, V_i \rangle \) for node \( j \). This requires indexing through node \( i \)'s connectivity list stored in vector NAME, and for each node in the list multiply and total the products. The cumulative sum of these multiplications is assigned as node \( i \)'s contribution for this term. Subroutine TERM3 performs this exact computation. All that is passed to TERM3 is the scalar field \( A \) and the sign of the inner product. TERM3 then computes the contribution for each node in the domain and accumulates it in the work vector RES.

Three other utility modules are: TERM1, which computes the first scalar multiplication for triple inner products (i.e. \( \langle A_j V_j, V_k, V_i \rangle \)). The product \( \langle V_j, V_k, V_i \rangle \), is again already computed and stored by subroutine INNER. TERM1 computes \( \langle A_j V_j V_k, V_i \rangle \) and constructs a compacted vector similar to the coefficient matrices. This reduces the effort of multiplying the second scalar to a TERM3 computation. TERM2 computes node interaction of the following type \( \langle A_j V_j, V_x \rangle \), where both the basis and the test functions are derivatives. Lastly, subroutine TERM4 computes node interaction for terms as \( \langle A_j V_j, V_i \rangle \), where only the basis function is a derivative.

When examining the right hand side of the equation sets III-23, III-24 and III-25, it is obvious this implementation is a subscripting nightmare; however, the use of the utility
modules TERM1, TERM2, TERM3 and TERM4 simplifies the implementation to only determining what order to call the utility modules. Examination of subroutines CONTEQ, VORTEQ and LIVEC, which compute the right hand sides for III-23, III-24 and III-25 respectively, illustrates this fact. No other subroutines or calculations were required. Implementation of these equations required minimal effort.
V. PRIMITIVE MODEL EXPERIMENT

The previous two chapters presented the detailed formulation and implementation of the vorticity-divergence, shallow water equation model. The results from this model will be compared with the results from the primitive model in Chapter VII. To facilitate interpretation of the comparisons, a brief description of the primitive model follows. See Kelley (1978) for a detailed discussion of the entire model.

Also presented in this chapter is an experiment which demonstrates significant improvement of the solution from the primitive model. Kelley's implementation used elements which were right triangles. Older (1981) showed that equilateral elements are far superior to triangular elements. This experiment re-implements the primitive model using equilateral elements and a comparison is made between the results of both implementations.

A. MODEL DESCRIPTION

A form of the barotropic, shallow water, primitive equations developed by Phillips (1959) is used as the governing equation set for this model. In Cartesian coordinates the equation set is
\[ \frac{\partial \phi}{\partial t} = - \frac{\partial}{\partial x} (u \phi) - \frac{\partial}{\partial y} (v \phi) \]  
\[ \frac{\partial u}{\partial t} = - \frac{\partial \phi}{\partial x} - u \frac{\partial u}{\partial x} - v \frac{\partial u}{\partial y} + v \phi \]  
\[ \frac{\partial v}{\partial t} = - \frac{\partial \phi}{\partial y} - u \frac{\partial v}{\partial x} - v \frac{\partial v}{\partial y} - u \phi \]

The finite element formulation of this set of equations evaluates the height and velocity components at the same nodal points. This is an important consideration, because the other models (the linearized model, see Chapter VI, and the vorticity-divergence model, see Chapter III) either stagger the dependent variables or have the property of a staggered formulation. When comparisons are made between the models, it is this lattice structure that is being compared.

This form of the shallow water equations includes gravity waves as a solution. Gravity waves have a maximum phase speed of about 300 meters/second. When the correct time step is calculated using equation III-2e, a considerably smaller time step is obtained compared to the larger time step permitted in the vorticity-divergence formulation. This is an important feature. If solutions from all models are equally as good, the best formulation would be determined using the computational time required to produce the desired forecast.
All models use the same domain structure. In fact, the domain described in Chapter III was patterned after the domain implemented by Kelley. Again, this domain simulates a belt around the earth, with cyclic continuity which eliminates the east-west boundaries. Rigid boundary walls exist at the equator and at 30 degrees north latitude. The domain is composed of a 12x12 point mesh and subdivided into the right triangular elements illustrated in Figure 9. Notice that the grid points are not shifted as in the equilateral element implementation shown in Figure 6.

The following boundary conditions are imposed:

1) - no cross channel flow at the latitude boundaries.

2) - a geostrophic balance at the channel walls imposed on the continuity equation V-1.

This model has a simple second order diffusive term in the equations of motion V-2 and V-3. However, for the purpose of evaluating these different formulations, this option was not implemented during the comparison phase.

Initial conditions consist of a single wave in the x direction and a half wave in the y direction. The initial fields for the three dependent variables are shown in Figure 10. The maximum zonal wind perturbation of 5.5 meters/second is superimposed on a mean zonal flow of 10 meters/second.
Figure 10. Initial fields for the primitive model. Both the x and y axes are multiplied by $10^6$ Km.
This cursory description of the primitive model mentions only those significant features that will weigh heavily in the comparisons later. The Galerkin implementation of this model is similar to that presented in Chapters II and III and the system of equations is solved using a Gauss-Seidel iterative procedure. Further details concerning this primitive model are given in Kelley (1976).

B. RESULTS

This experiment involves the shape of the elements. As mentioned previously, Kelley's implementation subdivides the domain into right triangular elements, as illustrated in Figure 9. Considerable small-scale noise was observed by Kelley in the 48 hour forecast solution.

The transition from right triangles to equilateral triangles changes the size of the domain. With right triangles, the $x$ and $y$ grid spacings are equal (300 KM). A 12x12 grid matrix has a length and width of 3600 KM. With equilateral triangles, the $x$ and $y$ grid spacings are no longer equal. Arbitrarily, the $y$ grid spacing is held constant (300 KM) and a new $x$ grid spacing computed by

$$\Delta x = \frac{\Delta y}{\cos(30)}$$

A 12x12 grid matrix with equilateral elements has a width of 3600 KM and a length of 4045 KM.

Figure 11 contains the 48 hour forecasts produced using both types of elements. The three dependent variables fields
Figure 11. 48 hour forecast comparison from the primitive model using both right triangles and equilateral triangles for element shapes. Arbitrary perturbation amplitude of 5.5 m/s.
are compared for each. The small-scale noise that Kelley observed is present. The three fields show varying degrees of distortions, which are especially noticeable along the boundaries.

Older (1981) found that the root mean square error (RMSE) was reduced 20 percent by using equilateral shaped elements. This improvement is apparent on viewing Figures 11b, d and f. The contours are smooth and the boundaries are noise free. Kelley showed excellent treatment of wave propagation by this primitive model. The lowest resolution grid ('6x6) tested by Kelley was within four percent of the true phase velocity. Changing the element shape had no apparent effect on the phase velocities.

