DM/P CRITICAL AREA DETERMINATION AND ITS IMPLEMENTATION ON MESSAGE-PASSING PROCESSOR,

A Thesis Presented to
The Faculty of The College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
JAIKISHAN RAJENDRAN

November, 1992
This thesis has been approved
for the Department of Electrical and Computer Engineering
and the College of Engineering and Technology

Associate Professor of Electrical and Computer Engineering
ACKNOWLEDGEMENTS

I express my utmost gratitude to Dr. Mehmet Celenk, Associate Professor of Electrical and Computer Engineering, for all his encouragement and support during this project. I also thank Mr. Michael DiBenedetto, Program Engineer, Avionics Engineering Center for his guidance throughout the project. Avionics Engineering Center is thanked for its support of the project and also for providing various resources for learning. The experience gained at the center will be very helpful for the future. I would also like to thank Dr. Robert Lilley, Director, Avionics Engineering Center, Dr. Roger Radcliff, Assistant chairman, Department of Electrical and Computer Engineering, and Dr. M. S. K. Sastry, Department of Mathematics, for their assistance and contributions in reviewing this work and serving as members of the Thesis committee. I sincerely thank Michael Braasch for his assistance throughout the project.

I thank my colleagues Soo, Stephan, Chuck, Keith, Mohamed, Matt Gordon, and Matt Weaver for all their assistance. I shall not forget the words of encouragement from Louise. Thank you. My thanks also to my friend Sam for his constant support - 'Thanks! Thanks! Thanks!'

I thank my parents and brothers for their constant love and encouragement. I am greatly indebted to Rama for showing great patience and support all along. Finally, I thank the Almighty for without him, nothing is possible.
TABLE OF CONTENTS

CHAPTER I
INTRODUCTION 1

CHAPTER II
BACKGROUND 16

CHAPTER III
PROBLEM STATEMENT 24

CHAPTER IV
APPROACH TO SOLUTION 28
A. Determination of Grid Size 28
B. Determination of Optimal Grid Spacing 39
   B.1  Theoretical Formulation 39
   B.1.1 Determination of Effective Bandwidth 40
   B.1.2 Discrete Approximation 56
   B.1.3 Verification of Optimal Grid Spacing 78

CHAPTER V
COMPUTER SIMULATION 85
A. MLS Simulation Package Structure 85
   A.1 Transmitter or Propagation Module 85
   A.2 Receiver Module 91
B. Contour Generation 93
CHAPTER VI
PARALLEL IMPLEMENTATION OF DME/P CONTOUR GENERATION

A. Symult S-2010 Message Passing Multicomputer System 101
B. Parallel Decomposition of DME/P Contour Generation 102
C. DME/P Contour Generation on S-2010 110
D. Determination of Optimal Grid Spacing 113
E. Verification of Optimal Grid Spacing 117
F. Performance Analysis 127

CHAPTER VII
SIMULATION RESULTS 149
A. Determination of Grid Size 149
B. Determination of Optimal Grid Spacing 150
C. Parallel Implementation of PFE Contour Analysis 156

CHAPTER VIII
CONCLUSIONS AND RECOMMENDATIONS 159

REFERENCES 163

APPENDIX A. The Overall Simulation Software 170
APPENDIX B. Simulation Results 261
APPENDIX C. Parallel Implementation Software 348
CHAPTER I

INTRODUCTION

The Microwave Landing System (MLS) is an all weather precision approach and landing system which provides three dimensional positional information to an approaching aircraft [1-6]. Figure 1.1 illustrates the MLS approach and landing operation. The International Civil Aviation Organization (ICAO) recognized the advantages of MLS and selected it to replace the currently used Instrument Landing System (ILS) [3]. Currently, MLS provides two hundred channels, with the availability of additional channels which enhances its capability to handle higher traffic volume in the future. The MLS ground system includes the following subsystems: Azimuth (AZ), Elevation (EL), Precision Distance Measuring Equipment (DME/P), Back Azimuth (BAZ) and basic and auxiliary data. The signal from AZ, BAZ, EL subsystems and data words are transmitted at C-Band (≈ 5 GHz) using a time-multiplexed format (Figure 1.2) while the DME/P signal is transmitted using a paired frequency in the L-Band (≈ 1 GHz). Figures 1.4, 1.6 and 1.9 illustrate the coverage provided by the various subsystems of MLS [19].

The azimuth subsystem [1,4-6,19-21] (Figure 1.3) provides the lateral position of an approaching aircraft relative to the antenna and is typically located behind the stop end of the runway on the extended centerline. The azimuth system operates in
Figure 1.2  Time multiplexed MLS signal format.
Figure 1.3 1° beamwidth azimuth equipment.
the frequency range of 5.031 to 5.0907 GHz and uses large antennas (=50-60λ) to generate a narrow vertical fan shaped beam (Figure 1.4). The antennas, however, are physically small due to the high operating frequencies. The azimuth beam scans the coverage area which can extend to about +62 degrees to -62 degrees (typically +40° to -40°) about centerline, in a "TO-FRO" pattern.

The elevation subsystem [1,4-6,19-21] (Figure 1.5) provides vertical guidance to the approaching aircraft and is typically sited at a distance of 800-900 feet from runway threshold with an offset of 250-400 feet from runway centerline. The elevation subsystem also uses a physically small but electrically large antenna to generate a narrow horizontal fan shaped beam. This beam scans the coverage area vertically from 0.9 degrees to at least 15 degrees above horizontal (Figure 1.6).

The AZ and EL subsystems are commonly designated as the angle portion of the MLS since they provide angular guidance information. The angular position, either elevation or azimuth, is determined by measuring the time difference between the TO- and FRO-pulses received in the aircraft. The angle measurement concept is illustrated in Figure 1.7.

The DME/P subsystem [7,8,11] (Figure 1.8) which may be collocated with the azimuth subsystem or located up to a distance of 900 feet away from the AZ site
Figure 1.4  Approach azimuth coverage.
MINIMUM HORIZONTAL COVERAGE SECTOR IS EQUAL TO THE APPROACH AZIMUTH PROPORTIONAL GUIDANCE REGION

Figure 1.6 Approach elevation coverage.
Figure 1.7 Azimuth angle measurement concept.
Figure 1.8  DME/P equipment.
(offset siting) provides line-of-sight range information to the airborne aircraft. The area of coverage extends out to about 22 miles from the antenna location (Figure 1.9). Through these subsystems MLS provides the high accuracy needed to achieve Category III (CAT-III) operations. CAT-III landings require high accuracy since the decision regarding landing may be made without virtually sighting the runway. Though MLS provides such high accuracy its performance is degraded by multipath arising from obstacles close to the antenna subsystems.

MLS signals are affected by multipath in the form of reflections, scattering and diffraction from obstacles (e.g., aircraft, hangars, light poles, etc) in the airport environment (Figure 1.10). The presence of these obstacles close to the antenna may cause [12-14] guidance errors. To ensure precise approach guidance, the errors have to be kept to a minimum in order to meet specified standards. To reduce guidance errors critical areas are identified around the transmitting antenna subsystems.

Critical area is defined as the area in which the presence of a scatterer aircraft will cause degradation of Microwave Landing System (MLS) signals. This degradation results in guidance error for an approaching aircraft. To reduce the guidance error critical areas are made clear of any obstacles. Critical area studies have been the subject of intensive investigation for azimuth and elevation subsystems [9,10]. Figure 1.11 illustrates an azimuth critical area for an offset azimuth case. However, the critical area criteria for the Precision Distance Measuring Equipment
Figure 1.10 Multipath sources.
Illustration of Azimuth critical area.

Figure 1.11
(DME/P) subsystem of the MLS have not been studied. It is assumed that the azimuth critical areas will provide adequate protection for DME/P. This may not necessarily be the case. Further, Federal Aviation Administration (FAA) specifications allow the DME/P antenna to be sited as much as 900 feet away from the azimuth antenna. Such offset siting is quite likely to place the antenna in close proximity to airport obstacles. Hence, a study of DME/P critical area is warranted. This work illustrates the determination of simulation parameters to be used for DME/P critical area study.

The determination of simulation parameters for the DME/P critical area requires extensive computer simulation. It is desired that the computational time for this process be reduced. Parallel processing offers a solution to this problem by exploiting the intrinsic parallelism in the problem domain. Hence, a parallel implementation of the above task is proposed in the 16-node message-passing multicomputer system (Symult S-2010) [34,35].

In chapter II a brief description of the DME/P subsystem is given. Chapter III gives an introduction to the determination of simulation parameters. Chapter IV describes the methodology involved in determining these parameters. Chapter V illustrates various features of MLS Math Model and its use in this critical area study. Chapter VI describes the parallel implementation of critical area determination. Chapter VII and VIII present the results and conclusions of this study.
CHAPTER II

BACKGROUND

DME/P is an important component of MLS for computed centerline (C.C.L) and area navigation (RNAV) approaches and also ensures safe landing and roll out. DME/P determines the slant range between the aircraft and the ground transponder by measuring the elapsed time between the dispatch of the interrogator pulse from the aircraft and the receipt of a reply pulse from the ground transponder [11]. Figure 2.1 illustrates the principle of range measurement. The slant range $R$ is given by

$$R = \frac{(t-\tau)c}{2} \quad (2.1)$$

where $t$ is the elapsed time between $t_1$, at which the reply is received and the time $t_0$, at which the signal is transmitted, $c$ is the propagation velocity of the signal and $\tau$ is the fixed time delay introduced by the transponder. Although the above equation illustrates the principle of range measurement, the actual measurement is much more complex in the presence of multipath.

A DME/P signal is degraded by reflections, scattering and diffraction from obstacles in the airport environment as in the case of azimuth and elevation signals.
Figure 2.1 DME/P range measurement principle.
Multipath from obstacles causes error in elapsed time measurement, which affects the predicted range value. The difference between the DME/P indicated range and the actual range of the approaching aircraft gives the guidance error. The guidance error has two components (Figure 2.2): the low frequency component known as Path Following Error (PFE), shown in Figure 2.3, can displace the aircraft from its intended course and the high frequency component, Control Motion Noise (CMN), may affect aircraft attitude and pilot confidence factor [6-8]. To keep this effect within established tolerances, critical areas are identified.

Determination of DME/P critical areas is essential to prevent multipath effects. This involves obtaining the guidance error as a function of scatterer location and orientation in the form of error contour plots. The time and expense involved rule out the use of flight tests to obtain error contours. Use of mathematical tools is desired to generate these contours. The available mathematical tool to predict these errors is the MLS mathematical model [15-17].

The MLS mathematical model was used to generate error contours in previous critical area work performed at Ohio University, for azimuth and elevation subsystems [10]. Error contours were generated for straight-in and advanced procedures to determine protection regions for specific combinations of antenna siting and ground aircraft types. Figure 2.4 shows one such contour plot.
Figure 2.3  Effect of PFE on aircraft guidance.
Figure 2.4 Sample PFE contour for the azimuth subsystem.
DME/P critical area determination also uses the MLS mathematical model to generate error contours. The MLS package includes transmitter and receiver modules for azimuth and elevation and the transmitter portion for DME/P. The transmitter module, also called the propagation model, simulates the propagation of the signal in the airport environment. It uses ray theory to determine the multipath components arriving at the flight path. This multipath arises due to the presence of obstacles in the airport environment as mentioned earlier. The MLS package is augmented by the DME/P signal model developed at Ohio University [11]. This portion serves as the DME/P receiver which is used to determine the guidance errors.

MLS package determines the guidance error by modeling the scatterer at each point on a rectangular grid (Figure 2.5). This grid is defined as the area in which interfering objects will have the maximum effect on the signal. The guidance errors are filtered to obtain the PFE component. The PFE component is then processed as per Federal Aviation Administration (FAA) established DME/P measurement methodology to determine the 95% peak PFE. The 95% peak PFE thus determined is then plotted on the grid to obtain error contours which are used to identify critical areas.
Figure 2.5  Rectangular grid for contour generation.
CHAPTER III

PROBLEM STATEMENT

Critical area determination involves analysis of error contours generated using the MLS mathematical model. Previous critical area work [10] for the angle subsystems has shown that the process of contour generation is time consuming. Table I illustrates the typical time taken for error contour generation on the multi-user IBM 4381 mainframe for different combinations of grid size and spacing. The number of contours that need to be analyzed for the development of critical area criteria makes it impossible to use such small grid spacing. Hence, optimization of contour generation is desired. To optimize the contour generation it is required that the optimal grid size and grid spacing be used so that the number of grid points be minimum [18-22]. To reduce the overall processing time of the above task, a parallel implementation of the above task is done on a Symult S-2010 message passing system.

Grid size is determined based on an estimation of interrogator-obstacle-transponder geometry dependent parameters, scalloping frequency, specular height and time delay. Previous multipath environment studies [12-14] have shown that multipath with delays in excess of 300 ns do not cause significant errors. Further, the interrogator output data filter can reject error components with scalloping frequency
<table>
<thead>
<tr>
<th>Grid Spacing</th>
<th>$X$ (ft)</th>
<th>$Y$ (ft)</th>
<th>5000 x 2000</th>
<th>9000 x 2000</th>
<th>12000 x 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>100,000</td>
<td>347</td>
<td>180,000</td>
<td>833</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>25,000</td>
<td>86</td>
<td>45,000</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>16,000</td>
<td>55</td>
<td>28,800</td>
<td>133</td>
</tr>
</tbody>
</table>

Time taken for one grid point = 5 minutes

(IBM 4381 MAINFRAME)
greater than 10 rad/s [8,18]. The estimation of specular height involves the determination of whether the specular point falls on the obstacle or not. These constraints are used in the determination of grid size used for contour generation.

Determination of optimal grid spacing was first proposed and implemented by Chamberlin [9,10]. In the method he devised, optimal grid spacing was determined based on spatial frequency analysis of guidance error contour data. This method, however, did not take into account the deformation in contours. Further, grid spacing along the x-direction was only determined. In this work a method based on 2-D Fast Fourier Transform is used to determine the optimal grid spacing along both x- and y-directions. Optimal grid spacing is determined in such a way that the reconstruction error does not exceed a prescribed limit. Once the optimal grid spacing is determined, contours are generated with the optimal grid spacing. The deformation between the optimal grid spacing contour and a high resolution contour are determined. This involves tracing the contours and measuring the displacement.

The entire process of contour generation and grid spacing determination requires extensive processing time. To reduce the overall processing time parallel processing is desired. A parallel implementation of the contour generation scheme was presented by Mylvaganam [22] and Mylvaganam and Celenk [23]. This implementation on a 16-node hypercube was complete for the AZ and EL subsystems only. Here, a parallel implementation of the DME/P contour generation package on
the Symult S-2010 message passing multicomputer system [34,35] is presented. A parallel implementation of the optimal grid spacing methodology is also discussed. Performance analysis of this implementation is presented.
DME/P critical area study involves optimal grid size and spacing determination and error contour generation for different airport scenarios and approach procedures. Optimal grid size determination involves computation of scalloping frequency, time delay, and specular height for a specified interrogator-obstacle-transponder geometry and optimal grid spacing requires the spatial frequency analysis of error contours. The optimal grid size is used to determine the optimal grid spacing for the PFE contour generation and the critical area determination. However, the error contour generation is a computationally intensive and time consuming process and it is repeated many times for different scenarios of an airport environment with different obstacles. It is therefore desired that the contour generation process be optimized to reduce the overall computation time for error contour generation by generating the optimal grid spacings and minimizing the degradation on the error contours.

A. Determination of Optimal Grid Size

Determination of optimal grid size involves the specification of a search grid extending over the airport environment. The grid boundaries are obtained by
excluding the areas on the airport proper in which multipath will have negligible effect on the transmitted signal (Figure 4.1). This is determined by the assessment of scalloping frequency, position of specular point, and time delay for a selected transponder-obstacle-receiver geometry. Restrictions on these three parameters [18] need to be satisfied by the points of the search grid in the airport scenario. In Figure 4.1, O represents the origin of the coordinate system which is located at the stop end of the runway, and O' represents the origin of the grid to be used for contour generation.

The first restriction is on the scalloping frequency which is defined as the rate of change of relative phase. Scalloping frequency for an airborne-interrogator-receiver with the velocity component $v_r$ along the runway centerline is given by

$$\omega_s = \frac{2\pi v_r}{\lambda}(\cos \alpha - \cos \beta) \quad (4.1)$$

where $\lambda$ is the wavelength, and $\alpha$ and $\beta$ are the angles sustained by the direct signal and multipath at the receiver as shown in Figure 4.2. The scalloping frequency can be determined for a given transponder-obstacle-receiver geometry provided that the velocity of the receiver and the frequency of the transmitter are known. In this study it is assumed that the velocity and the flight path of the receiver, and the position of the transponder are all specified by a given airport scenario while the
Figure 4.1

Determination of grid size.

- ENTIRE AIRPORT ENVIRONMENT
- COMPUTED GRID SIZE
- ADDITIONAL GRID SIZE FOR VERIFICATION
- ORIGIN OF THE RECTANGULAR GRID
- ORIGIN OF COORDINATE SYSTEM OF PHYSICAL SPACE
Figure 4.2

Determination of scalloping frequency and time delay.
obstacle position is varied to determine the scalloping frequency along the flightpath. If the computed scalloping (angular) frequency is less than 10 rad/sec, then the point is tested for time delay and specular height to determine whether or not it is in the grid; otherwise, the point is rejected. The obstacle is then repositioned at another point and the process is repeated until the entire airport environment is covered.

The second restriction involves the time delay which is a function of the path difference \((l_1 + l_2 - l)\), shown in Figure 4.2, and the propagation velocity of the signal, \(v_s\). Hence the time delay, \(\tau\), is given by

\[
\tau = \frac{l_1 + l_2 - l}{v_s}
\]  

(4.2)

Figure 4.3 shows the constant time delay ellipsoids for a specific interrogator-transponder geometry. The time delay calculated at each point is required to be less than 300 nanoseconds. This condition is a direct result from the fact that multipath delays greater than 300 nanoseconds do not cause time-of-arrival errors [7,8]. If this constraint is satisfied, then the point is tested for the specular height criterion.

The restriction on the specular height is a function of the transponder-obstacle-receiver geometry and the height of the obstacle. Since the critical area study is primarily concerned with surface vehicles (i.e., parked and taxiing aircrafts),
Constant time delay ellipsoids for a specified X-O-R geometry.
the height used is the tail fin height of an aircraft. Figure 4.4 shows the determination of specular height. Figure 4.4 (a) and 4.4 (b) illustrate the geometry in x-y and x-z planes, respectively, and Figure 4.4 (c) shows the similar triangles used to determine the height at which the specular point occurs at the test point. The specular height, \( h_s \), is then given by

\[
h_s = h_i \left( \frac{d_f}{d_r} \right)
\]  

(4.3)

where \( h_i \) is the altitude of the interrogator, \( d_f \) is the distance between the transponder and the grid point, and \( d_r \) is the distance measured from the transponder to the gridpoint then to the receiver. If \( h_s \) is less than the height of the tail fin of the aircraft, then the test point is assumed to lie within the search grid. This test ensures that the specular point lies on the obstacle for at least one transponder-obstacle-receiver geometry.