Because the outcome of this experiment was a significantly improved forecast solution, future comparisons with the primitive model will be made using equilateral elements.
VI. LINEARIZED MODEL EXPERIMENT

The previous chapter demonstrated how the shape of the element can significantly improve the solution. Williams and Zienkiewicz (1961) used differently shaped basis functions on a linearized equation set to produce excellent solutions when applied to the geostrophic adjustment problem.

Spatial staggering of dependent variables in finite difference formulations has given much better solutions to the geostrophic adjustment process, and these forms are widely used in meteorology. Schoenstadt (1950) found similar results with finite element formulations with piecewise linear basis functions. However, staggering nodal points is not a convenient method to implement, especially in two-dimensions with irregular boundaries, so alternative schemes are needed.

The implementation of the alternative scheme introduced by Williams and Zienkiewicz (1961) are presented in this chapter. As mentioned above, this formulation uses different basis functions for the height and the velocity fields. One of the basis functions is piecewise linear, while the other is piecewise constant, as is illustrated in Figure 12 for a one-dimensional domain.
Figure 12. Different shaped basis functions.
A. EQUATION REFORMULATION

The primitive form of the shallow water equations presented in Chapter V (V-1, V-2 and V-3) is used to derive the linearized equations needed for this experiment. The velocity equations V-2 and V-3 remain unchanged and a linear basis function \( (V_7) \) is used to approximate the \( u \) and \( v \) variables.

The continuity equation V-1 is linearized as follows:

\[
\frac{\partial \phi}{\partial t} = -u \frac{\partial \phi}{\partial x} - v \frac{\partial \phi}{\partial y} - \bar{\phi} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right)
\]

where \( \bar{\phi} \) is the average geopotential over the domain. A piecewise constant basis function \( (W_j) \) is used to approximate the geopotential. This linearization is reasonable in this case because the Rossby radius of deformation \( \frac{f}{\beta} \) is much larger than \( \Delta x \) [see Williams and Zienkiewicz (1981)]. The Galerkin method is applied to this linearized equation set using a piecewise linear test function for V-2 and V-3, and a piecewise constant test function for VI-1.

The piecewise constant basis function has the property of displacing the geopotential to the centroid location of the elements, which should give the same effect as staggering the grid points, as seen in Figure 13. The density of geopotential data in the domain is now greater.
Figure 13. Staggering of the geopotential about the velocity. The $\alpha$ symbol are the actual grid point locations and the $\bullet$ symbol the centroid location of each element.
than the density of the velocity data. Instead of a geopotential value for each node, there is one averaged value over each element.

The final form of the Galerkin equation for VI-1, V-2 and V-3 after performing the time differencing is:

\[ u^{n+1}_{j} \phi_{j} = u^{n}_{j} \phi_{j} - \alpha \left[ \phi^{n}_{j} \phi_{j} + u^{n}_{j} u_{k}^{n} \phi_{j} \phi_{k} \phi_{1} \right] \]

\[ u^{n+1}_{j} \psi_{j} = u^{n}_{j} \psi_{j} - \alpha \left[ \phi^{n}_{j} \phi_{j} + u^{n}_{j} u_{k}^{n} \phi_{j} \phi_{k} \phi_{1} \right] \]

\[ v^{n+1}_{j} \phi_{j} = v^{n}_{j} \phi_{j} - \alpha \left[ \phi^{n}_{j} \phi_{j} + v^{n}_{j} v_{k}^{n} \phi_{j} \phi_{k} \phi_{1} \right] \]

This linearized set of equations VI-2, VI-3 and VI-4 is solved using a Gauss Seidel iterative procedure. It is worth mentioning that the coefficient matrix \( \langle \phi_{j}, \phi_{1} \rangle \) in equation VI-2 has all non zero coefficients equal to one, since the integration of the inner product \( \langle \phi_{j}, \phi_{1} \rangle \) involves piecewise constant functions.

This equation set is implemented rather than the equation set V-1, V-2 and V-3 using all of the existing primitive model code. The major modification involved the way that the geopotential was referenced. With an average geopotential over each element instead of a value at each node for a 12x12 mesh, there are 288 geopotential points versus the 156 velocity (u,v) points.
E. RESULTS

The results from this linearized model are compared to those from the primitive model. The initial field for this experiment, Figure 14 has a maximum perturbation zonal wind that is one fifth of the value used in Chapter V, Figure 10. The mean zonal wind remains 10 meters/second. The 48 hour forecast solutions for each field are compared in Figure 15.

This alternative formulation shows some promise, although there are some minor perturbations in these contours compared to those in the primitive solution. No explanation is offered for this small-scale noise, although possibly the other formulations presented by Williams and Zienkiewicz (1981) would improve the solutions.
Figure 14. Initial fields for both the primitive model and the linearized model.
Figure 15. 48 hour comparison between the primitive model and the Linearized model. (APA = 1.1 m/s)
VII. VORTICITY-DIVERGENCE MODEL EXPERIMENT

As in the previous two chapters, a comparison between two models will be presented. The results from the vorticity-divergence model are compared to the results from the primitive model. To fully exploit the differences between the models performances and indicate the strengths and weaknesses of each formulation, three domain geometries and three initial conditions are used. All solutions are at 48 hour, except for one case which was extended to 96 hours.

From these two finite element formulations some additional insight is obtained concerning the execution time required as the grid resolution changes. Lastly, a brief discussion on the sensitivity of the computational technique is given.

A. TEST DOMAINS AND INITIAL CONDITIONS

The three domain geometries used in the model evaluation are illustrated in Figure 16. All domains consist of a 12x12 element mesh with equilateral shaped elements (156 grid points) and cyclic continuity is imposed on the east and west boundaries. The domain has dimensions of 4245 KM along the x axis and 3563 KM along the y axis.

The regular domain (Figure 16a) has a uniform distribution of grid points, with a minimum grid spacing...
Figure 16. Test domain geometries.
along the x axis of 337 KM. This is the most congenial domain and produces the best results.

The smooth domain (Figure 16b) has a smoothly varying distribution of grid points. This technique allows a smooth variation of resolution. It was developed by Elder (1961), who showed how it significantly reduced noise at all frequencies compared with other variable grid domains.

The degree of resolution variation is accomplished by selecting an appropriate value for the ratio of maximum stretch to minimum shrink along both axes. The experiments presented in this chapter use a 2.5 ratio for both directions. This produces a grid point concentration in the right center of Figure 16b and coarse resolution at the top and bottom left of the domain. The minimum grid spacing along the x axis is 199 KM in the dense grid point area.

The third domain was the least hospitable geometry for both models. The abrupt domain (Figure 16c) has a dense grid point concentration on the left and coarse spacing on the right of the domain. The minimum grid spacing along the x axis is 168 KM in the area on the left. The grid spacings are uniform except for the abrupt change along the center.

Although these three domains are simple in structure, they are adequate to test both models' performance. To further enhance some differentiating characteristics between the two formulations, three initial conditions are used. Two have previously been described in Chapters V and VI. Their
main distinguishing feature is the arbitrary perturbation amplitude (APA) magnitude. The nearly linear case has an APA = 1.1 m/s compared to 5.5 m/s for the more nonlinear case. All initial conditions have a 10 m/s mean flow component.

With the nearly linear case both models behave well. The introduction of the more nonlinear initial disturbance illustrated the boundary and computational technique sensitivity. The third initial condition is the nearly linear initial field with the wave length equal to half the domain length. This has the effect of producing two waves with the domain.

3. TEST CASE COMPARISONS

The comparisons between models are divided according to the domain geometry. The primitive model has three fields: geopotential, u, and v. The vorticity–divergence model has seven fields: geopotential, stream function, velocity potential, u, v, vorticity and divergence. The geopotential, u and v will be the only fields used for the comparison.

1. Regular Case

The regular domain geometry (see Figure 16a) is used for this first comparison. The initial conditions have an APA = 1.1 m/s, as shown in the contour plots in Figure 17.

The 48 hour solutions for both models are presented in Figure 18. As anticipated, all of the solutions have
Figure 17. Initial fields for both the primitive and vorticity-
divergence models using the regular domain and perturbation amplitude
of 1.1 m/s.
Figure 18. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the regular domain. (APA = 1.1 m/s).
smoothly contoured fields with no noticeable noise and their phase velocities are comparable.

With the nearly identical solutions, the distinguishing feature between the models is the computational time. The computational time is determined by the size of the time step each model is allowed to use as indicated on equation III-26. For this domain the grid spacing is 337 km. The maximum phase velocity possible is different for each model. The primitive model allows gravity waves so \( c = 320 \text{ m/s} \) whereas the vorticity-divergence model filters out these high frequency waves. A \( c = 10 \text{ m/s} \) is used for this formulation.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \Delta x ) (km)</th>
<th>( c ) (m/s)</th>
<th>( \Delta t ) (sec)</th>
<th># of steps</th>
<th>CPU units (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE</td>
<td>337</td>
<td>320</td>
<td>458</td>
<td>376</td>
<td>148.5</td>
</tr>
<tr>
<td>V-D</td>
<td>337</td>
<td>10</td>
<td>13.755</td>
<td>13</td>
<td>33.3</td>
</tr>
</tbody>
</table>

Comparisons of the computational times for a 48 hour forecast between the primitive model and the vorticity-divergence model.

Table 1 compares the results of the computational times for both models. The vorticity divergence model produced as accurate results over the regular domain using 22% of the computational time needed by the primitive model. In fact from geostrophic reasoning the vorticity-divergence model should produce better forecasts when small scale features are present.
2. Smooth Case

The smooth domain geometry (Figure 16b) is used in this second comparison. The initial fields shown in Figure 15 have a disturbance amplitude of 1.1 m/s.

The 48 hour solution comparison is presented in Figure 20. All fields again have smooth contoured plots. Close inspection of the two u fields, Figures 20c and d, shows that the primitive u field has small kinks along some contours and a weak tilt near the central boundary nodes, although this may be a function of the plotting routine. Notice the good symmetry for the vorticity-divergence u field.

As in the regular case, this smooth experiment produced two acceptable solutions. Again, the computational time is the differentiating criteria.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\Delta x$ (KM)</th>
<th>$c$ (m/s)</th>
<th>$\Delta t$ (sec)</th>
<th># of steps (iterations)</th>
<th>CPU units (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>199</td>
<td>300</td>
<td>271</td>
<td>638</td>
<td>324.9</td>
</tr>
<tr>
<td>V-D</td>
<td>199</td>
<td>10</td>
<td>8124</td>
<td>21</td>
<td>49.5</td>
</tr>
</tbody>
</table>

Comparison of the computational times for a 48 hour forecast between the primitive model and the vorticity-divergence model using the smooth domain.

Table 2 compares the computational times for both models. The vorticity-divergence has a 83.8% saving of CPU time.
Figure 19. Initial fields for both the primitive and vorticity-divergence models using the smooth domain and perturbation amplitude of 1.1 m/s.
Figure 20. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. (APA = 1.1 m/s)
In an effort to contrast the computational accuracy of the models, this experiment is repeated using the more nonlinear initial case. Figure 21 shows the initial fields with $\text{APA} = 5.5$ m/s. This larger initial disturbance is reflected in the greater geopotential amplitude and magnitudes of the contour lines.

Figure 22 shows the 48 hour solution comparison. As in the more linear case presented above, all of the plots have smooth contours with no noticeable noise. The vorticity-divergence geopotential field, Figure 22b, has the ridge extending farther north, and flattening of the southernmost contour, than does the primitive geopotential field, Figure 22a. The mean geopotential heights for both models have also increased. The primitive mean geopotential is now 49680 gpm and the vorticity-divergence geopotential is 49620 gpm.

These two discrepancies indicate that the boundaries are not handled accurately. During the implementation of the vorticity-divergence model, treatment of the boundaries was the most troublesome phase. The vorticity-divergence formulation is a complex equation set and time limitations restricted further investigation of more sophisticated boundary conditions.

This same initial condition is now extended to a 96 hour forecast, which is shown in Figure 23. The mean vorticity-divergence geopotential increased to 49600 gpm.
Figure 21. Initial fields for both the primitive and the vorticity-divergence models using the smooth domain and perturbation amplitude of 5.5 m/s.
Figure 22. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. (APF = 5.5 m/s)
Figure 23. 96 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. (APA = 5.5 m/s)
whereas the mean primitive geopotential remained constant. All fields have smooth contours. Close inspection of both u fields, Figure 23c and d, shows a slight skewing of the contours along the central channel grid points. The vorticity divergence u field has the most pronounced deviations. A hypothesis for this skewing, explained further below, is that it is caused by the computational technique employed. The relaxation schemes are extremely sensitive and fine tuning of the relaxation coefficient would have required more time than was available.