The entire test process is repeated for all the points in the airport environment. The selected points are enclosed in a rectangular area resulting in the optimal search grid size which is then used for PFE contour generation. Figure 4.5 illustrates this process. The corresponding FORTRAN code is given in Appendix A. The next step in the optimization process involves the determination of optimal grid spacing for this grid size.
Figure 4.4(a)  Determination of specular height.
Figure 4.4(b)

Determination of specular height.
Determination of specular height.

Figure 4.4(c)
Figure 4.5  Flow-chart for determination of grid size.
B. Determination of Optimal Grid Spacing

Once the optimal grid size is determined for a given airport scenario, then the 95% peak PFE contours are generated within this grid size. These error contours represent the peak guidance error as a function of scattering aircraft position and orientation. Generation of these contours requires that the scattering aircraft be shifted by a finite distance (grid spacing) for each simulation. This displacement should be selected in such a way that the PFE contour generation process is not computationally intensive and time consuming and the resultant contours include all the essential information. Hence, the grid spacing has to be optimal to satisfy these requirements.

B.1 Theoretical Formulation

Optimal grid spacing can be determined by PFE contour analysis which can be performed either in the spatial or in the frequency domain. Spatial domain analysis involves the recursive generation of PFE contours by gradually decreasing the grid spacing from its largest value until the process converges (i.e., no significant differences are observed between the successively generated error contours). This process, however, is time consuming and there is no unique methodology to determine whether or not the contour generation process has converged. The
frequency domain analysis involves the Fourier transformation of PFE contours generated on a high resolution grid spacing and the determination of the effective bandwidth from the resultant magnitude spectrum of the PFE contours in the frequency domain. In turn, this results in the Nyquist sampling rates in the corresponding spatial coordinates. This method has an advantage over the spatial domain analysis in the Shannon sampling theorem and signal recovery sense that it provides an optimal grid spacing. Further, it is easier to examine the rate of change of PFE contours as a function of spatial coordinates and also determine the significant frequency components which are related directly to optimal grid spacing determination. However, the approach used in the frequency domain analysis requires the generation of error contours using a high resolution grid spacing which is also a time consuming process. In conclusion, the frequency domain approach is adopted due to the above mentioned advantages. As a result, the problem of optimal grid spacing determination is converted to the problem of determining the effective bandwidth of the PFE contours in the 2-D frequency domain.

B.1.1 Determination of Effective Bandwidth

To determine the 2-D effective bandwidth, we consider the PFE contours (Figure 2.4) as a 2-D function \( \{f(x,y); 0 \leq x \leq x_{\text{max}} \text{ and } 0 \leq y \leq y_{\text{max}}\} \), where \( x \) and \( y \) are the spatial coordinates. The effective bandwidth \([24,25] \) is then defined as the
minimum range of 2-D frequency domain which when used satisfies the bounds on a figure of merit provided by an error criterion. This error criterion is proposed based on the value and shape differences, $E_r(x,y)$ and $E_e(x,y)$, between the signal $f(x,y)$ with spectrum $F(u,v)$ given by

$$F(u,v) = \int \int f(x,y) e^{-j2\pi(ux+vy)} dx \; dy$$

(4.4)

and the signal $f_{\text{rec}}(x,y)$ with restricted spectrum

$$F_{\text{REC}}(u,v) = \begin{cases} F(u,v), & -u_c < u < u_c \quad \text{and} \quad -v_c < v < v_c \\ 0, & |u| \geq u_c \quad \text{and} \quad |v| \geq v_c \end{cases}$$

(4.5)

where $u$ and $v$ represent the spatial frequencies along the $x$- and $y$-directions and $u_c$ and $v_c$ are the effective bandwidth of $f(x,y)$ (Figure 4.6). The error value, $E_r(x,y)$, between the original contour, $f(x,y)$, and the contours, $f_{\text{rec}}(x,y)$, with restricted spectrum is measured at every grid point along the PFE value (vertical) axis

$$E_r(x,y) = |f(x,y) - f_{\text{rec}}(x,y)|, \quad \forall \; (x,y)$$

(4.6)

(see Figure 4.7). The 2-D bandwidth, $(u_c,v_c)$, satisfying the above mentioned
Figure 4.6(a) Illustration of effective bandwidth.
Figure 4.6(b) Illustration of effective bandwidth.
Figure 4.7  Reconstruction error, $E_i(x, y)$, measurement along PFE axis.
constraint is derived as the extension of the effective bandwidth given in [24] for 1-D functions. This 2-D bandwidth is given by

\[ \omega_c = \frac{(N_x)_{\text{max}}}{2\pi^2 \varepsilon_f} \]
\[ \nu_c = \frac{(N_y)_{\text{max}}}{2\pi^2 \varepsilon_f} \]

(4.7)

where \( N_x \) and \( N_y \) denote the total variation of \( f(x,y) \) along the x- and y-axes keeping the other coordinate constant as shown in Figure 4.8. These variations are given by

\[ \sqrt{\sum_{x_{\text{min}}}^{x_{\text{max}}} [f(x,y=b) - f(x_i,y=b)]} \leq N_x \text{ for } y_{\text{min}} \leq y \leq y_{\text{max}} \] (4.8(a))

\[ \sqrt{\sum_{y_{\text{min}}}^{y_{\text{max}}} [f(x=a,y) - f(x,a,y_j)]} \leq N_y \text{ for } x_{\text{min}} \leq x \leq x_{\text{max}} \] (4.8(b))

Here \( a \) and \( b \) are two constant scalar values corresponding to the constant x and constant y-planes and \( N_i \) and \( N_j \) are the total number of maxima and minima along the \( y=b \) and \( x=a \) planes respectively. The total variations given by the above equations is the sum of the magnitude of all the differences between successive maxima and minima in the respective constant x- and constant y-planes. This is the basic procedure that should be followed in the spatial domain analysis mentioned earlier. However, this requires the generation of PFE contours with a very high
Figure 4.8 Measurement of total variations along constant $x = x_1$ and $y = y_1$ planes; Here, $N_y = |f(x_1, y_1) - f(x_1, y_0)| + |f(x_1, y_2) - f(x_1, y_1)| + |f(x_1, y_3) - f(x_1, y_2)| + |f(x_1, y_4) - f(x_1, y_3)|$ and $N_x = |f(x_1, y_1) - f(x_0, y_1)| + |f(x_2, y_1) - f(x_1, y_1)|$. 
resolution grid which is a computationally complex task. Hence, an iterative procedure to determine the effective bandwidth based on spatial frequency analysis of PFE contours is developed. Further, $E_r(x,y)$ is not a sufficient error measurement since the critical area determination is performed in the plane projections of PFE values greater than or equal to a specified PFE value, $c$. Hence it is required that the error, $E_r(x,y)$, be measured in the constant PFE planes between the original $f(x,y) \geq c$ and the corresponding reconstructed $f_{\text{rec}}(x,y) \geq c$ contours. Thus, the determination of $E_r(x,y)$ concerns the regions where the contour values, $f(x,y)$ and $f_{\text{rec}}(x,y)$, are greater than or equal to a specified PFE value. This error measurement reflects the PFE contour deformation (see Figure 4.9) in the constant PFE planes, which may be the major error component in the DME/P critical area determination process.

The error function $E_r(x,y)$ is determined in regions where the constant $f(x,y)$ and $f_{\text{rec}}(x,y)$ contours are greater than or equal to the specified PFE value, $c$. These regions for $f(x,y)$ and $f_{\text{rec}}(x,y)$ are represented by $I(x,y)$ and $I_{\text{rec}}(x,y)$ and are defined as

$$I(x,y) = \begin{cases} 1, & \forall f(x,y) \geq c \\ 0, & \forall f(x,y) < c \end{cases}$$

(4.9)

and
Figure 4.9

Actual and reconstructed (deformed) PFE contours in the constant $f(x,y)$ and $f_{\text{rec}}(x,y)$ planes.
Analysis of the contour deformation is carried out on the boundaries of $I(x,y)$ and $I_{rec}(x,y)$, represented by $b(x,y)$ and $b_{rec}(x,y)$ (see Figure 4.10). These boundaries are formed by detecting 0 to 1 or 1 to 0 transition points of the regions $I(x,y)$ and $I_{rec}(x,y)$. The contour deformation is measured along $b(x,y)$ in the negative direction of the gradient of $I(x,y)$ by considering the PFE contours as level curves (Figure 4.10). In the case of level curves [27-29] the direction of the gradient is towards the functions of higher values. However, the PFE contours are eccentric level curves around the error contours of higher values. As a result, if a point on a lower value contour has moved towards contours of higher values, then the resulting error is negligible since this point will be covered by one of the higher PFE contours. Hence we define the direction for measuring the error function $E(x,y)$ to be along the negative gradient of $I(x,y)$. This may be explained with reference to Figure 4.11 as follows. Let $P(x,y)$ be a point on a PFE contour of value $c_1$, $P_1(x_1,y_1)$ be a point on a neighbouring PFE contour of value $c_2=c_1-\delta c$ and $\delta R$ be the displacement vector from $P$ to $P_1$. Then,

$$
-I \cdot \delta R = \left[ -\frac{\partial I}{\partial x} - \frac{\partial I}{\partial y} \right] \cdot (\delta x \hat{i} + \delta y \hat{j}), \text{ for } (x,y) \in b(x,y)
$$

(4.11)

$$
-I \cdot \delta R = -\frac{\partial I}{\partial x} \delta x - \frac{\partial I}{\partial y} \delta y, \text{ for } (x,y) \in b(x,y)
$$

(4.12)
Figure 4.10 Actual (top) and reconstructed (bottom) contour illustrated as level curves.
Figure 4.11  Negative gradient determination at the boundary points of PFE contours for shape deformation measurement.
-\nabla I \cdot \delta R = -\delta I = -\delta c, \text{ for } (x,y) \in b(x,y) \tag{4.13}

where \( i \) and \( j \) are the unit vectors along the x- and y-axis. Now if \( P_1 \) lies on the same PFE contour as at \( P \), then \( -\delta c = 0 \); i.e., \( \nabla I \cdot \delta R = 0 \). This means that \( \nabla I \) is perpendicular to every \( \delta R \) lying on this contour. Thus \( -\nabla I \) is normal to the contour \( b(x,y) = c \).

Further, if \( P_1 \) lies along the gradient and \( |\delta R| = 1 \) then \( |\nabla I| = \delta c \). Consequently, the Euclidean distance between \( b(x,y) \) and \( b_{\text{rec}}(x,y) \) are computed as a measure of shape deformation along the negative direction of maximum rate of change of the PFE contours.

The gradient of \( I(x,y) \) are computed only on the points of the boundary contour \( b(x,y) \) as

\[
\nabla I(x,y) = \frac{\partial I}{\partial x} i + \frac{\partial I}{\partial y} j, \text{ for } (x,y) \in b(x,y) \tag{4.14}
\]

This provides the direction of maximum rate of change of \( I(x,y) \) at the point \( (x,y) \), which is perpendicular to the tangential line of \( b(x,y) \) at \( (x,y) \). The magnitude and direction of the gradient of \( I(x,y) \) are given by

\[
|\nabla I(x,y)| = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}, \text{ for } (x,y) \in b(x,y) \tag{4.15(a)}
\]
The deformation, $E_s(x,y)$, is illustrated by considering a point $P$ on $b(x,y)$ and a point $P_1$ on $b_{rec}(x,y)$ along the direction of the gradient at $P$ (Figure 4.12). $E_s(x,y)$ can then be expressed as the distance between $P$ and $P_1$, which is given by

$$E_s(x,y) = PP_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2}, \text{ for } (x,y) \in b(x,y) \text{ and } (x_1,y_1) \in b_{rec}(x,y) \quad (4.16)$$

$E_s(x,y)$ thus obtained should not exceed the upper bound, $\epsilon_s$, at every point on $b(x,y)$. The above two constraints are imposed on the selection of the effective bandwidth, which are the upper bounds, $\epsilon_f$ and $\epsilon_v$, on these error values. The proposed constraints are formulated as the selection of a 2-D bandwidth $(u,v)$ (see Figure 4.6) such that if $u \geq u_c$ and $v \geq v_c$, then the pointwise measured error, $E_f(x,y)$, between the original, $f(x,y)$, and the reconstructed, $f_{rec}(x,y)$, satisfies

$$E_f(x,y) = |f(x,y) - f_{rec}(x,y)| < \epsilon_f \quad \forall (x,y) \quad (4.17)$$

and the contour shape deformation, $E_s(x,y)$, measured in $f(x,y) \geq c$ and $f_{rec}(x,y) \geq c$ plane satisfies

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial x}\right) \quad \text{for } (x,y) \in b(x,y) \quad (4.15(b))$$
Figure 4.12 Measurement of deformation, $E_x(x,y)$, along the gradient defined at the boundary points of the PFE contours.
where \( c \) represents the PFE contour value selected for deformation analysis.

The 2-D bandwidth, \((u, v)\), satisfying the above mentioned constraints on \( E_1(x, y) \) and \( E_0(x, y) \) are used to calculate the optimal sampling interval (i.e., optimal grid spacing) \( x_{\text{opt}} \) and \( y_{\text{opt}} \) along x- and y-spatial coordinates in accordance with the Nyquist criterion. The desired sampling rates are given by

\[
\begin{align*}
x_{\text{opt}} &= \frac{1}{2u_c} \\
y_{\text{opt}} &= \frac{1}{2v_c}
\end{align*}
\]

(4.19(a))

(4.19(b))

Here it is assumed that \( u_c, v_c \) are in the unit of cycles/foot. The determination of \( u_c \) and \( v_c \) for a given airport scenario requires the generation of PFE contours in a continuous domain. However, since this is a computationally complex task, we describe a discrete approximation to this process in the following section.
B.1.2 Discrete Approximation

In the discrete domain, the PFE contours are generated on a lattice structure (m,n) with the origin defined at \((x_{\text{min}}, y_{\text{min}})\) of the continuous domain and with the same positive x- and y-directions (see Figure 4.13). Although the function \(f(m,n)\) is discrete in the spatial domain, the PFE values themselves are continuous. The sampling interval on \((x,y)\) to generate the high resolution grid \((m,n)\) is selected in a manner that the number of grid points, \(G\), given by

\[
G = \left( \frac{x_{\text{max}} - x_{\text{min}}}{x_h} + 1 \right) \left( \frac{y_{\text{max}} - y_{\text{min}}}{y_h} + 1 \right)
\]

is not very large. Table I illustrates the number of grid points and the time required to generate the contours for various grid sizes and different sampling intervals. In a discrete lattice, the PFE contours can be expressed as \(\{f(m,n), m=0,1,2...,M-1 \text{ and } n=0,1,2...,N-1\}\) where \(M=1+(x_{\text{max}}-x_{\text{min}})/x_h\) and \(N=1+(y_{\text{max}}-y_{\text{min}})/y_h\). Also, \(m=(x-x_{\text{min}})/x_h\) and \(n=(y-y_{\text{min}})/y_h\), where \(x_h\) and \(y_h\) represent the grid spacings along x- and y-directions. For illustration purposes, the discrete version of contours of Figure 4.14 is also shown in Figure 4.15.

In this 2-D optimization process, generation of high resolution PFE contours for the entire grid is extremely time consuming. To reduce the computation time, the
Figure 4.13 Specification of discrete (bottom) centerline-based high resolution grid (top).
Figure 4.14
A sample DME/P contour.
Figure 4.15

Discrete representation of contour in Figure 4.14.
95% PFE contours, $f(m,n)$, are first determined in a 100ft x 100ft low resolution grid spacing. The area in which the large value PFE contours are eccentric around the PFE contour with the largest value is located in a local window in the search space (Figure 4.16). This local window is specified by $x'_{\min}, y'_{\min}, x'_{\max}, y'_{\max}$. Error contours are then generated within the specified local window with a high resolution grid spacing to accurately describe the PFE contours as a function of a surface vehicle position. Here, a 25ft grid spacing is chosen only to demonstrate the method of optimal grid spacing determination. However, when data from the validated model becomes available the high resolution grid spacing will be determined through convergence. The high resolution contours are denoted as \{f(m',n'), m'=0,1,...,M'-1 and n'=0,1,...,N'-1\}, where $M'=1+(x'_{\max}-x'_{\min})/x_h$ and $N'=1+(y'_{\max}-y'_{\min})/y_h$. Also, $m'=(x'-x'_{\min})/x_h$ and $n'=(y'-y'_{\min})/y_h$, where $x_h=25ft$ and $y_h=25ft$ represent the grid spacing along x- and y-directions and $m'$ and $n'$ are subsets of $m$ and $n$, respectively.

Figure 4.16 shows the local window and the discrete coordinates for the contour in Figure 4.14. In this Figure, $O''$ represents the origin, $(x'_{\min},y'_{\min})$, of the local window.

To determine the 2-D effective bandwidth, $f(m',n')$ is first transformed to the frequency domain using 2-D discrete Fourier transform (DFT). The resultant 2-D spectrum, $F(k',l')$, (Figure 4.17) is given by

$$F(k',l') = \frac{1}{N'M'} \left[ \sum_{m'=0}^{M'-1} \sum_{n'=0}^{N'-1} f(m',n') e^{-j2\pi (k'm'/M', l'n'/N')} \right]$$

(4.21)

where $(k',l')$ are the discrete frequency coordinates corresponding to $(m',n')$ discrete...
Figure 4.16 Local window of the area of maximum rate of change of PFE contours in Figure 4.14.
Figure 4.17 2-D Fourier spectrum of PFE contours in Figure 4.16.
spatial axes [23]. \( F(k', l') \) gives the spatial frequency contents (i.e., magnitude and phase spectrums) of the PFE contours in cycles/foot. To obtain the magnitude spectrum centered at the origin of the frequency axes, the contour data \( f(m', n') \) is multiplied by \((-1)^{m'+n'}\) prior to taking the transform [30]. The resultant transformation is given by

\[
F(k', l') = \frac{1}{N'M'} \left[ \sum_{m'=0}^{M'-1} \sum_{n'=0}^{N'-1} (-1)^{m'+n'} f(m', n') e^{-j2\pi (m'm'/M' + n'n'/N')} \right] ;
\]

\( k' = 0, 1, \ldots, M'-1 \) and \( l' = 0, 1, \ldots, N'-1 \)

The 2-D FFT can be computed as two 1-D DFT's using the separability property of the Fourier Transform. This form is given as

\[
F(k', l') = \frac{1}{N'M'} \left[ \sum_{m'=0}^{M'-1} e^{-j2\pi k'm'/M'} \left( \sum_{l'=0}^{N'-1} f(m', n') e^{-j2\pi l'n'/N'} \right) \right] ;
\]

\( k' = 0, 1, \ldots, M'-1 \) and \( l' = 0, 1, \ldots, N'-1 \)

In the computer implementation, the 2-D DFT is obtained by determining \( N \) number of \( M \)-point 1-D DFT's (\( N \) row-wise DFT's) followed by the computation of \( M \) number of \( N \)-point 1-D DFT's (\( M \) column-wise DFT's).