Table 3

<table>
<thead>
<tr>
<th>Model</th>
<th>( \Delta x ) (km)</th>
<th>( C ) (m/s)</th>
<th>( \Delta t ) (sec)</th>
<th># of steps</th>
<th>CPU units (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>199</td>
<td>300</td>
<td>271</td>
<td>1276</td>
<td>594.0</td>
</tr>
<tr>
<td>V-D</td>
<td>199</td>
<td>10</td>
<td>8124</td>
<td>42</td>
<td>91.0</td>
</tr>
</tbody>
</table>

Comparison of the computational times for a 96 hour forecast between the primitive model and the vorticity-divergence model using the smooth domain.

Table 3 shows the comparison between both models for the 96 hour forecast. A savings of 85 percent is realized with the vorticity divergence model.

The above experiment points out the two areas where the vorticity—divergence model is presently weak, the increase of the geopotential and the sensitivity of the relaxation coefficients. Both these weaknesses can be improved and are not a result of the formulation but of the implementation. At present, their influence is not detected.
except in extremely long forecasts, such as in the 96 hour example. Further experiments may still be required if they demonstrate significant differences in the solutions of the two models.

The last experiment on the smooth domain uses the more linear initial condition $\alpha = 1.1 \sqrt{s}$, but its wave length is divided by two, so that two shorter waves are propagated through the channel. The initial conditions are shown in Figure 24. Decreasing the wave length has the same effect as decreasing the density of grid points. In this case six grid points are used to describe the wave structure versus the 12 used in the previous cases.

The 46 hour comparisons are shown in Figure 25. As in all previous cases, a computational time saving of 84% is gained with the vorticity-divergence model. With fewer grid points describing the wave structure, more small-scale noise is introduced into the solution. Comparing the primitive geopotential field, Figure 25a, to the initial geopotential, Figure 24a, shows a dampening of the wave amplitude, whereas the vorticity-divergence geopotential, Figure 25b, correlated well with the initial geopotential.

The high frequency noise is evident on both $u$ fields, Figures 25c and d. The primitive model $u$ field is poorly defined along the boundary and becomes irregular over the interior grid points.
Figure 24. Initial fields for both the primitive and the vorticity-
divergence models using the smooth domain. There are two waves embedded
in the flow and APA = 1.1 m/s.
Figure 25. 48 hour forecast comparison between the primitive model and the vorticity-divergence model using the smooth domain. Two waves are embedded in the flow and APA = 1.1 m/s.
The smooth domain allows a variable resolution of grid points and produces excellent results. As the wave length gets shorter, or the forecast length gets larger, some small-scale noise is apparent, especially with the primitive formulation.

3. Abrupt Case

The abrupt case comparison uses the abrupt domain geometry (see Figure 16c). This grid point configuration is used to further illustrate the effects of spatial resolution. The previous case using the smooth domain and half wave length introduced noise into the solution, but the spatial resolution changed slowly and gradually.

Consider the transition necessary in an operational model, where the luxury of having a long smooth transition into the region of high resolution may not be possible. The abrupt domain is an example of the results obtained when spatial resolution is decreased rapidly.

The initial fields are shown in Figure 26. The more linear case, $\text{APA} = 1.1 \text{ m/s}$, is used to eliminate effects due to the initial field, so that only the effects due to the grid geometry are seen.

The 48 hour comparisons are shown in Figure 27. Both solutions are affected by this geometry, but the primitive solutions are totally disorganized and unacceptable.

Table 4 shows the comparison of computational times for both models for a 48 hour forecast. An 85% savings in
Figure 26. Initial fields for both the primitive and vorticity- divergence models using the abrupt domain and perturbation amplitude of 1.1 m/s. Note that the element shapes are not equilateral triangles.
Figure 27. 48 hour warm air comparison between the primitive model and the vorticity-divergence model using the abrupt domain. (APA = 1.1 m/s).

104
CPU time is obtained using the vorticity-divergence model. Not only is there a computational savings with the vorticity-divergence model, but the solution is significantly better than the primitive model.

Table 4

<table>
<thead>
<tr>
<th>Model</th>
<th>$\Delta x$</th>
<th>c (m/s)</th>
<th>$\Delta t$ (sec)</th>
<th># of steps (iterations)</th>
<th>CPU units (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE</td>
<td>168</td>
<td>300</td>
<td>228</td>
<td>756</td>
<td>368.9</td>
</tr>
<tr>
<td>V-L</td>
<td>168</td>
<td>10</td>
<td>6858</td>
<td>25</td>
<td>55.7</td>
</tr>
</tbody>
</table>

Comparison of the computational times for a 48 hour forecast between the primitive model and the vorticity-divergence model using the abrupt domain.

C. COMPUTATIONAL SENSITIVITY

This section will offer an explanation for the skewing of the contours in the 96 hour forecast solution over the smooth domain (Figure 23). As mentioned previously, there was not enough time available for fine tuning the overrelaxation coefficient, which is used while solving the system of equations. The overrelaxation coefficient may be very sensitive and small changes can, on occasion, radically change the rate of convergence. The optimal value of the overrelaxation coefficient depends on the specific form of the coefficients of the equation and the error distribution.

The equation set to be solved consists of three equations and each equation required its own relaxation coefficient. When solving the equations over the regular domain, the entire system is well behaved and an optimum
relaxation coefficient is easily determined. However, as the domain geometry changes, the system of equations do not converge as rapidly and the relaxation coefficient requires further fine tuning.

The 96 hour forecast uses the smooth domain. The mid-latitude grid points are compacted, creates a denser belt in the middle of the channel. The coefficients originally computed using the regular domain need adjustments to properly solve the equations.

To illustrate the significance for fine tuning the relaxation coefficient, consider the series of plots in Figure 28. The vorticity divergence equation set can be simplified by assuming the flow is non-divergent, so that only the vorticity equation needs to be solved. Figure 28a is the 48 hour vorticity field using this equation over the regular domain with an overrelaxation coefficient of 1.3. The field is well defined with smooth contours.

Figure 28b is similar to the case in Figure 28a except that the smooth domain is used. Notice the V-shaped kink in the pattern with a steeper slope in the upper half. This increased bias in the upper half is caused by relaxing the field in the same direction during each pass over the domain. When the direction is reversed after each pass, the exaggerated bias in the upper half disappears, as is seen in Figure 28c. However, the V-shaped kink is not eliminated.
Figure 28. Computational sensitivity using the 48 hour vorticity fields. Fig. 28 A) the 48 hour forecast using the regular domain, 28 B) the same forecast using the smooth domain relaxing in one direction, 28 C) same as 28 B) except alternate the direction of relaxation after each pass over the domain, and 28 D) same as 28 C) except the relaxation coefficient was fine tuned from 1.3 to 1.297.
Varying the relaxation coefficient from 1.3 to 1.267 produces the much improved solution in Figure 28d. If the relaxation coefficients for the other two equations could also be fine tuned, it appears that improved solutions would result. There are also other relaxation techniques available that have potential for improving the solution while also converging at a faster rate. Some of these techniques will be tested in the future using this model.
VIII. CONCLUSIONS

This research investigated different finite element formulations for the shallow water equations. The two-dimensional domain was a channel which simulated a belt around the earth. Analytic initial conditions were used to simplify the comparisons. Two formulations were examined; one using different shaped basis functions and the other using a different form of the equation. Each was compared to the primitive form of the shallow water equations that was developed by Kelley (1975).