To determine the optimal grid spacing, \( F(k', l') \) is low pass filtered using a 2-D ideal low pass filter. The frequency response, \( H(k', l') \), of the filter is defined in a
square region of the 2-D discrete frequency domain (see Figure 4.18) as a separable ideal low pass filter

\[ H(k',l') = \begin{cases} 1, & |k'| < k'_c \text{ and } |l'| < l'_c \\ 0, & k'_c < |k'| \text{ and } l'_c < |l'| \end{cases} \] (4.24)

where \((k'_c,l'_c)\) represents the discrete domain cut-off frequencies corresponding to \((u_c,v_c)\) in the continuous domain. Selection of the square region of support for the ideal low pass filter ensures that the resultant Nyquist sampling rates along the x and y-spatial axes are identical and the sampling in the spatial domain is a 2-D square sampling [30].

The low pass filtered signal \(F_{LPP}(k',l')\) (Figure 4.19) is given by

\[ F_{LPP}(k',l') = F(k',l').H(k',l') \quad k' = 0,1,...,M'-1 \text{ and } l' = 0,1,...,N'-1 \] (4.25)

The reconstructed signal \(f_{rec}(m',n')\) is determined using inverse 2-D DFT as
Figure 4.18  Magnitude response of 2-D ideal low pass filter.
Figure 4.19  Low pass filtered spectrum of PFE contours in Figure 4.16.
The pointwise reconstruction error $E_{r}(m', n')$ can then be expressed as

$$E_{r}(m', n') = | f(m', n') - f_{rec}(m', n') | ; \quad m' = 0, 1, \ldots, M' - 1 \text{ and } n' = 0, 1, \ldots, N' - 1$$

(4.27)

The above discussion implies that $E_{r}(m', n')$ is a function of the low pass filter characteristic, $H(k', l')$, which is selected in accordance with the peak reconstruction error criterion given by

$$\frac{E_{r}(m', n')}{f(m', n')} < \frac{\epsilon_{f}}{f(m', n')} ; \quad m' = 0, 1, \ldots, M' - 1 \text{ and } n' = 0, 1, \ldots, N' - 1$$

(4.28)

This criterion ensures that the percentage reconstruction error is less than an empirical upper bound (e.g., 10%, 20% or 30%) at all points $(m', n')$ in the search grid. The determination of $E_{r}(m', n')$ is illustrated in Figure 4.20. Further, $H(k', l')$ is also constrained by the requirement that the contour deformation, $E_{s}(m', n')$, be less than the selected upper bound ($\epsilon_{s} = 100$ ft, based on FAA accuracy requirements for critical area determination). To formulate this constraint, the constant PFE regions $I(m', n')$ and $I_{rec}(m', n')$ are obtained by thresholding $f(m', n')$ and $f_{rec}(m', n')$ in the discrete domain as
Figure 4.20

Illustration of steps involved in the determination of optimal grid spacing and error measurement.
where \( c \) is the selected PFE value for analysis. The boundary points, \( b(m',n') \) and \( b_{rec}(m',n') \), of \( I(m',n') \) and \( I_{rec}(m',n') \) are generated using an edge detection method which classifies a point of value 1 an edge point if any of its vertical or horizontal neighbors is 0 [30]. This can be expressed as

\[
b(m',n') = I(m',n').(I(m',n') \oplus I(m'-1,n') + I(m',n') \oplus I(m'+1,n') \\
+ I(m',n') \oplus I(m',n'-1) + I(m',n') \oplus I(m',n'+1))
\]

(4.31)

\[
b_{rec}(m',n') = I_{rec}(m',n').(I_{rec}(m',n') \oplus I_{rec}(m'-1,n') + I_{rec}(m',n') \oplus \\
I_{rec}(m'+1,n') + I_{rec}(m',n') \oplus I_{rec}(m',n'-1) + I_{rec}(m',n') \\
\oplus I_{rec}(m',n'+1))
\]

(4.32)

The binary edge detection templates developed for this operation are illustrated in Figure 4.21. The boundaries \( b(m',n') \) and \( b_{rec}(m',n') \) are obtained as shown in Figure 4.22.
Figure 4.21  Binary edge detection templates developed for detecting the boundary points of PFE contours.
Figure 4.22  Boundaries of actual and reconstructed contours extracted using templates of Figure 4.21. Here, 1 and 2 represent the actual and reconstructed contours respectively, while 3 represents the superimposed points of the actual and reconstructed contours.
Given \( b(m',n') \) and \( b_{\text{rec}}(m',n') \), the deformation in contours is measured along the gradient determined at each boundary point, \( P \), of \( I(m',n') \) on \( b(m',n') \). The gradient is computed at \( P \) using the Sobel operator within the 3x3 local region as

\[
\nabla I(m',n') = \left[ \frac{\partial I}{\partial m'} \quad \frac{\partial I}{\partial n'} \right], \text{ for } (m',n') \in b(m',n')
\]  

(4.33)

where

\[
\frac{\partial I}{\partial m'} = \frac{(I(m'+1,n'-1) - I(m'-1,n'-1)) + 2(I(m'+1,n'+1) - I(m'-1,n'+1))}{(m',n')}
\]

(4.34(a))

\[
\frac{\partial I}{\partial n'} = \frac{(I(m'-1,n'+1) - I(m'-1,n'-1)) + 2(I(m'+1,n'+1) - I(m'+1,n'-1))}{(m',n'}
\]

(4.34(b))

The templates used in this computation are shown in Figure 4.23. The magnitude and the direction of the gradient are obtained as

\[
|\nabla I(m',n')| = \sqrt{\left(\frac{\partial I}{\partial m'}\right)^2 + \left(\frac{\partial I}{\partial n'}\right)^2}, \text{ for } (m',n') \in b(m',n')
\]

(4.35(a))
Figure 4.23  Sobel templates for gradient determination along the boundary points of PFE contours.
\[ \theta = \tan^{-1}\left( \frac{\partial E_0}{\partial m'} \right), \text{ for } (m',n') \in b(m',n') \]  

(4.35(b))

The deformation, \( E_s(m',n') \), of \( b(m',n') \) along the gradient is measured as the Euclidean distance between a point \( P \) on \( b(m',n') \) and the corresponding point \( P' \) on \( b_{rec}(m',n') \) in a local window (Figures 4.24 and 4.25) defined by the components of the gradient as

\[ E_s(m',n') = \overline{PP'} = \sqrt{(m'-m_i)^2 + (n'-n_i)^2}, \]

for \( (m',n') \in b(m',n') \) and \( (m_i,n_i) \in b_{rec}(m',n') \)  

(4.36)

The maximum value of deformation (Figures 4.24 and 4.25) thus obtained should not exceed \( \varepsilon_s = 100 \text{ ft} \) to satisfy the accuracy requirements for critical area study:

\[ E_s(m',n') = \overline{PP'} = \sqrt{(m'-m_i)^2 + (n'-n_i)^2} \leq \varepsilon_s \leq 100\text{ ft} \]  

(4.37)

for \( f(m',n') \geq c, \) \( f_{rec}(m',n') \geq c, \) \( (m',n') \in b(m',n') \) and \( (m_i,n_i) \in b_{rec}(m',n') \)

Figure 4.26 illustrates the steps involved in the determination of contour deformation.

The bandwidth of the low pass filter is varied until the peak reconstruction error and the deformation criteria are met. The 2-D bandwidth, \( (k',l') \), of the filter satisfying these constraints is related to \( (u_v,v_c) \) by
Figure 4.24 Measurement of shape deformation, $E_z(x,y)$, along the negative gradient defined at the boundary points of the actual contours.
Figure 4.25 Measurement of shape deformation, $E_s(x,y)$, along the positive gradient defined at the boundary points of the actual contours.
Figure 4.26 Flow-chart of the steps used in the measurement of deformation along the gradient.
The resultant optimal grid spacings $(x_{opt}, y_{opt})$, computed using equation 4.19, are expected to yield a minimum number of grid points, $G$, given by equation (4.20) to describe the error contours with an accuracy sufficient for the critical area work. This iterative process is described by the flowchart of Figure 4.27.

\begin{align*}
    u_c &= \frac{k_c'}{M x_h} \\
    v_c &= \frac{l_c'}{N y_h}
\end{align*}

(4.38)

B.1.3 Verification of Optimal Grid Spacing

The accuracy of the above iterative procedure is determined by comparing the contours $f_{opt}(m, n)$ generated using the optimal grid spacings with the high resolution contours $f(m', n')$. This analysis dictates any significant information loss in $f_{opt}(m, n)$ compared to $f(m', n')$. This involves the measurement of the error functions $E_{f}(m', n')$ and $E_{s}(m', n')$ between $f_{opt}(m, n)$ and $f(m', n')$ in the high resolution grid $(m', n')$ specified by the optimal sampling intervals. Here it is assumed that the optimal grid spacing $(x_{opt}, y_{opt})$ is multiple values of the grid spacing $(x_h, y_h)$ used for generating the high resolution contours; i.e.,
Figure 4.27  Flow-chart of the steps involved in the determination of optimal grid spacing.
where \( k_x \) and \( k_y \) denote the integer multiplication factors and are specified within the local window. The optimal grid spacings in the region outside the local window are given as

\[
\begin{align*}
    x_{opt} &= k_x x_i \\
    y_{opt} &= k_y y_i
\end{align*}
\] (4.40)

where \( x_i \) and \( y_i \) represent the spacings on the low resolution grid and \( K_x \) and \( K_y \) represent the corresponding multiplication factors for this region. The optimal grid spacing contours are first interpolated to obtain the interpolated optimal grid spacing contour, \( f_{ip}(m',n') \). Bilinear interpolation [26] used for this purpose is given by

\[
f_{ip}(m,n) = (1-\Delta_m)(1-\Delta_n)f_{opt}(n_1, n_2) + (1-\Delta_m)\Delta_n f_{opt}(n_1, n_2+1) + \\
\quad \Delta_m(1-\Delta_n)f_{opt}(n_1+1, n_2) + \Delta_m\Delta_n f_{opt}(n_1+1, n_2+1)
\]

where \( \Delta_m = \frac{m-n_1 T_1}{T_1} \) and \( \Delta_n = \frac{n-n_2 T_2}{T_2} \) (4.41)

and is illustrated in Figure 4.28.
Figure 4.28  Bilinear interpolation to interpolate optimal contour to high resolution grid.
The error function $E_t(m',n')$ is then given by

$$E_t(m',n') = |f(m',n') - f_{opt}(m',n')| \quad ; \quad m' = 0,1,..,M'-1 \text{ and } n' = 0,1,...,N'-1$$ \hspace{1cm} (4.42)

and the contour deformation $E_s(m',n')$ can then be expressed as

$$E_s(m',n') = PP_1 = \sqrt{(m-m_1)^2 + (n-n_1)^2}, \text{ for } (m,n) \in b(m',n') \text{ and } (m_1,n_1) \in b_{opt}(m',n')$$ \hspace{1cm} (4.43)

where $PP_1$ represents the distance between the points $P$ and $P_1$ on $b(m',n')$ and the linearly interpolated $b_{opt}(m,n)$ i.e., $b_{opt}(m',n')$ respectively (Figure 4.29). The determination of $b(m',n')$ and $b_{opt}(m',n')$ involves the steps illustrated in the previous section. The entire process is illustrated in Figure 4.30.

Once the pointwise error and the deformation are determined, these values are checked against tolerances required for the critical area study. If the tolerances are satisfied then the grid spacing obtained from the above procedure is verified; otherwise the bandwidth of the filter is increased and a larger grid spacing is determined. This iterative procedure is repeated until the comparison between the optimal grid spacing contour and the reconstructed contour satisfies the tolerances.
Figure 4.29 Measurement of deformation along the positive and negative gradient between the high resolution and the interpolated optimal grid spacing contour.
Figure 4.30 Steps involved in the comparison of the high resolution and the optimal grid spacing contour.
Critical area study uses the Microwave Landing System (MLS) computer model to generate error contours. This model simulates the effect of multipath arising from obstacles and terrain in the airport environment. This chapter describes the utilization of the MLS computer model to generate error contours to study multipath arising from parked aircrafts.

A. MLS Simulation Package Structure

The MLS model consists of four different modules; the transmitter or propagation module, the transmitter plotter, the receiver or system module and the receiver plotter. The structure of the model and the connection between the various units are given in Figure 5.1. The functions of the four modules are explained below.

A.1 Transmitter or Propagation Module

The transmitter stage models the pulses transmitted from the airborne
Figure 5.1  MLS computer model structure.
interrogator and their propagation in an airport environment. The signal at the ground transponder is then formed by the superposition of multipath and direct signal components. The multipath components are due to various phenomena such as reflection, scattering, and shadowing effects from terrain, buildings, and aircrafts. A formatted input file (Figure 5.2) to the transmitter portion of the MLS has been used to model these phenomena.

The formatted input file provides different sections to model the scenario including the flight path, the obstacles causing different kinds of multipath, the operating parameters for the different antenna subsystems including frequency of operation and antenna patterns. These sections are described as:

Section 0 provides a brief overview of the scenario data in terms of length and width of the runway, glide path angle, decision height and name of the scenario. Section 1 is subdivided into two sections (a) and (d). Section (a) describes the position of azimuth and elevation subsystems and their operating frequencies while section (d) provides the position and operating parameters for DME/P. Section 1a specifies pulse shape and encode time for the DME/P pulse. Section 2 on being selected enables ground reflections to be processed. The ground, itself, is modeled by its dielectric constant and roughness height. Section 3 input consists of coordinates of scattering aircraft. A maximum of ten aircrafts can be modeled. The model also offers the possibility of simulating different types of aircrafts which can be oriented in any direction depending on the scenario being modeled. Section 4 contains the coordinates of scattering buildings which are modeled as plates in the transmitter portion. A maximum of ten scattering buildings are allowed. Information provided includes building tilt angle, height difference between the base of the building and the transmitter, and dielectric constant of the building surface. Sections 5 and 6: Specular scattering from rectangular and triangular ground surfaces are modeled. The input consists of the coordinates of the plate, roughness height and dielectric constant. Section 7 models diffuse scattering from ground. Section 8 simulates shadowing from aircraft. Input includes the coordinates of the
SECTION 0
= SCENARIO DATA
  RUN ID#: ####
  TITLE: ____________________________________________
  AIRPORT: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
  RUNWAY: rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
  LENGTH: 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
SECTION 8
= SHADOWING BY AIRCRAFT
rrr - RUN SHADOWING AIRCRAFT (yes, no)
## X-VALU Y-VALU Z-VALU VEL ANG AT
--------- --------- --------- --------- ---------
nn sax2sax1 say2say1 say2say1 vvvvv aaaa tt
sax2sax2 say2say2 say2say2 saz2saz2

SECTION 9
= SHADOWING BY BUILDINGS
rrr - RUN SHADOWING BUILDINGS (yes, no)
## X-LEFT Y-LEFT X-RIGHT Y-RIGHT ELV HGT
--------- --------- --------- --------- ---------
nn xxxxxxxxxx yyyyyyyyy xxxxxxxxxx yyyyyyyyy eeee hhhhh

SECTION 10
= SHADOWING BY RUNWAY HUMP
rrr - RUN SHADOWING HUMP (yes, no)
X-FRONT Z-FRONT X-HUMP Z-HUMP X-BACK Z-BACK
--------- --------- --------- --------- ---------
xfxfxfxf zffzffzff xhhxhxxh xhhxhxxh xhxhxhxxh xbbxbbx

SECTION 11
= RUNWAY PROFILE
## XP YP ZP
--------- --------- ---------
rp xxxxxxxxxx yyyyyyyyy zzzzzzzz

SECTION 12
= FLIGHTPATH
FAF : fffff NAUTICAL MILES
DATUM : xxxxxxxx yyyyyyyyy zzzzzzzz
TYPE : ffff (measured, distance, orbit, radial, segmented, straight)
* VELOCITY : vvvvvvvv
INCREMENT: lllllllll
DATA RATE: dddddddd
* IF "straight" SUFFICIENT DATA IS AVAILABLE TO COMPUTE FLIGHTPATH
* IF "radial" ENTER ANGLE, ELEVATION & STARTING AND ENDING DISTANCES
* (nm from dme/p)
ANGLE: aaaaaaaaa
SDIST: dddddddd
EDIST: dddddddd
ELEV : eeeeeee
* IF "orbit" ENTER RADIUS (nm from dme/p) & ELEVATION
RADIUS: rrrrrrrrrrrr
ELEV : eeeeeee
* IF "measured" X,Y,Z COORDINATES AND TIME WILL BE READ FROM UNIT 15
* WITH VELOCITY AND DATA INCREMENT COMPUTED FROM INPUT
* IF "segmented" or "distance" ENTER SEGMENT #, X,Y,Z, VELOCITY AND
INCREMENT
## XS YS ZS VEL INC
--------- --------- ---------
nn xxxxxxxxxx yyyyyyyyy zzzzzzzz vvvvvvvv lllllllll

SECTION 13
= FLIGHTPATH AND AIRPORT LAYOUT AXIS LIMITS
* FLIGHTPATH PLOTS:
X/Y PLOT X/Z PLOT D/Z PLOT
--------- --------- ---------
MINIMUM X VALUE : xyyyyyyyyy xyyyyyyyyy xyyyyyyyyy
UNITS PER INCH : xyyyyyyyyy xyyyyyyyyy xyyyyyyyyy
MINIMUM Y VALUE : xyyyyyyyyy xyyyyyyyyy xyyyyyyyyy
UNITS PER INCH : xyyyyyyyyy xyyyyyyyyy xyyyyyyyyy
* AIRPORT LAYOUT PLOT:
--------- ---------
X/Y PLOT
--------- ---------
MINIMUM X VALUE : xyyyyyyyyy
UNITS PER INCH : xyyyyyyyyy
MINIMUM Y VALUE : xyyyyyyyyy
UNITS PER INCH : xyyyyyyyyy

Figure 5.2(contd) Formatted transmitter model input file.
SECTION 14
= ANGLE EQUIPMENT AXIS LIMITS

* MULTIPATH DIAGNOSTIC PLOTS:

<table>
<thead>
<tr>
<th>M/D</th>
<th>SEP ANG</th>
<th>SHADOWING</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdmd</td>
<td>sasasa</td>
<td>shshsh</td>
</tr>
<tr>
<td>mdmd</td>
<td>sasasa</td>
<td>shshsh</td>
</tr>
<tr>
<td>mdmd</td>
<td>sasasa</td>
<td>shshsh</td>
</tr>
<tr>
<td>mdmd</td>
<td>sasasa</td>
<td>shshsh</td>
</tr>
</tbody>
</table>

  MINIMUM X VALUE: mdmdmdmd sasasasas shshshsh
  UNITS PER INCH: mdmdmdmd sasasasas shshshsh
  MINIMUM Y VALUE: mdmdmdmd sasasasas shshshsh
  UNITS PER INCH: mdmdmdmd sasasasas shshshsh

* RECEIVER ERROR & FILTERED ERROR PLOTS:

  RAW       PFE       CMN
  --------  --------  --------
  mdmd      pppppppp  cccccc
  mdmd      pppppppp  cccccc
  mdmd      pppppppp  cccccc
  mdmd      pppppppp  cccccc

SECTION 15
= DISTANCE MEASURING EQUIPMENT AXIS LIMITS

* MULTIPATH DIAGNOSTIC PLOTS:

<table>
<thead>
<tr>
<th>M/D</th>
<th>TIM DELAY</th>
<th>SHADOWING</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdmd</td>
<td>mdmdmdmd</td>
<td>tdttdtdt</td>
</tr>
<tr>
<td>mdmd</td>
<td>mdmdmdmd</td>
<td>tdttdtdt</td>
</tr>
</tbody>
</table>

  MINIMUM X VALUE: mdmdmdmd tdttdtdt shshshsh
  UNITS PER INCH: mdmdmdmd tdttdtdt shshshsh

* INTERROGATOR ERROR & FILTERED ERROR PLOTS:

  RAW       PFE       CMN
  --------  --------  --------
  mdmd      pppppppp  cccccc
  mdmd      pppppppp  cccccc
  mdmd      pppppppp  cccccc
  mdmd      pppppppp  cccccc

END DATA

Figure 5.2(contd)  Formatted transmitter model input file.
shadowing aircraft, velocity and angle. The velocity factor helps to simulate both taxiing and parked aircrafts.