The use of equilateral shaped elements which was suggested by Mr. M.J.P. Cullen significantly improved the solutions compared to Kelley's model, which originally used right triangles as basis functions. Most of the other studies in this thesis used the equilateral triangles.

Williams and Zienkiewicz (1981) suggested the use of piecewise linear basis functions for the velocity field and piecewise constant functions for the height field. This formulation was tested with a linearized continuity equation. The results were poorer than those obtained with Kelley's model.

Most of the effort in this thesis was devoted to implementing and testing a vorticity-divergence model...
similar to the ones developed by Staniforth and Mitchell (1977) and Cullen and Hall (1979). Several tests were presented which compare this formulation with Kelley's model. It was found that this model executes approximately one order of magnitude faster than does the primitive formulation used by Kelley. Secondly, as the spatial resolution between grid points decreases, this formulation produces a solution that is far superior to the primitive form. A disadvantage is its computational sensitivity, which requires fine tuning in solving the elliptic equations for certain geometries. It also requires 25 percent more computer storage, due to the more complex equation set and the additional variables that are treated.

Implementation of finite element models is not easy. However, there is a lot of generality and redundancy imbedded in the computations. Versatile modules were written which significantly reduced the effort in implementing the vorticity-divergence model.

Further research is suggested using this finite element formulation. It has accurate phase propagation, is able to handle variable grid geometry, reduce the small-scale noise and decrease the model's execution time. Specifically more advanced methods of solving the elliptic equations should be investigated. Finally, the formulation should be tested with small-scale forcing, where its advantages should be most evident.
Mathematical and computational techniques are used to solve the problem of finding the absolute (potential) field of all magnetic, electric, or potential quantities. The vector is then calculated, and the motion field is computed.

The absolute field is calculated with the help of the motion field and the vector field components. The potential and electric field values are then calculated.
SET: THE FOLLOWING DATA FOR INPUT: 

\[
\begin{align*}
\begin{matrix}
\alpha & = \beta \\
\gamma & = \delta \\
\epsilon & = \zeta \\
\eta & = \theta \\
\iota & = \kappa \\
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{matrix}
\end{align*}
\]

\[
\begin{align*}
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\alpha & = \beta \\
\gamma & = \delta \\
\epsilon & = \zeta \\
\eta & = \theta \\
\iota & = \kappa \\
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\alpha & = \beta \\
\gamma & = \delta \\
\epsilon & = \zeta \\
\eta & = \theta \\
\iota & = \kappa \\
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\alpha & = \beta \\
\gamma & = \delta \\
\epsilon & = \zeta \\
\eta & = \theta \\
\iota & = \kappa \\
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\alpha & = \beta \\
\gamma & = \delta \\
\epsilon & = \zeta \\
\eta & = \theta \\
\iota & = \kappa \\
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]

\[
\begin{align*}
\lambda & = \mu \\
\nu & = \xi \\
\omicron & = \pi \\
\rho & = \sigma \\
\tau & = \upsilon \\
\phi & = \chi \\
\psi & = \omega \\
\end{align*}
\]
ACM COMPUTER LjV.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.

10.
INPUT THE CHANNEL

$M = M + 1$

$V = A * \sin(\alpha) * VEL$

$\alpha = 9.2 * TAN^{-1}(0.00001)

\theta1 = \theta1 + \frac{\pi}{180}

\theta2 = \theta2 + \frac{\pi}{180}

VERT1 = THE CHANNEL'S LOWER LATITUINAL BOUNDARY RADIUS FROM THE EARTH'S SURFACE

VERT2 = THE CHANNEL'S UPPER LATITUINAL BOUNDARY RADIUS FROM THE EARTH'S SURFACE

CONVERT FROM SPHERICAL TO ECLIPTIC HAP PROJECTIONS

$x = \sin(\theta1) / (1 - \sin(\theta1))$

$y = \cos(\theta1) / (1 + \sin(\theta1))$

$z = 1 + \frac{\pi}{180}$

$Y = \text{length of the channel in meters}$

$Y = Y - \text{value}$

$DELTA = \text{geometric space between north/south grid pts}$

$DELTA = \text{value}$

$N = \text{channel's mid latitude value in radians}$

$N = (\theta1 + \theta2) / 2$

$DELTA = \text{geometric space between east/west grid pts}$

$DELTA = DELTA / \cos(\theta1)$

$\text{if } \theta1 = \theta2$}

$X = 2 * \cos(\theta1)$

$Y = \text{value for the } \theta1 \text{ channel latitude}$

$Y = Y + \text{value}$
24

NOW TRANSFORM COMPUTER

RETURN

IF 

CALL OUTPUT(S)

CONTINUE

YV10 = 1.1 * x + y

YQ 25 YV = 1 * LAT

YQ = YLAT (YV * XQ + 1) - YLAT (YV * XQ)

YV1. = y

CONTINUE

FOR A, B, Y, TRANSFORMATION

COMPUTE XY = ST YQ

IF (XY < 10.5) XY41 = Y11

XY = (XY/10.5) * (5.4) + 0.3

LXY = 1 * FIX (FRACT)

CONTINUE

RETURN

END
EXECUTE PIC

12345

CALCULATED THE AREA UNDER THE POLYGON, THE TRANSFORMATION PERFORMED.

TRANSFORMATIONS FROM CARTESIAN TO AREA COORDINATES

COMPUTE THE A COEFFICIENTS (HORIZONTAL TRANSFORMATION)

COMPUTE THE B COEFFICIENTS (DUE TO CYCLIC COVARIANCE)

COMPUTE THE C COEFFICIENTS (VERTICAL TRANSFORMATION)

RESULT IS TIMES THE AREA UNDER THE POLYGON, T=}

RETURN
SUBROUTINE LNS

CALL SUBRoutines in (6) SUBROUTINE LNS

RETURN

END

TEST EXPONENTS STORED = 38 x 38 x 38 x 38.  
INITIAL DECILES FOR LI STAT-IX EFF THE EXAMPLES.

1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
1.2 x 9 x 2 = .9 = .9 = .9 = .9 = .9.
SUBROUTINE ASMCMD (CLMXK G)

CALL G OUTROUT A PACKED ONE DIRECTIONAL ARRAY F.

CLOSE (10,13)

FILL ARRAY I 0 AS A THERMAL MASS MATRIX.

LOCATE THE CONNECTIVITY LIST FOR THE NODES OF ELEMENT

MODE = 1

START = ISTART(MODE)

SELECT ONE OF PLANNED MDES, CALL IT MDES

MODE = 1

MDES = ELEM (MODE, MODE)

LOCATE MDES'S CONNECTIVITY, FIND MATCH WITH NODES

INDEX = 1

START = ISTART(INDEX)

MDES = NAE (INDEX)

INDEX = INDEX + 1 TO 10

A MATCH IS FOUND BETWEEN MDES AND MDES'S NODES?

IF NOT STORES 11 33 IS ENTERED TO THE MASS MATRIX.

END OUTROUT = 3 (MDES, MDES) + 0 (XNTM) 30 TO 30

CONTINUE

CONTINUE

END
CALL RVRHS (2,1)
CALL \( \text{(E00) \text{ CALL}} \)

\( \text{SIG} = 1.0 \)

IF \( \text{SIG} < 0.0 \) \( \text{SIG} = -1.0 \)

CALCULATE FOR EACH ELEMENT

\( \text{SIG} = \text{SIG} \) \( \text{SIG} = \text{SIG} \) \( \text{SIG} = \text{SIG} \)

COMPLETE DERIVATIVE FOR BOTH THE STREAM FUNCTION AND VELocity POTENTIAL \( V \) \( V \) \( V \)

\( \text{SIG} = \text{SIG} \) \( \text{SIG} = \text{SIG} \) \( \text{SIG} = \text{SIG} \)

ADD RESULTS TO WORK VECTOR \( W \) \( W \) \( W \)

\( \text{SIG} = \text{SIG} \) \( \text{SIG} = \text{SIG} \) \( \text{SIG} = \text{SIG} \)

CONTINUE

RETURN

END
\[ H = 1.3, M = 1.3 \]

\[ H = 1.3, M = 1.3 \]

\[ H = 1.3, M = 1.3 \]
...
J. ACM (1970) 17, 149-170

THE JACKSON NET

A generalized Jackson network with a finite number of
parallel queues. Each server is served by a single
channel that is controlled by a central computer system.

1. Introduction

The Jackson net is a network of queues with
multiple servers and multiple channels. Each server
is served by a single channel that is controlled by
a central computer system. The network is
organized into a set of queues, each with a
finite number of servers. The server at each queue
is served by a single channel that is controlled
by a central computer system.

2. The Model

The Jackson net is defined as follows:

- A set of queues, \( Q \), each with a finite
  number of servers, \( s_i \), and a finite
  number of channels, \( c_i \).
- A central computer system that
  controls the activity of all channels.
- A set of customers, \( C \), that
  arrive at the queues according to
  a Poisson process with rate \( \lambda \).
- Each channel has a finite
  number of customers, \( n_i \), that
  can be served simultaneously.
- The system is stable if the
  total number of customers in the
  system is finite and the
  arrival rate is less than the
  service rate.

3. The Solution

The solution to the Jackson net problem
involves the use of the Jackson network
algorithm. The algorithm is as follows:

- Initialize the system with an empty set of
  queues and customers.
- For each queue, determine the
  service rate, \( \mu_i \), and the
  arrival rate, \( \lambda_i \).
- For each channel, determine
  the number of customers, \( n_i \),
  that can be served simultaneously.
- For each customer that arrives,
  determine the queue that
  it should be assigned to.
- For each queue, determine
  the number of customers
  that are currently
  being served.
- Update the system state
  by removing customers
  from the queue and
  adding them to the
  service.
- Repeat until the system
  is stable.

4. Conclusion

The Jackson net is a powerful tool
for modeling and analyzing
complex systems. The algorithm
provides a way to efficiently
solve the problem of
optimizing
resource allocation in
such systems.
BEGIN

2.1. TOTL, THE ALGORITHM FOR THE ALPHABETIC EXAMPLE

2.2. THE ALGORITHM FOR THE ALPHABETIC EXAMPLE

2.3. THE ALGORITHM FOR THE ALPHABETIC EXAMPLE

2.4. THE ALGORITHM FOR THE ALPHABETIC EXAMPLE

2.5. THE ALGORITHM FOR THE ALPHABETIC EXAMPLE

END
DEFINE TOL15 = (0.1, 0.1, 0.1)

TOL = 1.0E-3

IF (TOL > TOL15) Go To 10

CALL TERM1 (1, 0.092, 1, 0.092, 0.001, 0.001, 0.001, 0.001)

TOL = 1.0E-3

CALL TERM2 (1, 0.092, 1, 0.092, 0.001, 0.001, 0.001, 0.001)

CALL TOL3 (1, 0.092, 1, 0.092, 0.001, 0.001, 0.001, 0.001)

CALL TOL4 (1, 0.092, 1, 0.092, 0.001, 0.001, 0.001, 0.001)

CALL TOL5 (1, 0.092, 1, 0.092, 0.001, 0.001, 0.001, 0.001)

END
JUL 05, 1974

**CALCULATII**

**VALVE, WHERE**

**[Mathematical notation and code]**

**JUL 05, 1974**
J. FINISH ST. PHASE

CALL INIT (J, I, C)

T = T + DT*2

CALL PRINT (AF, '.I', I, OF

CALL OUTPUT (7)

CALL OUTPUT (77)

STOP

RETURN

END
SUBROUTINE RELAX (N, IO, IPTR)

C THIS SUBROUTINE FOR RELAXATION OF ALL NODES SCOTT

C integer I, J, K, N

C real X(N), A(N,N), Y(N)

C real EPS, MAXIT, ALPHA

C real total, last, first

C integer iter, node

C C CHARACTERISTIC VALUES OF SCOTT'S RELAXATION SCOTT

C C C = 1, ..., N, NODES

C C IPTR(I) = FIRST(I) + 1, ... + LAST(I)

C C X(N) = Solution vector

C C A(N,N) = Stiffness matrix

C C Y(N) = Right hand side vector

C C EPS = Convergence criterion

C C START = Start of relaxation

C C STOP = Stop of relaxation

C C C = 0, START

C C C = 1, STOP

C C MAXIT = Maximum number of iterations

C C N = Number of nodes

C REAL EPS, MAXIT, ALPHA

C INTEGER I, J, K, N

C REAL A(N,N), X(N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE

C REAL X(N)

C REAL A(N,N)

C INTEGER IPTR(N)