Section 9 models shadowing from buildings. The coordinates, height and elevation of the building are provided.

Section 10 simulates shadowing from runway humps. The coordinates for the front and back of the hump are provided along with coordinates for the hump itself.

Section 11 provides the runway profile information.

Section 12 models the flightpath. The user is permitted to simulate different types of flight path. The six different flight paths available are measured, distance, orbit, segmented, straight and radial.

Section 13: The information is utilized to produce different plots from the output data.

The transmitter portion simulates the propagation of DME/P signal in the environment modeled by the input file, producing two output files which contain information about the multipath components at each flight path point. This information which includes the Multipath to Direct signal ratio (M/D), phase of signal, conical separation angles and time delay is further processed by the receiver module and the transmitter plotting routine to determine guidance errors and multipath plots respectively.

A.2 Receiver Module

The DME/P receiver module (Figure 5.3) accepts the output of the transmitter module and calculates the guidance error resulting from multipath effects. The receiver combines the direct and multipath signals received to form one composite pulse. This pulse is video filtered to obtain the output pulse which is then processed
Figure 5.3  DME/P receiver model.
using a Delay-Attenuate-Compare circuit (Figure 5.4) to obtain the time delay. This circuit compares a delayed version of the pulse to an attenuated version of the same pulse. Pulse arrival is declared when the delayed signal amplitude exceeds the attenuated signal. The time delay multiplied by the velocity of light gives the raw guidance error. This raw error is then filtered into low frequency (PFE) and high frequency components (CMN) using filter circuits. The PFE component thus obtained represents the error in feet by which the aircraft is displaced from its intended position. Thus given a flightpath scenario, the transmitter module and the receiver module together produce the guidance error.

The receiver module produces the error output file. These errors are processed using additional routines to produce the 95% peak error caused. Thus, the BMLS package is supported with supplementary routines to complete the contouring package. The implementation of the contouring package along with the additional routines is described in the following section.

B. Contour Generation

The MLS computer model is used for generating error contours for the critical area study which involves the identification of areas in the airport environment in which scatterers can cause guidance error for an approaching aircraft.
Figure 5.4  Principle of delay-attenuate-compare (DAC) circuit.
Critical area study requires that a scatterer be positioned at various points in the airport environment and the guidance error due to the scatterer determined.

A surface vehicle (say a B-747 aircraft) is positioned on the search grid (Figure 5.5) to study the effect of multipath on the DME/P signals. The position of the scatterer is entered into Sections 3 and 8 of the input file. Error in the elapsed time measurement, and thus the measured range can occur due to DME/P pulse distortion by multipath. The raw DME error is defined as the difference between the actual distance and that measured by the DME. The output of the model consists of flight path points and the error as seen by the aircraft at each point.

The windowing routine processes the error along the flight path obtained from the receiver to determine the Path Following Error (PFE) component (Figure 5.6). DME/P measurement methodology requires that the filtered PFE values do not exceed the specified error limits for a total period that is more than 5 percent of the evaluation time interval (T=40 secs for initial approach and T=10 secs for final approach mode). The error value that fails this requirement is the 95% peak error for that flight trace. The 95% peak error is determined for various decision heights for each flight trace. This requires the analysis of filtered PFE from the starting point on the flight path up to the corresponding decision height (see Figure 5.7). Thus, the 95% peak error is obtained at different decision heights for a particular location of the obstacle in the airport environment. The scatterer is then moved to
Figure 5.5  Search grid for contour generation scheme.
Figure 5.6  Sliding window for 95% peak PFE determination.
another point and the process repeated until the entire grid is covered. The entire process is illustrated in Figure 5.8.

The 95% peak errors obtained are plotted to obtain error contours which represent the guidance error caused by the multipath arising from the presence of the scatterer. These contours are analyzed to identify areas causing significant error. The determination of final critical area criteria will then involve analysis of error contours generated for various combinations of runway length, aircraft type, aircraft orientation and different approaches.

As a part of the project Job Control Language (JCL) was developed on the IBM 4381 mainframe at Ohio University to automate the contour generation process. This required the installation and validation of the Baseline Microwave Landing System math model (BMLS math model) on the mainframe. Controlling Execs were developed and existing plotting routines (Appendix A) were modified to complete the package for contour generation for critical area study.
Figure 5.8 Flow-chart of the contour generation process.
CHAPTER VI

PARALLEL IMPLEMENTATION OF DME/P CONTOUR GENERATION

As mentioned in Chapter III, the process of obtaining the PFE contours is computationally intensive and time consuming [9,10,22-25]. The time taken for the generation of an error contour for a grid of size $x_{\text{max}}, y_{\text{max}}$ is proportional to the number of grid points $G$ given by equation 4.20. Parallel processing offers an alternate solution to reduce the overall processing time, while maintaining a proper balance between performance and cost. The effectiveness of a parallel implementation of the PFE contour generation task was first illustrated by Mylvaganam [22], and Celenk and Mylvaganam [23]. This implementation was done on the Ametek S-14 16-node hypercube system for the AZ and EL subsystems of MLS without the inclusion of the DME/P subsystem. Further, due to memory constraints this implementation did not include the windowing routine required to generate the 95% peak PFE values. Here a parallel implementation of the DME/P contour generation simulation software was realized in the S-2010 message passing multicomputer system [34,35], which has certain advantages over the implementation on the hypercube system. First, message passing architecture utilizes special message routing hardware which makes the underlying topology transparent to the user while providing a flexible system configuration. Also a message passing system has an efficient communications network [38] which achieves higher data rates than the
A hypercube network. The size of the local memory in this system is also large enough to accommodate sufficiently large executable codes. In the following sections, first we describe the organization of S-2010 multicomputer message passing system. Then we present the parallel implementation of the model which includes all the essential contouring and windowing routines. Finally, we discuss the parallel implementation of the optimal grid spacing determination and give a detailed performance analysis of this implementation.

A. S-2010 Message Passing Multicomputer System

S-2010 is a logically and physically distributed multicomputer system with 16-nodes connected in a 2-D mesh configuration. S-2010 system is a multi-user single instruction multiple data (SIMD) machine with each processing element (PE or node) executing the same program on different data. The basic SIMD system organization is shown in Figure 6.1. In this organization, all the nodes are connected to one another through an interconnection network, and a common bus connects the nodes to the control unit (i.e., the corner node) which can communicate with the host system. Each node of the system consists of a processor and a local memory. This is essentially a distributed memory architecture which not only avoids memory contention but also provides an easily expandable system architecture. The major disadvantage of this architecture however is the interconnection network which is
Figure 6.1  Conventional SIMD system organization.
complex and expensive. In addition, this loosely coupled system does not have a
global memory and interprocessor communication is achieved through message
passing [35].

The S-2010 is primarily an asynchronous SIMD machine with a set of
dedicated processors (one per node) for data routing purposes. The message routing
processors also allow individual nodes to access disk files on the host directly. The
corner node of the S-2010 is connected to the host system (Sun 3/80) (Figure 6.2)
which supports program development, debugging, diagnostics and the application
layer of the operating system. The host is capable of downloading and receiving data
to and from the nodes, downloading executable code, and providing the necessary
control signals to the node processors. The host runs the parent program which
controls the child process executed on the nodes. Each node of S-2010 consists of
a 25MHz 68020 processor, a 68881 floating-point coprocessor, a message packet
processor (MPP), and 1.5 Mbytes of dual ported local memory which is expandable
to 8 Mbytes. It also has ports to connect a vector floating point accelerator and disks.
The node board block diagram is given in Figure 6.3. MPP shares memory with the
node processor and coordinates message passing on a First-In-First-Out (FIFO) basis
under the guidance of the operating system [34].

The nodes are interconnected through a gigalink network which consists of
Automatic Message Routing Devices (AMRD) connected in a 2-D mesh organization
Symult S-2010 system configuration.

Figure 6.2
Figure 6.3  Node board block diagram.
An AMRD has 5 bidirectional input-output channels, four of which are connected to the other AMRDs while the fifth input and output channel is connected to the corresponding node through the MPP. This network provides bidirectional communication at the rate of 20Mbytes/sec. The nodes communicate with each other via messages which are divided into smaller packets, each of which consists of header information, message, and a tail bit. The header includes operating system information and two routing bytes. The Reactive Kernel (RK) operating system of S-2010 places packets on a send queue. MPPs send packets from the send queues and AMRD reroutes them first in the x-direction and then in the y-direction until the routing bytes of messages become zero. Multiple messages are multiplexed in a single channel. S-2010 uses the worm-hole message routing which in contrast to the store-and-forward routing does not interrupt the node process to receive and forward an incoming message [34,35,38]. The use of worm-hole routing eliminates delay in intermediate nodes and also overcomes the problem of excessive memory utilization at intermediate nodes. At the destination, the packets are placed in a receive queue from which RK reconstructs the message and places it in the node's local memory.

As mentioned above the RK operating system of the S-2010 plays a significant role in interprocessor communication. The various functions of RK are included in the computational node software structure shown in Figure 6.5. The RK consists of an inner kernel which resides on the nodes. This portion of the RK is responsible for assembling/disassembling messages into/from packets. It also ensures handler
Figure 6.4

2-D mesh organization of AMRD's.
Figure 6.5  Structure of reactive kernel operating system.
activation on receipt of a message. The next layer of the RK consists of the kernel service routines while the outermost layer of the RK consists of handlers and file servers. Some of the functions of the handlers include dynamic memory allocation for messaging, creation and termination of processes, retrieval and monitoring of node information.

B. Parallel Decomposition of DME/P Contour Generation

Parallel decomposition of the contour generation task involves two related issues; domain partitioning and data transfer [22-25,36-37]. The S-2010 provides a flexible system configuration to tackle these issues and to utilize of the system more efficiently. The message packet processor at each node in conjunction with the Gigalink network provides faster and efficient data transfer and access to disk files on the host. This simplifies the problem of mapping the decomposed task domain onto the nodes. In domain partitioning, the entire grid is divided into different rectangular regions, each of which is assigned to a processing element (PE). Thus all the processing elements execute the same DME/P contouring package for the specified groups of grid points in the given airport scenario. This eliminates the problem of imbalance load distribution among the nodes.
Three possible decomposition schemes (vertical, horizontal and block decomposition) are considered for contour generation as shown in Figure 6.6. The selection of a particular decomposition scheme over the others is based on system performance measured by the speedup factor. The speedup factor, $S$, is defined as

$$S = \frac{T(1)}{T(n)} = \frac{T_{\text{comp}}}{\frac{T_{\text{comp}}}{n} + T_{\text{comm}}}$$  \hspace{1cm} (6.1)$$

Here, $T(1)$ is the time taken by a single node to complete a task (i.e., $T_{\text{comp}}$), $T(n)$ is the time taken to perform the equivalent calculation on n-nodes, and $T_{\text{comm}}$ is the interprocessor communication time required for data transfer among the n-nodes during the execution of a particular task. The maximum achievable speedup (i.e., $S=n$) is obtained from a parallel implementation if $T_{\text{comm}}$ is kept minimum (i.e., $T_{\text{comm}}=0$). In the case of the contour generation all the nodes execute the same code and there is no interprocessor communication involved in this process. Although all three decomposition schemes are equally efficient for contour generation, the 2-D mesh configuration of the S-2010 offers a simpler problem domain to architecture mapping strategy. However, the vertical or horizontal decomposition is preferred over the block decomposition for the 2-D FFT computation due to the fact that the former decomposition schemes require relatively less number of interprocessor communication and data rearrangement compared to the latter scheme. On the other hand, the block decomposition is preferred for the contour analysis scheme which involves window operations for error measurements. In this case, block
Figure 6.6  Decomposition of search grid for parallel implementation.
decomposition provides higher speedup for window operations as illustrated in [37] compared to the other two (horizontal and vertical decomposition schemes). The use of two processors (node processor and the message packet processor) on the S-2010 message passing system provides a mechanism to improve system performance by overlapping communications and computation whenever interprocessor communication is needed. In conclusion, a horizontal decomposition of the grid is used for the contour generation and block decomposition is adopted for the comparison of actual and reconstructed contours in this parallel implementation.

C. DME/P Contour Generation on S-2010

The contour generation process illustrated in Figure 5.8 is mapped onto the S-2010 system by developing a parent program which runs on the host system and a child program which runs on the processing elements (PE's) of S-2010. The structure of the parent and child programs are shown in Figure 6.7.

The main function of the parent program is to transfer the executable code of the child program (Propagation and DME/P receiver models) from disk to nodes. The parent program initiates the execution of the child process on the nodes of S-2010. The host process also controls the assignment of grid points to the nodes using the horizontal decomposition shown in Figure 6.6. Sixteen grid points, one from each
Figure 6.7 General structure of parallel program involving the parent and the child programs.
horizontal group, are assigned to the nodes of the S-2010. Once the assignment of
grid points is completed, the host process prepares sixteen different input files (one
per grid point) corresponding to the 16 grid points. Each node accesses the
respective input file based on its logical node number. The child program then
computes the 95% peak PFE values for the grid point assigned using the propagation
and DME/P receiver models.

The parallel implementation of contour generation on a grid of \( G \) points is
derived from the sequential implementation which is given as

\[
\begin{align*}
\text{DO} & \quad 100 \ I = 1,G \\
& \quad \text{READ DATA FOR OBSTACLE AT POINT } I \\
& \quad \text{EXECUTE BMLS TRANSMITTER MODEL FOR OBSTACLE AT POINT } I \\
& \quad \text{EXECUTE DME/P RECEIVER MODEL FOR OBSTACLE AT } I \\
& \quad \text{EXECUTE 95% WINDOW} \\
& \quad \text{WRITE OBSTACLE POSITION AND 95% PFE} \\
\end{align*}
\]

100
CONTINUE
END

The parallel implementation can then be written in concurrent programming notation
as

\[
\begin{align*}
\text{COBEGIN:} \\
\text{DO} & \quad 100 \ L=1,G/n \\
& \quad K=P:(xy) \\
& \quad \text{READ DATA FOR OBSTACLE AT POINT } K \\
& \quad \text{EXECUTE BMLS TRANSMITTER MODEL FOR OBSTACLE AT POINT } K \\
& \quad \text{EXECUTE DME/P RECEIVER MODEL FOR OBSTACLE AT } K \\
& \quad \text{EXECUTE 95% WINDOW} \\
& \quad \text{WRITE OBSTACLE POSITION AND 95% PFE} \\
\end{align*}
\]
In this implementation, the function \( P:(x,y) \) represents the assignment of a grid point to a particular node using the horizontal decomposition.

The object code of the above implementation is first downloaded to the nodes. Then the input files corresponding to the grid points assigned to each node are downloaded. These input files contain the airport scenario data for the corresponding grid point. Each node then executes the object code for a particular grid point. Once all the nodes complete their tasks, they signal the corner node which requests the parent to assign each child with a new grid point from the corresponding horizontal group (Figure 6.6). The host process waits for the nodes to signal the completion of the task. Upon receipt of the completion signal, the host process sorts the 95% peak PFE from the nodes into various disk files and reassigns the nodes with new grid points. This process is repeated until the entire grid is covered. The contour generation process described above is used in the determination and verification of optimal grid spacing as described in the following sections.
D. Determination of Optimal Grid Spacing

The determination of optimal grid spacing involves the spatial frequency analysis of PFE contours as described in Chapter IV. The parallel implementation of this task follows the same methodology illustrated in Figure 4.27. First a low resolution PFE contour (100ft x 100ft) is generated on a predetermined grid. The grid used for this purpose can be decomposed using any of the three decomposition schemes as illustrated earlier. The low resolution contours generated are inspected by the user to determine the location and size of the local window (Figure 4.16). Once the local window is determined high resolution contours, \( f(m',n') \), are generated in this local window. The generation of the high resolution contour, however, utilizes the horizontal decomposition scheme in order to avoid interprocessor communication and data rearrangement for the next step in the optimal grid spacing determination process; i.e., the determination of the 2-D Fourier spectrum of the high resolution contour.

The 2-D Fourier spectrum of the high resolution contour data required for this analysis is obtained using a row-column decomposition based parallel implementation of the 2-D discrete Fourier transform (DFT). To reduce the computation time, the 2-D DFT of the 95% PFE contours is realized using a 2-D fast Fourier transform (FFT) algorithm [26,29,33]. The 2-D FFT of the high resolution PFE contours, \( f(m',n') \), generated in a \( M'xN' \) local window, is decomposed into two 1-D FFTs in \( m' \)
and \(n'\) directions [36]. The \(M'\)-point 1-D FFT of \(f(m',n')\) in the \(m'\)-direction is given by

\[
F(k', n') = F_1(k', n') + \bar{W}_{M'}^{k'} F_2(k', n') \\
\text{for } 0 \leq k' \leq (M'/2) - 1
\]

\[
F(k', n') = F_1(k' - M'/2, n') - \bar{W}_{M'}^{k'} F_2(k' - M'/2, n') \\
\text{for } M'/2 \leq k' \leq M' - 1
\]

where \(F_1(k', n')\) and \(F_2(k', n')\) are the \((M'/2)\)-point FFTs of the even and odd points in a particular row of \(f(m',n')\) and \(W_{M'} = e^{j2\pi M'}.\) Here it is assumed that \(M'\) is an integer power of 2. In the case that this requirement does not hold then we expand the data by means of zero padding at the rightmost end of the data array. The butterfly pattern for the 1-D FFT of 128 points is shown in Figure 6.8. A 128-point FFT is used here since \(M'\) and \(N'\) are both less than 100.