C INTEGER START, STOP

C REAL EPS, MAXIT, ALPHA

C INTEGER NODE, ITER

C REAL RESOL, RESID

C INTEGER FIRST, LAST

C INTEGER NODE
SUBROUTINE PHI (L,TRIP, I )  

call PHI

/phi

ct

! psi

/psi

cn

! ch

/ch

rest
CALL GAMS (PSI, <CH1, 2>)
CALL SILEX (J; PSI, 1)

CALL = (S)

CALL THE RANDOM COEFFICIENT
SIGMA = 1.
CALL PH30JY (J, PSI, SIGMA)
CALL PH31JY (J, PSI, SIGMA)

CALL TES3 (J, <PHI(1:-1), J, J, J, VI>)
SIGMA = -1.
CALL TES5 (J, <PHI(3:-1), J, J, J, J, J, J, VI>)

CALL TES7 (J, <J:J, J:J, J:J, J:J>)
SIGMA = 0.
CALL TES9 (J, <J:J, J:J, J:J, J:J>)

CALL TES11 (J, <J:J, J:J, J:J, J:J>)
SIGMA = -1.

CALL TES13 (J, <J:J, J:J, J:J, J:J>)

CALL = (S)

CALL = (S)

CALL = (S)
SUBROUTINE RELAX (CODE, NODE)

Elapsed time is used for relaxation of the nodal potentials.

IF (ITER.EQ.1) THEN
    I = 0
ELSE
    I = 1
ENDIF

DO 10 CODE = 1, NCODE
    IF (I.EQ.1) THEN
        FIRST = 1
        LAST = NNODE
    ENDIF
    IF (I.EQ.2) THEN
        FIRST = NODE
        LAST = LAST + NODE - 1
    ENDIF
    IF (I.EQ.3) THEN
        FIRST = FIRST + NODE - 1
        LAST = LAST + NODE - 1
    ENDIF
10 CONTINUE

OVER RELAX TO SOLVER SOLUTION

CODE = IF (I.EQ.1) THEN
    IF (NDIM.EQ.1) THEN
        NODE = LAST
        TOTAL = 0
    ELSE
        NVODE = (NDIM + 1) / 2
        START = NVODE
        I = 0
        J = 1
        CODE = CODE + 1
    ENDIF
ENDIF
Determine the time levels

\#1 = 1
\#2 = 3
\#2 = 0 CHECK \#1 and \#2

CALL MAIN (TACT)

CALL CLAX (T, T)
CALL CLAX (T, T)
CALL CLAX (T, T)
CALL CLAX (T, T)

CALL DIVE (T, T)

CALL DIVER

TEST FOR OUTPUT CONDITIONS
IF \#1 = HOUR + T / 3600 THEN
IF \#1 = IT (1) THEN CALL OUTPUT (1)
IF \#1 = IT (2) THEN CALL OUTPUT (1)
CALL OUTPUT (1)
CALL OUTPUT (1)
CALL OUTPUT (1)

\#1 = \#1 / 10
\#1 = \#1 / 100
\#1 = \#1 / 1000

CALL IT (1)

CALL IT (1)
SUBROUTINE FIVR (NX, NY, NJVE, JVE) 

IF (CODE = 1, CODE) 
1) END 

IF (CODE = 1, CODE) 
3) END 

IF (CODE = 1, CODE) 
5) END 

IF (CODE = 4, CODE) 
7) END 

IF (CODE = 4, CODE) 
9) END 

END
I-T

1. I-T
2. I-T
3. I-T
4. I-T
5. I-T
6. I-T
7. I-T
8. I-T
9. I-T
10. I-T
11. I-T
12. I-T
13. I-T
14. I-T
15. I-T
16. I-T
17. I-T
18. I-T
19. I-T
20. I-T
21. I-T
22. I-T
23. I-T
24. I-T
25. I-T
26. I-T
27. I-T
28. I-T
29. I-T
30. I-T
31. I-T
32. I-T
33. I-T
34. I-T
35. I-T
36. I-T
37. I-T
38. I-T
39. I-T
40. I-T
41. I-T
42. I-T
43. I-T
44. I-T
45. I-T
46. I-T
47. I-T
48. I-T
49. I-T
50. I-T
51. I-T
52. I-T
53. I-T
54. I-T
55. I-T
56. I-T
57. I-T
58. I-T
59. I-T
60. I-T
61. I-T
62. I-T
63. I-T
64. I-T
65. I-T
66. I-T
67. I-T
68. I-T
69. I-T
70. I-T
71. I-T
72. I-T
73. I-T
74. I-T
75. I-T
76. I-T
77. I-T
78. I-T
79. I-T
80. I-T
81. I-T
82. I-T
83. I-T
84. I-T
85. I-T
86. I-T
87. I-T
88. I-T
89. I-T
90. I-T
91. I-T
92. I-T
93. I-T
94. I-T
95. I-T
96. I-T
97. I-T
98. I-T
99. I-T
100. I-T
CALL CALL (IXX), JXX, JXY, JRX, PLS, I, 1, 13, CL, IX
LC, IX, 1, TITLE, (AX, YAX, M, IV)
PLUT = I PLUT = I TEST(I INDEX)
PRINT(145, L72) L72
CONTINUE

RETURN
END
SUBROUTINE PLTM3 (MODE,MAXC,LAT,LON,LC,LAT,LON)

CALL XECM3 (MODE,MAXC,LAT,LON,LC,LAT,LON)

LAT = LAT + 1

TO (40,30)

MODE = MODE + 1

IF (MODEL = 1) MODEL = 2

IF (T(NODE) = MAX(NODE) = MAX(NODE)) NEXT = NEXT + 1

CONTINUE

TO (1000..21.1) GO TO 25

MODE = MODE + 1

MODE = MODEL + 1

NEXT = NEXT + 1

CONTINUE

TO (46)

STORE MULTI-DIMENSIONAL ARRAY

NEXT = NEXT + 1

CONTINUE

TO (75)

CONTINUE

END
SUB TIME - VAL

100:  "TIME" - SETS,  "TIME" - ANALYSIS,  "TIME" - CANCELS  "TIME" - EXECUTIVE  
   "TIME" - RESERVES  "TIME" - GRANTS  "TIME" - GENRATES  "TIME" - SETS  "TIME" - 
   "TIME" - EXECUTIVE

200:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

300:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

400:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

500:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

600:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

700:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

800:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

900:  "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
   "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

1000: "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
    "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

1100: "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
    "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

1200: "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
    "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE

1300: "TIME" - SETS  "TIME" - EXECUTIVE  "TIME" - RESERVES  "TIME" - GRANTS  
    "TIME" - GENRATES  "TIME" - SETS  "TIME" - EXECUTIVE
SUBROUTINE INTERP (XX,YY,JJ,XXL,YYL,XXM,YYM,XXH,YYH)

C Intermediate between resolution fields.