The row-wise transformation is followed by the \(N'\)-point 1-D FFT of \(F(k', n')\) in the \(n'\)-direction which is computed as

\[
F(k', 1') = F_1(k', 1') + \bar{W}_{N'}^{1'} F_2(k', 1') \\
\text{for } 0 \leq 1' \leq (N'/2) - 1
\]

\[
F(k', 1') = F_1(k', 1' - N'/2) - \bar{W}_{N'}^{1'} F_2(k', 1' - N'/2) \\
\text{for } N'/2 \leq 1' \leq N' - 1
\]
Flow graph for a 128-point FFT computation.

Figure 6.8
where $F_1(k',l')$ and $F_2(k',l')$ are the $(N'/2)$-point FFTs of the even and odd points in a particular column of $F(k',l')$ and $W_N = e^{\frac{j2\pi}{N'}}$. $N'$ is also assumed to be an integer power of 2; otherwise, zero padding is performed as mentioned above prior to the implementation.

The parallel computation of 2-D FFT involves three phases. In the first phase, each PE performs $(N'/16)$ 1-D FFT computations each on a $M'$-point array of real data. In the second phase, each PE broadcasts the row-wise transformed data, $F(k',n')$, which is received by all the nodes. The received data is then rearranged and transposed to obtain $[F(k',n')]^T$. By transposing the data, the column-wise transformation can be performed using the algorithm developed for the row-wise transformation. In the third phase, each node performs $(M'/16)$ 1-D FFT computations on an array of $N'$ complex numbers to yield the 2-D transform, $F(k',l')$. The determination of 2-D FFT is shown in Figure 6.9. Thus, the parallel implementation of 2-D FFT described above requires the following steps:

1. Generate the high resolution contour data such that each node generates the 95% peak PFE at $M'xN'/16$ grid points.

2. Determine the row-wise transformation of $f(m',n')$ in parallel (i.e., determine $N'$ number of independent $M'$-point 1-D FFT's).

3. Broadcast the resultant row-wise transformed data ($F(k',n')$).

4. Receive the row-wise transformed data.

5. Compute the column-wise transformation of $F(k',n')$ in parallel (i.e.,
Figure 6.9
Row and column decomposition based parallel implementation of 2-D DFT computation of PFE contours in a MxN spatial grid.
determine M' number of N'-point 1-D FFT's).

(6) Send FFT output \( F(k',l') \) to corner node.

The next step in the determination of optimal grid spacing involves low pass filtering of the resultant 2-D spectrum, \( F(k',l') \), of the high resolution PFE contours. Each node then performs the pointwise filtering operation given by equation 4.25 using the bandwidth of the filter specified by the user. The filtered output, \( F_{LPF}(k',l') \), is then reconstructed using 2-D inverse fast Fourier Transform (IFFT). The 2-D IFFT of \( F_{LPF}(k',l') \) is decomposed into two 1-D IFFTs. The M'-point 1-D IFFT of \( F_{LPF}(k',l') \) in the k'-direction is given by

\[
F_{REC}(m', l') = F_{REC1}(m', l') + W_{M'}^{m'} F_{REC2}(m', l') \quad \text{for } 0 \leq m' \leq (M'/2) - 1
\]

\[
F_{REC}(m', l') = F_{REC1}(m' - M'/2, l') - W_{M'}^{m'} F_{REC2}(m' - M'/2, l') \quad \text{for } M'/2 \leq m' \leq M' - 1
\]

where \( F_{REC1}(m',l') \) and \( F_{REC2}(m',l') \) are the \((M'/2)\)-point IFFTs of the even and odd points in a particular row of \( F_{LPF}(k',l') \) and \( W_{M'} = e^{i2\pi M'} \). This row-wise transformation is followed by the N'-point 1-D IFFT of F(m',l') in the l'-direction computed as
\[ f_{rec}(m', n') = f_{rec1}(m', n') + W_{N'}^n f_{rec2}(m', n') \]
\[
\text{for } 0 \leq n' \leq (N'/2) - 1
\]
\[ f_{rec}(m', n') = f_{rec1}(m', n' - N'/2) - W_{N'}^n f_{rec2}(m', n' - N'/2) \]
\[
\text{for } N'/2 \leq n' \leq N' - 1
\]

where \( f_{rec1}(m', n') \) and \( f_{rec2}(m', n') \) are the \((N'/2)\)-point IFFTs of the even and odd points in a particular column of \( F_{REC}(m', l') \) and \( W_{N'} = e^{2\pi i N'}. \)

The parallel computation of 2-D IFFT involves three phases. In the first phase, each PE performs \((N'/16)\) 1-D IFFT computations each on a \( M'\)-point array of data. In the second phase, each PE broadcasts the row-wise transformed data, \( F_{REC}(m', l') \), which is received by all the nodes. The received data is then rearranged and transposed to obtain \([F_{REC}(m', l')]^T\). In the third phase, each node performs \((M'/16)\) 1-D IFFT computations on an array of \( N' \) complex numbers to yield the reconstructed contour, \( f_{rec}(m', n') \). The reconstructed contour is then compared to the actual contour to determine the pointwise reconstruction error (Figure 4.20), \( E_r(m', n') \), and the deformation (Figures 4.24 - 4.26), \( E_s(m', n') \). The actual and reconstructed contour data is first distributed by means of block decomposition. Block decomposition provides a better performance for the window operations [37] used in the determination of optimal grid spacing. Thus, the \( M' \times N' \) actual and reconstructed PFE contours are decomposed into smaller blocks, each of size...
for 16-node implementation, prior to reconstruction error measurement.

The determination of pointwise reconstruction error, $E_{(m',n')}$, given by equation 4.27 involves point operations only. Hence, no data transfer is required for this purpose. The measurement of deformation, $E_d(m',n')$, in the constant PFE planes, involves thresholding, edge detection, gradient determination and distance measurement. The thresholding operation given by equations 4.29 and 4.30 is also a point operation and does not require additional data transfer. The edge detection operation given by equations 4.31 and 4.32 is defined in a 3x3 local region and hence requires data transfer from four connected neighbouring nodes (Figure 6.10). However, the determination of gradient using the Sobel templates of Figure 4.23 requires data from all of its eight connected nodes but also from the eight connected neighbouring nodes. The processing of corner points as illustrated in Figure 6.10 necessitates this data transfer. A data transfer technique similar to that developed in [37] is utilized here to transfer data required for the edge detection and gradient determination operations. In this approach, each node sends its top row (row 1) to the node to its north and then receives the top row (row 1) of the south node and assigns it to the expanded block of data as row $(M'/\sqrt{16} + 1)$. Then each node sends its bottom row (row $M'/\sqrt{16}$) to the node to its south. This is followed by receiving the bottom row of the north node obtaining row 0 of the expanded block of data. Thus, each node has a $(M'/\sqrt{16}+2) \times (N'/\sqrt{16})$ portion of the entire contour. The
Figure 6.10 Data transfer among neighboring processing elements of S-2010 in the parallel implementation of the shape deformation measurement between the actual and reconstructed PFE contours.
nodes then obtain the 0th column and \((N'/\sqrt{16}+1)\) column from their east and west neighbours. Thus, each node gets a block of size \((M'/\sqrt{16}+2)\times(N'/\sqrt{16}+2)\). Finally, each node then transmits the corner points to its eight connected nodes. This data transfer is actually a two step process (i.e., the data is transferred in 2 hops); however, due to the use of message passing this process is considered as a single step data transfer. This completes the data rearrangement required for the comparison of actual and reconstructed contours.

The contours are thresholded to obtain \(I(m',n')\) and \(I_{\text{rec}}(m',n')\) which are then processed to determine the gradient at each edge point on the actual and reconstructed contours. The shape deformation, \(E_{\text{r}}(m',n')\), is then determined by finding the point of intersection of the gradient with the edge of the binary reconstructed contour. To determine the point of intersection, each node sends the coordinates of the edge point on the actual contour, the computed gradient value and its logical node number to the nodes that lie within the local window specified by the components of the gradient by means of asynchronous communication. The nodes that lie in the window specified by the gradient then receive the coordinates of the edge points and the gradient, and determine the point of intersection. The point of intersection is determined by finding the closest point that has the same gradient in each subportion of the reconstructed contour. If a point of intersection is found then the deformation is calculated and sent to the node which requested the information. The node that originated the request selects the minimum distance as the
deformation at that point. This is illustrated in Figure 6.11. The gradient at point P is calculated by PE 5 which then sends the coordinates of this point and the direction of the gradient to the PE's 1, 2, 3, 6, 7, 8, 9, 12 and 13. The distance between the actual and reconstructed contour in the corresponding node as measured by each of the PE's is returned to PE 5. The distance, \( PP_i \), determined by PE 9 is minimum and is hence chosen as the deformation. Finally, the pointwise reconstruction error and the deformation are checked against specified tolerances. The least bandwidth satisfying the tolerances is determined by repeating the above reconstruction and error measurement process for smaller bandwidths of the 2-D low pass filter. The optimal grid spacing is then determined using equation 4.19.

E. Verification of Optimal Grid Spacing

Optimal grid spacing contour is generated using the grid spacing determined. The optimal grid spacing contour is then interpolated (equation 4.41) and compared with the high resolution contour to measure the deformation of PFE contours (Figure 4.29). The method used is similar to the comparison of high resolution and reconstructed contours. If the deformation is within tolerance limits then the optimal grid spacing is verified. Otherwise, the grid spacing is decreased and the comparison is conducted again. This is repeated until the verification process validates the grid spacing used.
Figure 6.11  Interprocessor communication along the direction of the gradient in the parallel implementation of the shape deformation measurement.
F. Performance analysis

The performance analysis of the parallel implementation described above is studied via timing equations and speedup factor. In the following analysis, we consider the determination of optimal grid spacing on S-2010 which involves the PFE contour generation and 2-D spectral analysis. The speedup factor, \( S \), defined earlier by equation 6.1 is given for this analysis as the ratio of time \( T_{\text{GRID}}(1) \) taken by a single node to complete the task of determining the optimal grid spacing to the time \( T_{\text{GRID}}(n) \) for the equivalent calculation performed on a \( n \)-node parallel processor:

\[
S = \frac{T_{\text{GRID}}(1)}{T_{\text{GRID}}(n)} \quad (6.6)
\]

In this particular implementation, the number of nodes \( n \) is 16 as specified by the system (S-2010) used for computer simulation.

The time for determination of optimal grid spacing on one node, \( T_{\text{GRID}}(1) \), can be expressed as

\[
T_{\text{GRID}}(1) = T_{\text{DOWNLOAD}}(1) + T_{\text{PROC}}(1) + T_{\text{UPLOAD}}(1) \quad (6.7)
\]
where \( T_{\text{DOWNLOAD}}(1) \) is the time taken to download the input (Propagation model Input Data or PID) files to one node of S-2010; \( T_{\text{PROC}}(1) \) is the processing time on one node for PFE contour generation, optimal grid spacing determination, error computation, and model verification; and \( T_{\text{UPLOAD}}(1) \) is the time taken to upload the low resolution PFE contours, the high resolution contours \( (f(m',n')) \), the reconstruction errors \( (E_x(m',n') \text{ and } E_y(m',n')) \), the reconstructed contours \( (f_{\text{rec}}(m',n')) \), and the optimal grid spacing contours \( (f_{\text{opt}}(m',n')) \). The download time is given by

\[
T_{\text{DOWNLOAD}}(1) = \left( \frac{L_1}{BW_1} + T_{\text{DISKOVER}} \right) \cdot (G + G' + G'')
\]  

(6.8)

where \( L_1 \) is the number of bytes in a PID, \( BW_1 \) is the host to node communication channel bandwidth in KBytes per second, \( T_{\text{DISKOVER}} \) is the overhead introduced due to disk access, \( G \) is the number of points in the high resolution grid given by

\[
G = \left( \frac{x_{\text{max}}'-x_{\text{min}}'}{x_h} + 1 \right) \cdot \left( \frac{y_{\text{max}}'-y_{\text{min}}'}{y_h} + 1 \right)
\]  

(6.9)

\( G' \) is the number of points in the optimal grid given by
and $G''$ is the number of points in the low resolution grid given by

$$
G'' = \left( \frac{X_{\text{max}} - X_{\text{min}}}{100} + 1 \right) \cdot \left( \frac{Y_{\text{max}} - Y_{\text{min}}}{100} + 1 \right) \tag{6.11}
$$

The processing time $T_{\text{PROC}}(1)$ can be expressed as

$$
T_{\text{PROC}}(1) = \left( t_{\text{MLS}} + t_{\text{DME/P}} + t_{\text{WINDOW}} \right) \cdot \left( G + G' + G'' \right) + T_{2D-\text{FFT}}(1) + \sum_{\text{# of iter}} T_{\text{REC}}(1) \tag{6.12}
$$

where $t_{\text{MLS}}$, $t_{\text{DME/P}}$, and $t_{\text{WINDOW}}$ are the execution times for the propagation and DME/P signal models and the 95% window routine for one grid point, respectively. $T_{2D-\text{FFT}}(1)$ is the execution time for the 2-D FFT of an $M' \times N'$ high resolution PFE contours. It should be noted that $G = M' \times N'$. The determination of 2-D FFT involves the computation of $N'$ number of $M'$-point FFT's and $M'$ number of $N'$-point FFT's. One $M'$-point FFT requires $M'.\log_2 M'$ number of complex additions and $M'.\log_2 M'$ complex multiplications. Each complex addition requires two addition operations on four 4byte numbers and each complex multiplication requires four 4byte multiplications and three additions. An $N'$-point FFT requires $N'.\log_2 N'$ complex
additions and $N' \cdot \log_2 N'$ complex multiplications. Therefore, the 2-D FFT is obtained from $N' \cdot M' \cdot (\log_2 M' + \log_2 N')$ complex additions and $N' \cdot M' \cdot (\log_2 M' + \log_2 N')$ complex multiplications. Hence,

$$T_{2D-FFT}(1) = (N' \cdot M' \cdot (\log_2 M' + \log_2 N')) \cdot (t_{add} + t_{mult}) + T_t \quad (6.13)$$

where $t_{add}$ and $t_{mult}$ are the time taken by a node of S-2010 to perform a complex addition and multiplication and $T_t$ is the time to transpose the row-wise transform, $F(k',n')$. The time to calculate the reconstruction error, $T_{REC}(1)$, depends on the time to calculate the 2-D IFFT and the reconstruction error. $T_{REC}(1)$ is given by

$$T_{REC}(1) = T_{2D-IFT}(1) + G \cdot t_{fil} + T_{ERR}(1) \quad (6.14)$$

Here, $t_{fil}$ is the time taken for filtering one point of the spectrum (i.e., a complex multiplication)

$$T_{2D-IFT} = (N' \cdot M' \cdot (\log_2 M' + \log_2 N')) \cdot (t_{add} + t_{mult}) + T_t \quad (6.15)$$

$$T_{ERR}(1) = G \cdot t_f + T_g(1) + T_{comp}(1) \quad (6.16)$$
where $t_r$ is the time required to calculate pointwise reconstruction error (i.e., time for a subtract operation), $E_r(m',n')$, at each point; $T_s(1)$ is the time required to calculate the shape deformation, $E_s(m',n')$, as a result of the reconstruction process; and $T_{\text{comp}}(1)$ is the time to compare the optimal and the high resolution contour. Here, $T_s(1)$ and $T_{\text{comp}}(1)$ are given by

$$
T_s(1) = 2G \cdot t_{th} + 2G \cdot t_{ed} + 2G \cdot t_{sobel} + G \cdot t_{\text{shape}} \quad (6.17)
$$

$$
T_{\text{comp}}(1) = G' \cdot t_{\text{interpol}} + G \cdot t_{th} + G \cdot t_{ed} + G \cdot t_{sobel} + G \cdot t_{\text{shape}} \quad (6.18)
$$

where $t_{th}$ is the time taken for one thresholding operation (i.e., comparison), $t_{ed}$ is time required to detect an edge, $t_{sobel}$ is the time required to determine the gradient at each point on the actual and reconstructed binary edge detected contour (i.e., time for 5 additions, 6 subtractions, 4 multiplications and 21 divisions), $t_{\text{shape}}$ is the time required to calculate the deformation at a point (i.e., time for 1 addition, 2 subtractions, 2 multiplications and 10 divisions), and $t_{\text{interpol}}$ is the time required for one interpolation operation (required to interpolate the optimal grid spacing contour to the high resolution grid during the comparison process). Each interpolation operation requires 3 additions, 4 subtraction, 2 multiplications and 10 divisions.

The error contour analysis operation results in 8 arrays of real numbers which include a low resolution contour of size $G''$, and 7 $M'xN'$ arrays. The seven arrays
include \(f(m',n')\), \(f_{\text{rec}}(m',n')\), \(E_i(m',n')\) and \(E_s(m',n')\), and interpolated \(f_{\text{opt}}(m',n')\), \(E_i(m',n')\) and \(E_s(m',n')\) obtained from the verification process. Each real number is represented by 4 bytes. Hence, we upload \((7 \cdot 4MN' + 4G'')\) bytes to the host. The upload time is then given by

\[
T_{\text{UPLOAD}}(1) = 7 \cdot \left( \frac{4M'N'}{BW_1} + T_{\text{DISKOVER}} \right) + 4 \cdot \frac{G''}{BW_1} + T_{\text{DISKOVER}} \quad (6.19)
\]

The parallel processing time on \(n\) nodes, \(T_{\text{GRID}}(n)\), depends on time for downloading \((T_{\text{DOWNLOAD}}(n))\), uploading \((T_{\text{UPLOAD}}(n))\), processing \((T_{\text{PROC}}(n))\), and interprocessor communication time \((T_{\text{IPCOMM}}(n))\). \(T_{\text{GRID}}(n)\) can be written as

\[
T_{\text{GRID}}(n) = T_{\text{DOWNLOAD}}(n) + T_{\text{PROC}}(n) + T_{\text{IPCOMM}}(n) + T_{\text{UPLOAD}}(n) \quad (6.20)
\]

\(T_{\text{DOWNLOAD}}(n)\) and \(T_{\text{UPLOAD}}(n)\) are given by

\[
T_{\text{DOWNLOAD}}(n) = \left\{ \frac{L_1}{BW_1} + \frac{L_1}{BW_2} + T_{\text{DISKOVER}} \right\} \cdot (G + G' + G'') \quad (6.21)
\]

\[
T_{\text{UPLOAD}}(n) = 7 \left( \frac{4M'N'}{BW_1} + \frac{4M'N'}{BW_2} + T_{\text{DISKOVER}} \right) + (\frac{4G''}{BW_1} + \frac{4G''}{BW_2} + T_{\text{DISKOVER}}) \quad (6.22)
\]
where \( BW_2 \) is the node-to-node communication channel bandwidth in Kbytes/sec, and \( L_H \) is the length of the packet header (Figure 6.12). In this case all the nodes send the low resolution contour, the high resolution contour, the reconstructed contour, the reconstruction errors \( (E_{r}(m',n') \) and \( E_{s}(m',n') \)), the optimal grid spacing contours and the errors, \( E_{r}(m',n') \) and \( E_{s}(m',n') \), obtained during the verification process to the corner node which then stores the data into corresponding disk files. The interprocessor communication time, \( T_{IPCOMM}(n) \), may be expressed as

\[
T_{IPCOMM}(n) = T_{SETUP}(n) + T_{BROADCAST}(n)
\]  

(6.23)

Here, \( T_{SETUP}(n) \) is the time required to compose data packets resulting from the determination of 2-D FFT, the gradient and distance measurements, and \( T_{BROADCAST}(n) \) is the time taken to broadcast the data within S-2010. These times are dependent on the amount of data transferred among the nodes. The total amount of data for the entire broadcast comprises of data transferred for the determination of 2-D FFT (i.e., \( 8M'N' \) bytes) and the data transferred during the error analysis. Before error analysis \( 4M'N'/16 + 2(M' + N')/16 + 4.4 \) bytes of each contour data is distributed and \( 2.4b_1.4 \) bytes of data is transferred to complete the deformation measurement. Here, we assume that a worst case number of edge points, \( b_1 \), are located in a single node. Similar data transfer is also required for the verification process. The equations below consider the time elapsed for the above mentioned
Figure 6.12

Data downloading and distributing PFE contour data points among the processing elements and uploading the results (low resolution contour, high resolution contour, reconstructed contour, optimal grid spacing contour, shape deformation and pointwise reconstruction error from determination and verification of optimal grid spacing) to the host.
data transfer operations. $T_{\text{SETUP}}(n)$ is given as

\[
T_{\text{SETUP}}(n) = \left( \text{Total number of data packets} \right) \cdot t_{\text{COMM OVERHEAD}} \\
= T_{\text{SETUP-2D-FFT}}(n) + \sum_{\# \text{ of iter}} T_{\text{SETUP-REC}}(n) \\
= T_{\text{SETUP-2D-FFT}}(n) + \sum_{\# \text{ of iter}} \left( T_{\text{SETUP-2D-IFFT}}(n) + T_{\text{SETUP-ERR}}(n) \right)
\]

(6.24)

where $t_{\text{COMM OVERHEAD}}$ is the operating system overhead for preparing a message, $T_{\text{SETUP-2D-FFT}}$ and $T_{\text{SETUP-2D-IFFT}}$ are the time taken to setup data packets during the 2-D FFT and inverse FFT computations. These are given as

\[
T_{\text{SETUP-2D-FFT}}(n) = \left( N', \left\lceil \frac{8 \cdot M'}{256} \right\rceil \right) \cdot t_{\text{COMM OVERHEAD}}
\]

(6.25)

\[
T_{\text{SETUP-2D-IFFT}}(n) = \left( N', \left\lceil \frac{8 \cdot M'}{256} \right\rceil \right) \cdot t_{\text{COMM OVERHEAD}}
\]

(6.26)

Here, $\lceil . \rceil$ represents the ceiling value. $T_{\text{SETUP-ERR}}(n)$ is the data packet setup time during the determination and verification of the optimal grid spacing. It is given by

\[
T_{\text{SETUP-ERR}}(n) = T_{\text{SETUP-SOBER}}(n) + T_{\text{SETUP-SHAPE}}(n) + T_{\text{SETUP-COMP}}(n)
\]

(6.27)
T_{\text{SETUP-ERR}}(n) \text{ is the sum of the data packet setup times for the gradient determination (T}_{\text{SETUP-SOBEL}}(n)), deformation measurement (T_{\text{SETUP-SHAPE}}(n)), and optimal grid spacing verification (T_{\text{SETUP-COMP}}(n)). These timings are given by

\begin{equation}
T_{\text{SETUP-SOBEL}}(n) = 2 \cdot \left( \frac{2}{\sqrt{16}} \cdot \frac{4}{256} \right) + \frac{M' \cdot N'}{16} \cdot \frac{4}{256} + 4 \cdot \frac{4}{256} \cdot t_{\text{COMM OVERHEAD}}
\end{equation}

(6.28)

\begin{equation}
T_{\text{SETUP-SHAPE}}(n) = 2 \cdot 4 \cdot b_1 \cdot \frac{4}{256} \cdot t_{\text{COMM OVERHEAD}}
\end{equation}

(6.29)

\begin{equation}
T_{\text{SETUP-COMP}}(n) = (\frac{2}{\sqrt{16}} \cdot \frac{4}{256} \cdot \frac{M' \cdot N'}{16} \cdot \frac{4}{256} + 4 \cdot \frac{4}{256}) + 2 \cdot 4 \cdot b_1 \cdot \frac{4}{256} \cdot t_{\text{COMM OVERHEAD}}
\end{equation}

(6.30)

The data broadcast time, T_{\text{BROADCAST}}(n), can then be given as

\begin{equation}
T_{\text{BROADCAST}}(n) = T_{\text{BROADCAST-2D-FFT}}(n) + \sum_{\text{of iter}} T_{\text{BROADCAST-REC}}(n)
= T_{\text{BROADCAST-2D-FFT}}(n) + \sum_{\text{of iter}} (T_{\text{BROADCAST-2D-IFFT}}(n) + T_{\text{BROADCAST-ERR}}(n))
\end{equation}

(6.31)
Here, \( T_{\text{broadcast-2D-FFT}}(n) \) and \( T_{\text{broadcast-2D-IFFT}}(n) \) are the time taken to broadcast 8\( M' \cdot N' \) bytes of data during the computation of the 2-D FFT and 2-D IFFT respectively. These are given by

\[
T_{\text{broadcast-2D-FFT}}(n) = N' \cdot \left[ \frac{8 \cdot M'}{256} \right] \cdot \frac{L_H}{B_{W_2}} + \frac{8 \cdot M' \cdot N'}{B_{W_2}} \tag{6.32}
\]

\[
T_{\text{broadcast-2D-IFFT}}(n) = N' \cdot \left[ \frac{8 \cdot M'}{256} \right] \cdot \frac{L_H}{B_{W_2}} + \frac{8 \cdot M' \cdot N'}{B_{W_2}} \tag{6.33}
\]

\( T_{\text{broadcast-err}}(n) \) is the data broadcast time during the determination and verification of the optimal grid spacing. It is given by

\[
T_{\text{broadcast-err}}(n) = T_{\text{broadcast-sobel}}(n) + T_{\text{broadcast-shape}}(n) + T_{\text{broadcast-comp}}(n) \tag{6.34}
\]

\( T_{\text{broadcast-err}}(n) \) is the sum of the data broadcast times for the gradient determination \( (T_{\text{broadcast-sobel}}(n)) \), deformation measurement \( (T_{\text{broadcast-shape}}(n)) \), and optimal grid spacing verification \( (T_{\text{broadcast-comp}}(n)) \). These are given as
\[ T_{BROADCAST-ERR}(n) = T_{BROADCAST-SOVEL}(n) + T_{BROADCAST-SHAPE}(n) + T_{BROADCAST-COMP}(n) \]

\[ T_{BROADCAST-SOVEL}(n) = \frac{2 \cdot L_H}{BW_2} \cdot \left( \left\lfloor \frac{2 \cdot (M' + N')}{\sqrt{16}} \right\rfloor \cdot \frac{4}{256} + \left\lfloor \frac{M' \cdot N'}{16} \cdot \frac{4}{256} \right\rfloor + 4 \cdot \frac{4}{256} \right) + \]
\[ \left( \frac{M' \cdot N'}{16} \cdot \frac{4}{256} \right) + \frac{2 \cdot (M' + N') \cdot 4}{\sqrt{16}} + 4 \cdot 4 \cdot \frac{2}{BW_2} \]

\[ (6.36) \]

\[ T_{BROADCAST-SHAPE}(n) = 2 \cdot \left( 4 \cdot b_1 \cdot \frac{4}{256} \right) + \frac{L_H}{BW_2} + \frac{2 \cdot 4 \cdot b_1 \cdot 4}{BW_2} \]

\[ (6.37) \]

\[ T_{BROADCAST-COMP}(n) = \left( \left\lfloor \frac{2 \cdot (M' + N')}{\sqrt{16}} \right\rfloor \cdot \frac{4}{256} + \left\lfloor \frac{M' \cdot N'}{16} \cdot \frac{4}{256} \right\rfloor + 4 \cdot \frac{4}{256} \right) + \]
\[ \left( \frac{M' \cdot N'}{16} \cdot \frac{4}{256} \right) + \frac{4 \cdot 4 + 4 \cdot b_1 \cdot 4}{BW_2} \]

\[ (6.38) \]

The processing time, \( T_{PROC}(n) \), for the determination of optimal grid spacing determination on \( n \) nodes is evaluated as

\[ T_{PROC}(n) = \left\lfloor \frac{G}{16} \right\rfloor + \left\lfloor \frac{G'}{16} \right\rfloor \cdot (t_{MLS} + t_{DMB/F} + t_{WINDOW}) + T_{2D-FFT}(n) + \sum_{\# \, of \, iter} T_{REC}(n) \]

where \( T_{2D-FFT}(n) \) is
Here $T_{\text{REC}}(n)$ and $T_{\text{ERR}}(n)$ are expressed as

$$T_{\text{REC}}(n) = T_{\text{2D-FFT}}(n) + \left[ \frac{G}{n} \right] \cdot t_{\text{fill}} + T_{\text{ERR}}(n)$$  \hspace{1cm} (6.41)$$

$$T_{\text{ERR}}(n) = \left[ \frac{G}{16} \right] \cdot t_{\text{f}} + T_s(n) + T_{\text{comp}}(n)$$  \hspace{1cm} (6.43)$$

where $T_s(n)$ is the time to determine the shape deformation, $E_s(m',n')$ on $n$ nodes, and $T_{\text{comp}}(n)$ is the time to compare the optimal grid spacing contour with the high resolution contour. These are given by

$$T_s(n) = 2 \cdot \left[ \frac{G}{16} \right] \cdot t_{\text{th}} + 2 \cdot \left[ \frac{G}{16} \right] \cdot t_{\text{ed}} + 2 \cdot \left[ \frac{G}{16} \right] \cdot t_{\text{shape}}$$  \hspace{1cm} (6.44)$$

$$T_{2D-\text{FFT}}(n) = (\lfloor \frac{N'}{16} \rfloor \cdot M' \cdot \log_2 M' + \lfloor \frac{M'}{16} \rfloor \cdot N' \cdot \log_2 N') \cdot (t_{\text{add}} + t_{\text{mult}}) + T_c$$  \hspace{1cm} (6.42)$$
Hence we derive the speedup factor for this implementation as

$$T_{\text{comp}}(n) = \left[ \frac{G}{16} \right] \cdot t_{\text{th}} + \left[ \frac{G}{16} \right] \cdot t_{\text{interpol}} + \left[ \frac{G'}{16} \right] + \left[ \frac{G}{16} \right] \cdot t_{\text{ed}} + \left[ \frac{G}{16} \right] \cdot t_{\text{sobel}} + \left[ \frac{G}{16} \right] \cdot t_{\text{shape}}$$

(6.45)

Hence we derive the speedup factor for this implementation as

$$S = \frac{T_{\text{DOWNLOAD}}(1) + T_{\text{UPLOAD}}(1) + T_{\text{PROC}}(1)}{(T_{\text{DOWNLOAD}}(n) + T_{\text{UPLOAD}}(n)) + T_{\text{PROC}}(n) + T_{\text{IPCOMH}}(n)}$$

(6.46)

To compute the speedup for optimal grid spacing determination we consider the system parameters as $L_1 = 8 \text{ Kbytes}, BW_1 = 64 \text{ Kbytes/sec}, L_H = 2 \text{ bytes}, BW_2 = 20 \text{ Mbytes/sec}, T_{\text{DISKOVER}} = 1 \text{ millisecond}$ and $t_{\text{COMM OVERHEAD}} = 168 \text{ microseconds}$. The execution time (single run) for the simulation package; i.e., $t_{\text{MLS}} + t_{\text{DMEM}} + t_{\text{WINDOW}}$, is taken to be 22.5 minutes (1350 seconds). The time to transpose the partial results of the 2-D FFT, $T_n$, is considered during data routing and is assumed to be 10 milliseconds. The execution time for various operations ($t_{\text{flt}}, t_{\text{sobel}}, t_{\text{shape}}, t_{\text{add}}, t_{\text{mult}}, t_{\text{div}}, t_{\text{ed}}, t_{\text{interpol}}$ and $t_{i}$) are determined by specifying the number of the assembly language instructions (e.g., addition, multiplication, subtraction, division) required for the execution of these operations in the node processor (68020) of S-2010 as given in TABLE I. The best case, cache case and worst case values are determined by multiplying the number of assembly language instructions by the number of clock cycles required for the execution of the assembly language instructions as given in [41]. TABLE II lists the number of clock cycles per instruction. The resultant values
### TABLE II - BREAKDOWN OF VARIOUS PROCESSES INTO BASIC OPERATIONS

<table>
<thead>
<tr>
<th>Execution Time</th>
<th>Basic Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addition</td>
</tr>
<tr>
<td>$t_{filt}$</td>
<td>3</td>
</tr>
<tr>
<td>$t_{add}$</td>
<td>2</td>
</tr>
<tr>
<td>$t_{mult}$</td>
<td>3</td>
</tr>
<tr>
<td>$t_{th}$</td>
<td>-</td>
</tr>
<tr>
<td>$t_{f}$</td>
<td>-</td>
</tr>
<tr>
<td>$t_{ed}$</td>
<td>3</td>
</tr>
<tr>
<td>$t_{sobel}$</td>
<td>5</td>
</tr>
<tr>
<td>$t_{interpol}$</td>
<td>3</td>
</tr>
<tr>
<td>$t_{shape}$</td>
<td>1</td>
</tr>
</tbody>
</table>
### TABLE III - CLOCK CYCLES REQUIRED FOR BASIC OPERATIONS

<table>
<thead>
<tr>
<th>Execution Time</th>
<th>Best Case (clock cycles)</th>
<th>Cache Case (clock cycles)</th>
<th>Worst Case (clock cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Subtraction</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Multiplication</td>
<td>41</td>
<td>43</td>
<td>44</td>
</tr>
<tr>
<td>Division</td>
<td>88</td>
<td>90</td>
<td>91</td>
</tr>
<tr>
<td>Comparison</td>
<td>16</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Exclusive-OR</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>
are listed in TABLE III. The best case timing is obtained when all the instructions are in the cache and a maximum overlap is obtained due to other instructions. The cache case timing is obtained when all the instructions are in the cache but there is no overlap. The worst case timing is obtained when the instruction is not in the cache. Figure 6.13 shows the speedup plots, \( S = S(G, n_{\text{constant}}) \), as a function of \( G \) for several values of \( n \) and Figure 6.14 illustrates the speedup curves, \( S = S(n, G_{\text{constant}}) \) versus \( n \) for various values of \( G \). In Figure 6.13, the speedup monotonically increases towards the maximum value (close to \( n \)) in the interval \( k_G n < G < (k_G + 1)n \), where \( k_G \) is an integer. The maximum value of speedup (\( n \)) is obtained when \( G = k_G n \) and then drops to a minimum value of \( \frac{(k_G n + 1)}{n} \) when \( G = k_G n + 1 \). This implies that there are more nodes available than the grid points (i.e., some of the nodes are idle). In Figure 6.14, the speedup for a given number of grid points increases towards the maximum when \( n < G \), reaches the maximum when \( n = G \) and remains constant at \( G \) for \( n > G \). Increasing the number of nodes does not improve the performance any further and the efficiency of the system drops once again. The speedup loss in the above analysis is attributed to the communications between the host and the nodes and the nodes themselves.
<table>
<thead>
<tr>
<th>Execution Time</th>
<th>Best Case (seconds)</th>
<th>Cache Case (seconds)</th>
<th>Worst Case (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{\text{filt}} )</td>
<td>6.92x10^{-6}</td>
<td>7.36x10^{-6}</td>
<td>7.76x10^{-6}</td>
</tr>
<tr>
<td>( t_{\text{add}} )</td>
<td>2.4x10^{-7}</td>
<td>3.2x10^{-7}</td>
<td>4.8x10^{-7}</td>
</tr>
<tr>
<td>( t_{\text{mult}} )</td>
<td>6.92x10^{-6}</td>
<td>7.36x10^{-6}</td>
<td>7.76x10^{-6}</td>
</tr>
<tr>
<td>( t_{\text{th}} )</td>
<td>6.4x10^{-7}</td>
<td>7.2x10^{-7}</td>
<td>7.2x10^{-7}</td>
</tr>
<tr>
<td>( t_{f} )</td>
<td>1.2x10^{-7}</td>
<td>1.6x10^{-7}</td>
<td>2.4x10^{-7}</td>
</tr>
<tr>
<td>( t_{\text{ed}} )</td>
<td>2.48x10^{-6}</td>
<td>2.84x10^{-6}</td>
<td>3.44x10^{-6}</td>
</tr>
<tr>
<td>( t_{\text{sobel}} )</td>
<td>8.18x10^{-5}</td>
<td>8.424x10^{-5}</td>
<td>8.612x10^{-5}</td>
</tr>
<tr>
<td>( t_{\text{interpol}} )</td>
<td>2.1x10^{-5}</td>
<td>2.208x10^{-5}</td>
<td>2.304x10^{-5}</td>
</tr>
<tr>
<td>( t_{\text{shape}} )</td>
<td>3.884x10^{-5}</td>
<td>3.992x10^{-5}</td>
<td>4.064x10^{-5}</td>
</tr>
</tbody>
</table>
Figure 6.13  Speedup plots as a function of number of grid points $G$ and a constant number of nodes ($n_{\text{constant}}$) i.e., $S= S(n_{\text{constant}}, G)$. 
Figure 6.14  Speedup plots as a function of number of nodes (n) and a constant number of grid points (G_{constant}) i.e., S=S(n,G_{constant}).
Chapter VII

SIMULATION RESULTS

A. Determination of Optimal Grid Size

The methodology developed to determine the optimal grid size was coded in FORTRAN and implemented on a Sun workstation and IBM PC. The optimal grid size was obtained for different scenarios involving a simulated 3° centerline approach in the Final Approach region of the DME/P system. The approach starts at a distance of 5 nautical miles (nm) from threshold and proceeds down to a point 50ft above the runway threshold as illustrated in Figure 5.7. The final approach region is selected because the inaccuracies induced by multipath are determined to be the most significant in this region. Scenarios with different combinations of ground aircraft and runway lengths were used in the simulation to investigate the dependence of grid size on runway length and obstacle size. This approach to the problem provides an insight into the location of the areas in the airport environment that need to be examined for critical area determination. It also helps to understand the sensitivity of these areas for different runway lengths and obstacles. Runway lengths examined were 5000 feet, 9000 feet and 12000 feet. Boeing-747 and 727 aircraft were selected for the simulation since these are the largest in the wide-body and transport
class of aircraft, respectively. These aircraft would cause the worst case MLS errors [5]. Figure 2.5 shows the centerline based grid used for the computer simulations.