XXL = XX(JJ+1)
YYL = YY(JJ+1)
XXM = XX(JJ)
YYM = YY(JJ)
XXH = XX(JJ-1)
YYH = YY(JJ-1)

IF (XX(JJ) .NE. XX(JJ-1)) THEN
   T = Abs(XX(JJ) - XX(JJ-1))
   T2 = T*T
   T3 = T*T2
   T4 = T3*T
   T5 = T4*T
   XXN = XX(JJ-1) + T2
   YYN = YY(JJ-1) + T3
   T6 = (T4 - T5) / (T4 + T5)
   XXM = XX(JJ) * T6 + XX(JJ-1) * (1 - T6)
   YYM = YY(JJ) * T6 + YY(JJ-1) * (1 - T6)
   C = 0.5 * (YYN - YYM) / T4
   KK = 0
   IF (XXM .GE. XX(JJ)) THEN
      L = 2
      KK = 0
   ELSE
      L = 1
      KK = 1
   END IF
   IF (YYL .GE. YY(JJ)) THEN
      M = 2
      KK = 0
   ELSE
      M = 1
      KK = 1
   END IF
   X1 = XXM - XX(JJ) + XX(JJ+1)
   X2 = XX(JJ+1) - XXM
   Y1 = YYM - YY(JJ) + YY(JJ+1)
   Y2 = YY(JJ+1) - YYM
   IF (L .eq. 0) THEN
      IF (K .eq. 0) THEN
         X = XX(JJ)
         Y = YY(JJ)
      ELSE
         X = XX(JJ+1)
         Y = YY(JJ+1)
      END IF
   ELSE
      IF (M .eq. 0) THEN
         X = XX(JJ)
         Y = YY(JJ)
      ELSE
         X = XX(JJ+1)
         Y = YY(JJ+1)
      END IF
   END IF
   Print, 10, XXM, YYM, XXN, YYN, X1, X2, Y1, Y2
END IF

10 Format(3F14.6), I=1,2,4,6
LIST OF REFERENCES


<table>
<thead>
<tr>
<th>No.</th>
<th>Name and Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Defense Technical Information Center&lt;br&gt;Alexandria Station&lt;br&gt;Alexandria, Virginia 22314</td>
</tr>
<tr>
<td>2</td>
<td>Library, Code 2142&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93942</td>
</tr>
<tr>
<td>3</td>
<td>Dr. R.T. Williams, Code 634W&lt;br&gt;Department of Meteorology&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93942</td>
</tr>
<tr>
<td>4</td>
<td>Commander, Air Weather Service&lt;br&gt;Military Airlift Command&lt;br&gt;United States Air Force&lt;br&gt;Scott Air Force Base, Illinois 62225</td>
</tr>
<tr>
<td>5</td>
<td>Captain Edward T. Woodward&lt;br&gt;4414 Felene Dr.&lt;br&gt;Bellevue, Nebraska 68022</td>
</tr>
<tr>
<td>6</td>
<td>Captain Brian Van Orman&lt;br&gt;AFIT/CFPF&lt;br&gt;Kirtland-Patterson AFB, Ohio 45433</td>
</tr>
<tr>
<td>7</td>
<td>Mr. David A. Archer&lt;br&gt;Douglas Trubon Ranchford, Inc.&lt;br&gt;6150 Chevy Chase&lt;br&gt;Houston, Texas 77027</td>
</tr>
<tr>
<td>8</td>
<td>Prof. Gordon Bradley, Code 523Z&lt;br&gt;Department of Computer Science&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93942</td>
</tr>
<tr>
<td>9</td>
<td>Lt. P.J. Burga, Code 35&lt;br&gt;Department of Meteorology&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93942</td>
</tr>
<tr>
<td>10</td>
<td>Prof. Gilles Cantin, Code 6261&lt;br&gt;Department of Mechanical Engineering&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93942</td>
</tr>
</tbody>
</table>
11. Dr. W.J.P. Cullen  
Meteorological Office  
Fleetmills, Berk, United Kingdom

17. Dr. P.L. Isberry, Code 63Es  
Department of Meteorology  
Naval Postgraduate School  
Monterey, California 93943

13. Dr. J.A. Gelt  
NOAA Pac War Envir Lab  
University of Washington  
Seattle, Washington 98125

14. Dr. R.L. Haney, Code 63Hy  
Department of Meteorology  
Naval Postgraduate School  
Monterey, California 93943

15. LCDR Donald E. Rinsman, Code 35  
Department of Meteorology  
Naval Postgraduate School  
Monterey, California 93943

16. Prof. Uno R. Kordes, Code 52H  
Department of Computer Science  
Naval Postgraduate School  
Monterey, California 93943

17. Dr. Robert L. Lee  
Atmospheric and Geophysical Science Division  
University of California  
P.O. Box 808  
Livermore, California 94552

18. Prof. A.K. MacPherson  
Lehigh University  
Department of Mechanical Engineering  
Bethlehem, Pa 18015

19. Prof. E.N. Mooers, Code 68Mr  
Chairman, Department of Oceanography  
Naval Postgraduate School  
Monterey, California 93943

27. Prof. R. Newton, Code 69Ne  
Department of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943
21. Dr. M.A. Phillips
National Meteorological Center/NOAA
World Weather Building
Washington, D.C. 20233

22. Dr. T. Fosmond
Naval Environmental Prediction Research Facility
Monterey, California 93940

23. Prof. E. Salinas, Code 692
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, California 93943

24. Dr. Y. Sasaki
Department of Meteorology
University of Oklahoma
Norman, Oklahoma 73069

25. Prof. A.L. Schoenstadt, Code 53Zh
Department of Mathematics
Naval Postgraduate School
Monterey, California 93943

26. Dr. Andrew Staniforth
Research et Prevision Numerique
West Isle Office Tower, 5 ieme etage
2191 route Trans-Canada
Porvill, Quebec F9P1J3, Canada

27. Dr. W.C. Thacker
National Oceanic and Atmospheric Administration
Fickenbacker Causeway
Miami, Florida 33149

28. Dr. A. Keinstein
Naval Environmental Prediction Research Facility
Monterey, California 93940

29. Dr. H.J. Winninghoff, Code 75
Department of Meteorology
Naval Postgraduate School
Monterey, California 93943

30. Prof. J.C. Zienkiewicz
Head of Civil Engineering Department
Applied Science Building
Singleton Park
Swansea SA2 8PP
United Kingdom