Grid sizes determined for the combination of runway lengths and aircraft types are summarized in Table V. The grid sizes show a significant dependence on the size of the obstacle. The areas are larger for the larger aircraft (B-747). This is mainly due to the restriction on specular height. For example, the height of the tail fin for a B-747 is almost twice as high as the tail fin of a B-727. It is also noted that size of the grid increases with runway length. The next step in this research was the demonstration of the method for the determination of optimal grid spacing for the resultant grid sizes.

B. Determination of Optimal Grid Spacing - Demonstration of the Method

Error contours were generated for two different orientations of a B-747 aircraft obstacle for a 9000 ft runway; parallel and perpendicular (with respect to the runway). The grid size determined above was used for the contour generation task. A low resolution grid (100ft x 100ft) was selected to generate contour maps using the data output of the MLS math model. These maps are shown in Appendix B. It was found that the model predicts errors (incorrectly) for the perpendicular orientation. Hence, the grid spacing was determined only for the parallel orientation of a B-747 obstacle.
### TABLE V - RESULTS OF GRID SIZE DETERMINATION

| Runway Length (ft) | Grid Area Determined (ft²) | Type of Aircraft |
|-------------------|---------------------------|-----------------
|                   | B-747                     | B-727           |
| 5000              | 5500 x 800                | 3300 x 800      |
| 9000              | 9500 x 1200               | 5500 x 600      |
| 12000             | 12500 x 1300              | 6700 x 1200     |
From the low resolution contour maps generated for a B-747 oriented parallel to the runway, the area of maximum rate of change of error was located in a local window of size 1200 ft x 600 ft. Error contours were generated using a high resolution grid (25 ft X 25 ft) in this local window. These high resolution contours were then analyzed to determine the optimal grid spacing.

The spatial frequency analysis of high resolution contours was conducted to obtain the reconstructed contours. The degradation of the contours was then measured along the PFE values, $E_i(m',n')/f(m',n')$, as the percentage error, and also in the constant PFE planes, $E_i(m',n')$, as the absolute contour deformation. Upper bounds on these error measurements were specified for $E_i(m',n')/f(m',n')$ and $E_i(m',n')$ as 20% to 45% and 100 ft, based on accuracy requirements for critical area study.

The high resolution contours were obtained for the decision heights 900ft, 800ft, 700ft, 600ft, 500ft, 400ft, 350ft, 300ft, 250ft, 200ft (CAT-I DH), 150ft, and 100ft (CAT-II DH). It is observed that the PFE contours for the 100ft decision height provide all the significant information that is obtained at other decision heights. This can be attributed to the determination of 95% peak PFE for each decision height (see Figure 5.7) in which the data analyzed for the 50ft decision height would include that portion of the flightpath used in the analysis of higher decision heights. Hence, the analysis of high resolution contours for 100ft decision height would provide a
conservative estimate of the grid spacing. Different contour levels (3ft, 4ft, 5ft and 6ft) were examined for the 900 ft decision height and the CAT-II decision height. Analyzing these two decision heights is expected to give the upper and lower bounds for the effective bandwidth. The optimal grid spacing was determined for the various combinations of decision heights and contour values by varying the bandwidth of the 2-D ideal low pass filter until the specified error criteria were satisfied. The grid spacings for the 3ft and 5ft contours lie between 40ft (900ft DH) and 70ft (CAT-II DH). The 6ft contour, however, requires much smaller grid spacings since its region of support is much smaller compared to the other contour levels. Similar values of grid spacings were obtained for the 4ft contour level also. Further analysis of the 4ft contour showed that the 100ft deformation occurs due to a single discontinuity in the contour. Such a discontinuity is not truly representative of the DME/P system. This is further attributed to assumptions in the Mathematical model. The deformation measured is below 100ft if the single continuity is not considered for the 4ft contour. It is noted that the analysis for the 900ft DH provides smaller values for the grid spacings compared to the analysis for CAT-II DH. The peak and mean values of $E_t(m',n')/f(m',n')$ and $E_r(m',n')$ for different bandwidths (grid spacings) are summarized in Table VI (for 900 ft DH) and Table VII (for CAT-II DH). Error distributions were obtained and plotted for different bandwidths of the 2-D ideal low pass filter. The distribution plots illustrate the effect of reducing the bandwidth on the reconstruction error. Reconstruction errors increase as the bandwidth is reduced. These plots are also given in the Appendix B.
TABLE VI - RESULTS OF ERROR ANALYSIS FOR 900 FT DH

<table>
<thead>
<tr>
<th>Contour Level (ft)</th>
<th>Grid Spacing (ft)</th>
<th>Reconstruction Error</th>
<th>Pointwise Error</th>
<th>Deformation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Max (%)</td>
<td>Mean (ft)</td>
<td>Max (ft)</td>
</tr>
<tr>
<td>3</td>
<td>47</td>
<td>0.459</td>
<td>3.249</td>
<td>88.4</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.466</td>
<td>3.362</td>
<td>103.1</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>0.357</td>
<td>2.313</td>
<td>90.15</td>
</tr>
<tr>
<td></td>
<td>25.8</td>
<td>0.364</td>
<td>2.369</td>
<td>111.8</td>
</tr>
<tr>
<td>5</td>
<td>42.1</td>
<td>0.437</td>
<td>3.529</td>
<td>55.9</td>
</tr>
<tr>
<td></td>
<td>44.45</td>
<td>0.447</td>
<td>3.649</td>
<td>111.8</td>
</tr>
<tr>
<td>6</td>
<td>25.806</td>
<td>0.343</td>
<td>3.087</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>26.67</td>
<td>0.349</td>
<td>3.169</td>
<td>103.1</td>
</tr>
<tr>
<td>Contour Level (ft)</td>
<td>Grid Spacing (ft)</td>
<td>Reconstruction Error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>----------------------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pointwise Error</td>
<td>Max</td>
<td>Mean (ft)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deformation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>0.414</td>
<td>2.119</td>
<td>79.05</td>
</tr>
<tr>
<td></td>
<td>53.35</td>
<td>0.433</td>
<td>2.178</td>
<td>103.1</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>0.234</td>
<td>1.421</td>
<td>90.15</td>
</tr>
<tr>
<td></td>
<td>26.67</td>
<td>0.243</td>
<td>1.542</td>
<td>111.8</td>
</tr>
<tr>
<td>5</td>
<td>61.5</td>
<td>0.43</td>
<td>3.286</td>
<td>75.0</td>
</tr>
<tr>
<td></td>
<td>66.67</td>
<td>0.445</td>
<td>3.5</td>
<td>111.8</td>
</tr>
<tr>
<td>6</td>
<td>30.77</td>
<td>0.267</td>
<td>2.166</td>
<td>81.83</td>
</tr>
<tr>
<td></td>
<td>32.0</td>
<td>0.284</td>
<td>2.254</td>
<td>147.9</td>
</tr>
</tbody>
</table>
The optimal grid spacings obtained were verified using contours generated with 50ft and 75ft grid spacings. The results obtained (TABLE VIII) indicate that the 50ft grid spacing is sufficient for the 3ft, 5ft and 6ft contour levels. However, for the 4ft contour it is seen that a much smaller grid spacing is required. This is attributed to the single discontinuity as explained above. Hence, the choice of a 50ft grid spacing would provide the accuracy required for the critical area determination, for the sample data used in this work.

C. Parallel Implementation of PFE Contour Analysis

The worst case theoretical timing measurements for the PFE contour analysis scheme presented in Chapter VI are summarized in TABLE IX. It is noted that the determination of gradient requires the maximum time. This is due to the number of operations required in the 3x3 window. Finally, the contour generation time (elapsed time) on S-2010 is compared with that on IBM-4381 mainframe as given in Table X. It must be noted that the implementation on the S-2010 message passing system is approximately 4 times faster than the IBM-4381 mainframe. It must be noted that the IBM-4381 is a multiuser system while the S-2010 is a dedicated system.
### TABLE VIII - VERIFICATION OF GRID SPACING FOR 900 FT DH

<table>
<thead>
<tr>
<th>Contour Level (ft)</th>
<th>Grid Spacing = 50 ft</th>
<th>Grid Spacing = 75 ft</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean E&lt;sub&gt;s&lt;/sub&gt;</td>
<td>Max E&lt;sub&gt;s&lt;/sub&gt;</td>
</tr>
<tr>
<td>3</td>
<td>25.1118</td>
<td>87.95</td>
</tr>
<tr>
<td>4</td>
<td>31.67</td>
<td>145.78</td>
</tr>
<tr>
<td>5</td>
<td>24.845</td>
<td>98.825</td>
</tr>
<tr>
<td>6</td>
<td>25.7</td>
<td>70.07</td>
</tr>
</tbody>
</table>

### TABLE IX - PROCESSING TIME SUMMARY OF THE IMPLEMENTATION IN THE S-2010

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contour generation (single run)</td>
<td>1350 (average)</td>
</tr>
<tr>
<td>Thresholding</td>
<td>7.2 x 10&lt;sup&gt;-7&lt;/sup&gt;</td>
</tr>
<tr>
<td>Edge detection</td>
<td>3.44 x 10&lt;sup&gt;6&lt;/sup&gt;</td>
</tr>
<tr>
<td>Gradient determination</td>
<td>8.612 x 10&lt;sup&gt;5&lt;/sup&gt;</td>
</tr>
<tr>
<td>Pointwise error determination</td>
<td>2.4 x 10&lt;sup&gt;-7&lt;/sup&gt;</td>
</tr>
<tr>
<td>Interpolation</td>
<td>2.305 x 10&lt;sup&gt;7&lt;/sup&gt;</td>
</tr>
<tr>
<td>Deformation measurement</td>
<td>4.064 x 10&lt;sup&gt;5&lt;/sup&gt;</td>
</tr>
</tbody>
</table>
TABLE X - COMPARISON OF CONTOUR GENERATION TIME ON S-2010 AND IBM MAINFRAME

<table>
<thead>
<tr>
<th></th>
<th>S-2010 16-node Message Passing System</th>
<th>IBM-4381 Mainframe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of grid points completed during one simulation run.</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>Time taken for one simulation run</td>
<td>1350 secs (average)</td>
<td>300 secs(average)</td>
</tr>
<tr>
<td>Time taken for one grid point</td>
<td>84.375 secs</td>
<td>300 secs</td>
</tr>
</tbody>
</table>
CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

Methodologies were presented for the determination of optimal grid size and optimal grid spacing. The method of optimal grid size determination developed here is based on the analysis of scalloping frequency, time delay and specular height. These parameters are dependent on the interrogator-obstacle-transponder geometry, velocity of the interrogator, and frequency and velocity of propagation of the signal. The grid size is determined for various combinations of runway length and aircraft types based on constraints on these three parameters. The grid size determined is then bound within a rectangular area which is then used for the simulation.

A method for optimal grid spacing determination was developed based on spatial frequency analysis of PFE contours. Although, a 1-D spatial frequency analysis of error contours was first proposed by Chamberlin [10] for the azimuth subsystem his method did not account for the case when the maximum frequency component occurs in the diagonal direction and hence could not be used for other subsystems of the MLS. Further, his method did not provide a method to determine the contour deformation due to the use of a larger grid spacing. In the method presented here, the shortcomings of the earlier method are overcome by using a 2-D spatial frequency analysis of PFE contours. Error criteria based on the pointwise
reconstruction error, $E_n$, and shape deformation, $E_s$, were also developed in this method. $E_t$ was determined by a pointwise comparison of the reconstructed and actual contours while $E_s$ was measured using a gradient based method. The measurement of $E_s$ along the gradient at each point on the boundary of the actual contour required the definition of the direction of measurement along the gradient.

The optimal grid spacing was determined for the data generated using the unvalidated simulation package described earlier. The receiver portion of the MLS Mathematical model could not be used due to problems associated with its incorrect handling of shadowing phenomena for DME/P. It was found that the time delay parameter in the model had been set to zero for the case of shadowing. This, however, does not reflect the actual operation and hence needs to be investigated. As a direct consequence the worst case scatterer orientation could not be determined. Validation activities continue for the Baseline Microwave Landing System (BMLS) receiver model. Upon validation the worst case aircraft orientation will be determined.

Further research is needed for the optimal determination of $(u_c, v_c)$ and $(x_{opt}, y_{opt})$ instead of the iterative method used here. The error criteria, $E_n$, on error contour degradation and, $E_s$, on contour deformation need to be combined to derive analytical expressions for the selection of 2-D effective bandwidth. It is also recommended that the method presented here should be tested using data generated
using the validated DME/P model or real data collected from different airport environments and scenarios. It is worth mentioning that the method presented here can be used to solve the problem of optimal 2-D function representation and sampling required for object recognition and reconstruction in the fields of image processing and computer vision.

A time efficient implementation of the DME/P contouring package was carried out on a 16 node message passing multicomputer system. Parallel implementation was achieved by dividing the grid points into 16 groups, each of which was assigned to a particular node of the S-2010 system. Each node then executed the same code for a different set of gridpoints. A parallel implementation of 2-D FFT was also presented. Speedup analysis showed the effectiveness of the implementation. The low speedup losses observed are attributed to the communications between the host and the nodes in the form of disk access and partially due to node to node data broadcasting. This limitation of the implementation can be overcome by eliminating intermediate files between the various stages of the parallel implementation. This can be achieved by mapping the intermediate files into internal data arrays as illustrated in [22]. Accessing these intermediate files (Transmitter output file and Receiver output file) causes bus contention on the system which results in loss of speedup. Further performance improvement can be achieved by providing direct disk access to the nodes. The extensive interprocessor communication required for the 2-D FFT implementation also results in speedup loss. A parallel implementation of
the 2-D FFT with no interprocessor communication as described in [39] would be an alternative to overcome this problem. In this algorithm the data is first processed using the Discrete Radon Transform followed by 1-D FFT's without any interprocessor communication. The use of the Vector Floating Point Accelerator option will also improve the performance. Further a concurrent message passing implementation of the entire simulation package is recommended to improve the performance of the implementation.
REFERENCES


18. M. Dibenedetto, "DME/P Critical Areas: Proposed Error Budgets and Outline


APPENDIX A.

THE OVERALL SIMULATION SOFTWARE

In this section, an overview of the simulation software developed is provided. This includes a detailed description of software for grid size determination, contouring, grid spacing determination, and verification. Figures I and II provide an outline of the interaction between the various Job Control Language (JCL) and Fortran modules developed as a part of the project.

In Figure I the steps required to determine the optimal grid spacing are outlined. Figure II illustrates the contour generation package. Grid spacing determination starts with the determination of grid size using the DMEGRID routine (Appendix A.1). The grid size determined and other input parameters are included in the main contouring exec (SPLMS EXEC). Then the transmitter input file is setup as desired for a particular scenario. Once these steps are completed, contour generation can be started in the disconnect mode. Contour generation starts by calling SPLMS EXEC (Appendix A.2.1). This routine prepares the transmitter input file for the Baseline Microwave Landing System Transmitter (BMLST) model. SPLMS then calls TRANS2 EXEC (Appendix A.2.2) to run BMLST. Once TRANS2 completes its task, NNDM EXEC (Appendix A.2.3) is run. This routine executes the DME/P receiver portion (NNDME) (Appendix A.2.7). The output of the transmitter model (BMLSRIN DATA C) serves as the input to the receiver routine. The output of the receiver is then processed by the 95% windowing routine (WINK2) (Appendix A.2.8) which is called by WINX EXEC (Appendix A.2.4). The entire process is repeated until the contour generation is completed. SPLMS EXEC ensures the repetition of the process until the entire grid is covered.

The contour data thus obtained is analyzed using the grid spacing determination routine (GSPACE) (Appendix A.3). The resultant grid spacing is used to generate an optimal grid spacing contour. The next step involves the verification of the optimal grid spacing using VERISP (Appendix A.4). This routine compares the optimal grid spacing contour to the high resolution contour and determines the deformation. This verification ensures that the accuracy requirements for critical area study are satisfied.
Determine scenario to run (aircraft type, runway length, orientation of aircraft, etc)

Determine Grid Size (DMEGRID - App. A.2)

Contouring Package (App. A.2)

Low resolution contour data

Determine local window

High resolution contour data

Determine optimal grid spacing (GSPACE - App. A.3)

Optimal grid spacings

Verify optimal grid spacing (VERIFY - App. A.4)

Figure I Flow chart for overall simulation software.
Figure II  Flow chart for contour generation.

Start

Rundisc (App.A.2.5)

SPLMS (App. A.2.1)

Modify Input File
Run Transmitter Model
Run Receiver Model
Run Window Routine
Write results to disk files

End of Grid ?

Y

Contour Data

N

Trans2 (App.A.2.2)

NNDM (App.A.2.3)

WINK2 (App.A.2.4)
APPENDIX A.1

GRID SIZE DETERMINATION ROUTINE - DMEGRID

DMEGRID is used to determine the optimal grid size for DME/P contour generation process. This program determines the scalloping frequency, time delay and specular height along a 3° centerline approach with the obstacle (B-747 or B-727) positioned at various points on the grid. Based on the above mentioned parameters the program decides if a point should be considered for critical area determination. The flowchart for this routine is given in Figure A.1.

The main routine opens the output files and requests the user to enter the location of the transponder, the grid limits and increment, the runway length, the length of flightpath, and the obstacle type. Given this scenario, the program then positions the obstacle chosen at a grid point and determines the parameters along each point on the flightpath. If the constraints on all three parameters are satisfied for at least one point on the flightpath, then the obstacle location is chosen to be a part of the grid and is written to the file called FAILED. Otherwise the point is written to the file called PASSED. The entire process is repeated for all points on the grid. The output file FAILED then contains all the points that are required for contour generation.

The DMEGRID routine has the following subroutines with different functions:

SUBROUTINE SPECSB: This subroutine calculates the specular height at each grid point.
Input: None
Output: Specular height

SUBROUTINE TIMESB: This subroutine calculates the time delay between the direct signal and the multipath signal. X-O-R reflected ray path is simulated in this routine.
Input: None
Output: Time delay

SUBROUTINE SCLPSB: This subroutine calculates the scalloping frequency along the
flightpath.

**Input**: None

**Output**: Scalloping frequency

**SUBROUTINE FLHTPATH:** This subroutine determines the next waypoint along the flightpath.

**Input**: None

**Output**: Modifies common PLANE (XPLANE, YPLANE, ZPLANE)

**SUBROUTINE FAIL:** This subroutine writes out the grid points that should be modeled to the output file FAILED.

**SUBROUTINE PASS:** This subroutine writes out the grid points which need not be modeled to the output file PASSED.
Figure A.1 Flow chart for grid size determination (DMEGRID).
DMEGRID
This program determines the grid size based on analysis of scalloping frequency, time delay and specular height for a specified interrogator-obstacle-transponder geometry.

c PROGRAM DMEGRID
C
C Program Name: DMEGRID
C
C
PROGRAM DME
C
C VARIABLE DECLARATIONS
REAL XSTART, YSTART, XEND, YEND, XGINC, YGINC
REAL XPLANE, YPLANE, ZPLANE, XTEMP, YTEMP, PI, DEG
REAL XDME, YDME, ZDME, TALFIN, TIMDLY, DH, FLTDIS
REAL SCFREQ, SPECHT, RUNCTN, FLTTEMP, DEGRAD
REAL XGRID, YGRID, ZGRID, C, PLNVEL, CNTDUM
C
CHARACTER*3 AIRCFT
C
INTEGER COUNTR
C
C VARIABLE EXPLANATIONS
C
XSTART is the X starting coordinate of the grid
C
YSTART is the Y starting coordinate of the grid
C
XEND is the X ending coordinate of the grid
C
YEND is the Y ending coordinate of the grid
C
XGINC is the X grid incrementing value
C
YGINC is the Y grid incrementing value
C
XPLANE is the X coordinate of the approaching aircraft
C
YPLANE is the Y coordinate of the approaching aircraft
C
ZPLANE is the Z coordinate of the approaching aircraft
C
XTEMP is a temporary variable used for calculating the
X grid increment
C
YTEMP is a temporary variable used for calculating the
Y grid increment
C
XDME is the X coordinate of the DME transmitter
C
YDME is the Y coordinate of the DME transmitter
C
ZDME is the Z coordinate of the DME transmitter
C
TALFIN is the tail fin height of the modeled aircraft
C
in feet
C
TIMDLY is the time delay value of the transmitted signal
C
in seconds
C
DH is the decision height in feet (at threshold)
C
FLTDIS is the distance of the flight path in nautical miles
FLTDIS is the distance of the flightpath in nautical miles
SCFREQ is the scalloping frequency of the transmitted signal
in radians per second
SPECHT is the height of the specular point in feet
RUNLEN is the length of the runway in feet
FLT_TMP is the distance of the flightpath in feet
AIRCFT is the model of aircraft being modeled (727 or 747)
PI is equal to 3.14159265359
DEG is the angle of the flightpath in degrees
c is the speed of light in feet per second
PLNVEL is the velocity of the approaching aircraft
in feet per second
DMEFRQ is the frequency of the DME transmitter in hertz

COMMON
COMMON /GRID/ XGRID,YGRID,ZGRID
COMMON /DMEE/ XDME,YDME,ZDME
COMMON /PLANE/ XPLANE,YPLANE,ZPLANE
COMMON /MISC/ PI,DEG,C,PLNVEL,DMEFRQ,DEGRAD

Constants
DH = 50.0
PI = 3.14159265359
DEG = 3.0
C = 9.83571056E+08
PLNVEL = 232.1
DMEFRQ = 1.088E+09

Open output files
OPEN(UNIT=1,FILE=\'FAILED\',STATUS=\'UNKNOWN\')
OPEN(UNIT=2,FILE=\'PASSED\',STATUS=\'UNKNOWN\')

BEGIN MAIN PROGRAM
WRITE(6,*)'Program Commencing...
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*) 'Please enter the X,Y,Z coordinates of the DME',
&'transmitter, separated by a comma (F7.2)...'
READ(5,*) XDME,YDME,ZDME
WRITE(6,*)
WRITE(6,*) 'Please enter the starting X and Y grid points',
&'separated by a comma (F7.2)...'
READ(5,*) XSTART, YSTART
WRITE(6,*)
WRITE(6,*) 'Please enter the ending X and Y grid points',
&'separated by a comma (F7.2)...'
READ(5,*) XEND,YEND
WRITE(6,*)
WRITE(6,*) 'Please enter the X and Y grid increment values', &' separated by a comma (F7.2)...
READ(5,*) XGINC,YGINC
WRITE(6,*)
WRITE(6,*) 'Please enter the distance of flightpath', &' in nautical miles (F7.2)...'
READ(5,*) FLTDIS
WRITE(6,*)
WRITE(6,*) 'Please enter the length of the runway', &' in feet (F7.2)...
READ(5,*) RUNLEN
WRITE(6,*)
WRITE(6,*) 'Please enter the aircraft type (727 or 747) you', &' wish to model...'
CONTINUE
READ(5,20) AIRCFT
FORMAT(F7.2)
FORMAT(A3)
FORMAT(2F7.2)
FORMAT(3F7.2)

Set up output file headings

WRITE(1,*) 'SCENARIO DATA' WRITE(1,*) 'DME TRANSMITTER COORDINATES... ',XDME,YDME,ZDME
WRITE(1,*) 'GRID SIZE ...',(XEND-XSTART),', BY ',(YEND-YSTART)
WRITE(1,*) 'FLIGHTPATH ...',FLTDIS,'NAUTICAL MILES'
WRITE(1,*) 'RUNWAY LENGTH ... ',RUNLEN,' FEET'
WRITE(1,*) 'AIRCRAFT TYPE ...',AIRCFT
WRITE(1,*) 'DECISION HEIGHT ...',DH,'FEET'
WRITE(1,*) 'PLANE VELOCITY ...',PLNVEL,'FEET/SEC'
WRITE(1,*)
WRITE(2,*) 'SCENARIO DATA' WRITE(2,*) 'DME TRANSMITTER COORDINATES... ',XDME,YDME,ZDME
WRITE(2,*) 'GRID SIZE ...',(XEND - XSTART),', BY ',(YEND - YSTART)
WRITE(2,*) 'FLIGHTPATH ...',FLTDIS,'NAUTICAL MILES'
WRITE(2,*) 'RUNWAY LENGTH ... ',RUNLEN,'FEET'
WRITE(2,*) 'AIRCRAFT TYPE ...',AIRCFT
WRITE(2,*) 'DECISION HEIGHT ...',DH,'FEET'
WRITE(2,*) 'PLANE VELOCITY ...',PLNVEL,'FEET/SEC'
WRITE(2,*)
WRITE(1,*) 'FAILED GRID PTS.' WRITE(1,*) ' X GRID PT. Y GRID PT. SPECHT(ft) ',
& ' TIMEDELAY(sec) SCALPFREQ.(rad/sec)'
WRITE(2,*) ' PASSED GRID PTS.' WRITE(2,*) ' X GRID PT. Y GRID PT. SPECHT(ft) ',
& 'TIMEDELAY(sec) SCALPFREQ.(rad/sec)'

Initialize grid points

XGRID = XSTART
YGRID = YSTART

Convert degree of flightpath to radians

DEGRAD = DEG * (PI / 180.0)

Determine tailfin height of grid plane

IF (AIRCFT .EQ. '747') THEN
  TALFIN = 63.0
ELSEIF (AIRCFT .EQ. '727') THEN
  TALFIN = 32.0
ELSE
  WRITE(6,*)
  WRITE(6,*) 'Please enter either 727 or 747...' GOTO 33
ENDIF

Begin Looping

Position starting point of approaching aircraft

CONTINUE

FLTTMP = FLTDIS * 6076.0
ZPLANE = SIN(DEGRAD) * FLTTMP + DH
XPLANE = COS(DEGRAD) * FLT@MP + RUNLEN
YPLANE = 0.00

Begin Calculation Looping

CONTINUE

Determine the specular point for the given grid point and approaching aircraft position.

CALL SPECSB(SPECHT)

Determine if specular point is below tail fin height of grid plane. If so, the timedelay is checked. Otherwise, the next grid point is checked.

IF (SPECHT .LE. TALFIN) THEN
  ZGRID = SPECHT
  CALL TIMESB(TIMDLY)
ELSE
If the time delay is less than zero seconds, the time delay is set to zero. This is due to round off error.

IF (TIMDLY .LT. -0.00000001) THEN
    TIMDLY = 0.0
END IF

If the time delay is less than 300 nanoseconds, the scalloping frequency is checked. Otherwise, the next grid point is checked.

IF (TIMDLY .LE. 300.OE-09) THEN
    CALL SCLPSB (SCFREQ)
ELSE
    GOTO 55
END IF

If the scalloping frequency is less than 10.0 rad/sec, the grid point is a candidate for contouring. Otherwise, the flightpath is continued. If flightpath at end, the grid point isn't a candidate for contouring.

IF (SCFREQ .LE. 10.0) THEN
    CALL FAIL (SPECHT,TIMDLY,SCFREQ)
    GOTO 88
ELSE
    GOTO 55
ENDIF

Increment the position of the approaching plane.

CONTINUE

IF (ZPLANE .LE. DH) THEN
    CALL PASS (SPECHT,TIMDLY,SCFREQ)
    GOTO 88
ELSE
    CALL FLHTPH
ENDIF

Check to see if approaching aircraft is at end of flightpath.

IF (ZPLANE .GE. DH) THEN
    GOTO 22
ELSE
    CALL PASS (SPECHT,TIMDLY,SCFREQ)
    GOTO 88
ENDIF

Increment X grid position
CONTINUE
XGTEMP= XGRID
XGRID=XGTEMP+XGINC

Check to see if X grid position is equal to last X grid position
IF (XGRID .LE. XEND) THEN
  GOTO 11
ELSE

Increment Y grid position
XGRID =XSTART
YGTEMP= YGRID
YGRID=YGTEMP+YGINC

CNDUM=(YEND-YGRID)/YGINC+1.0
COUNTR=INT(CNDUM)
WRITE(6,*)'Processing... Y loops to be completed=',COUNTR
ENDIF

Check to see if Y grid position is equal to last Y grid position
IF (YGRID.LE.YEND) THEN
  GOTO 11
ELSE

Program finished
WRITE(6,*)
WRITE(6,*)'Program completed...'
ENDIF

END

SUBROUTINES
A subroutine to calculate the distance from the DME transmitter to the grid point which has ZGRID equal to the DME phase center height(PCH).

SUBROUTINE SPECSB ( SPECHT )

VARIABLE DECLARATIONS
REAL SPECHT,LTH1 ,LTH2,LTH3 ,SPCTMP

VARIABLE EXPLANATIONS
LTH1 is the distance from grid point to the DME transmitter
LTH2 is the distance from the grid point to the plane
LTH3 is the distance from the DME transmitter to the
approaching aircraft

SPCTMP is the temporary specular point
COMON /GRID/ XGRID,YGRID,ZGRID
COMON /DMEE/ XDME,YDME,ZDME
COMON /PLANE/ XPLANE,YPLANE, ZPLANE

Calculate the distance from the DME transmitter to the grid point with the ZGRID point equal to the DME phase center height

\[ LTH1 = \sqrt{(XGRID-XDME)^2 + (YGRID-YDME)^2 + (ZDME-ZDME)^2} \]

Calculate the distance from the grid point to the approaching aircraft with the ZPLANE point equal to the DME phase center height.

\[ LTH2 = \sqrt{(XPLANE-XGRID)^2 + (YPLANE-YGRID)^2 + (ZPLANE-ZDME)^2} \]

Calculate the distance from the DME to the approaching aircraft with the ZPLANE point equal to the DME phase center height.

\[ LTH3=LTH1+LTH2 \]

Calculate the temporary specular point

\[ SPCTMP = LTH1 * (ZPLANE - ZDME) / (LTH3) \]

Add the phase center height to the temporary specular point height to get the true specular point height.

\[ SPECHT = SPCTMP + ZDME \]

END

C A subroutine to calculate the time delay between the direct signal and the multipath signal.
C

SUBROUTINE TIMESB ( TIMDLY )

VARIABLE DECLARATIONS
REAL TIMDLY,LTH4,LTH5,LTH6

VARIABLE EXPLANATIONS
LTH4 is the distance from the DME transmitter to the grid point
LTH5 is the distance from the grid point to the approaching aircraft
LTH6 is the distance from the DME transmitter to the approaching aircraft
COMMON /GRID/ XGRID,YGRID,ZGRID
COMMON /DMEE/ XDME,YDME,ZDME
COMMON /PLANE/ XPLANE,YPLANE,ZPLANE
COMMON /MISC/ PI,DEG,C,PLNVEL,DMEFRQ,DEGRAD

Calculate the distance from the DME transmitter to the grid point.

\[ LTH4 = \sqrt{(XGRID-XDME)^2 + (YGRID-YDME)^2 + (ZGRID-ZDME)^2} \]

Calculate the distance from the grid point to the approaching aircraft.

\[ LTH5 = \sqrt{(XPLANE-XGRID)^2 + (YPLANE-YGRID)^2 + (ZPLANE-ZGRID)^2} \]

Calculate the distance from the DME to the approaching aircraft.

\[ LTH6 = \sqrt{(XPLANE-XDME)^2 + (YPLANE-YDME)^2 + (ZPLANE-ZDME)^2} \]

Calculate the time delay (in seconds).

\[ TIMDLY = (LTH4 + LTH5 - LTH6) \times \frac{1}{C} \]

END

A subroutine to calculate the scalloping frequency of the signal.

SUBROUTINE SCLPSB ( SCFREQ )

VARIABLE DECLARATIONS
REAL SCFREQ,LTH7,LTH8,LAMBDA,INVANG,SCTEMP

VARIABLE EXPLANATIONS
LTH7 is the distance from the grid point to the approaching aircraft in the X-Y plane
LTH8 is the distance from the grid point to the approaching aircraft in the X plane only
LAMBDA is the wavelength of the DME transmitter signal
INVANG is the cosine of the angle between the grid approaching aircraft line and centerline
SCTEMP is the scalloping frequency of the transmitted signal in hertz

COMMON /GRID/ XGRID,YGRID,ZGRID
COMMON /PLANE/ XPLANE,YPLANE,ZPLANE
COMMON /MISC/ PI,DEG,C,PLNVEL,DMEFRQ,DEGRAD

Calculate the distance from the grid point to the approaching plane. This is done only in the X,Y plane.

\[ LTH7 = \sqrt{(XPLANE-XGRID)^2 + (YPLANE-YGRID)^2} \]
Calculate the distance from the point along centerline, which is adjacent from the grid point to the approaching aircraft. The Y position for both is zero.

\[ LTH8 = (XPLANE - XGRID) \]

Calculate the \(\cos\) of the angle between the grid-approaching plane line and centerline.

\[ INVANG = \frac{LTH8}{LTH7} \]

Calculate the wavelength of the DME signal.

\[ \text{LAMBDA} = \frac{C}{DMEFRQ} \]

Calculate the scalloping frequency.

\[ \text{SCTEMP} = (\text{PLNVEL} / \text{LAMBDA}) \times (1 - \text{INVANG}) \]
\[ \text{SCFREQ} = (2 \times \pi) \times \text{SCTEMP} \]

A subroutine to determine the next way point of the approaching aircraft.

```
SUBROUTINE FLHTPH

VARIABLE DECLARATIONS
REAL ZTEMP,ZPINC,XTEMP,XPINC,PLNINC

VARIABLE EXPLANATIONS
ZTEMP is temporary variable used to calculate the new z coordinate of the approaching aircraft
ZPINC is the increment of the Z coordinate of the approaching aircraft
XTEMP is a temporary variable used to calculate the new x coordinate of the approaching aircraft
XPINC is the increment of the X coordinate of the approaching aircraft
PLNINC is the increment along the flightpath in feet

COMMON /PLANE/ XPLANE,YPLANE,ZPLANE
COMMON /MISC/ PI,DEG,C,PLNVEL,DMEFRQ,DEGRAD

Calculate the increment along the flightpath, this corresponds to 5 samples per second.

\[ PLNINC = \frac{PLNVEL}{5.0} \]

Determine the Z position of the approaching aircraft.

\[ ZTEMP = ZPLANE \]```
ZPINC = PLNINC * SIN (DEGRAD)
ZPLANE = ZTEMP - ZPINC

Determine the X position of the approaching aircraft.

XTEMP = XPLANE
XPINC = PLNINC * COS (DEGRAD)
XPLANE = XTEMP - XPINC

The Y position of the approaching aircraft is zero because the flightpath is taken to be on centerline.

YPLANE = 0.00

END

C A subroutine to output the grid points which should be modeled for contouring purposes.

SUBROUTINE FAIL (SPECHT,TIMDLY,SCFREQ)

C VARIABLE DECLARATIONS

REAL SPECHT,TIMDLY,SCFREQ

COMMON/GRID/ XGRID,YGRID,ZGRID
COMMON/PLANE/ XPLANE,YPLANE,ZPLANE

C Output to File

C WRITE(1,44) XGRID,YGRID
WRITE(1,44) XGRID,YGRID,SPECHT,TIMDLY,SCFREQ
WRITE(1,45) XPLANE,YPLANE,ZPLANE

44 FORMAT(1X,F10.4,1X,F10.4,7X,F6.3,2X,E20.9,2X,F15.8)
45 FORMAT(1X,F15.7,1X,F15.8,1X,F15.8)

END

C A subroutine to output the grid points which do not need to be modeled for contouring purposes.

SUBROUTINE PASS (SPECHT,TIMDLY,SCFREQ)

C VARIABLE DECLARATIONS

REAL SPECHT,TIMDLY,SCFREQ

COMMON/GRID/ XGRID,YGRID,ZGRID
COMMON/PLANE/ XPLANE,YPLANE,ZPLANE

C Output to File
C
C
WRITE(2,66) XGRID,YGRID
WRITE(2,66) XGRID,YGRID,SPECHT,TIM DLY,SCFREQ
WRITE(2,76) XPLANE,YPLANE,ZPLANE

66 FORMAT(1X,F10.4,1X,F10.4,7X,F6.3,2X,E20.9,2X,F15.8)
76 FORMAT(1X,F15.7,1X,F15.8,1X,F15.8)

C END
APPENDIX A.2

DME/P CONTOURING PACKAGE

The DME/P contouring package is used to generate error contours as illustrated in Section V. This package consists of Job Control Language (JCL) routines which control the execution of the BMLS Transmitter model, the DME/P receiver model, and the window routine.

The JCL routines used are as follows:

(a) SPLMS Exec (Appendix A.2.1)
(b) TRANS2 Exec (Appendix A.2.2)
(c) NNDM Exec (Appendix A.2.3)
(d) WINX Exec (Appendix A.2.4)
(e) RUNDISC Exec (Appendix A.2.5)

In the following sections these Execs and contour generation (Appendix A.2.6) are explained in detail. Finally, the DME/P receiver model (Appendix A.2.7) and the 95% windowing routine (Appendix A.2.8) are also described.
A.2.1 SPLMS EXEC

This exec modifies the input file and runs the contouring package until the entire grid for contouring is covered.

This is the main controlling Exec in the contouring package. The entire routine can be divided into the following sections:

1. section to handle user interaction
2. section to modify input file
3. section to call other routines to execute BMLST, DME/P receiver and Window
4. control section to check if entire grid is done.

In Section 1 the user enters the grid size given by the variables XINIT, YINIT, XMAX and YMAX. XINIT and YINIT denote the origin of the grid while XMAX and YMAX denote the maximum x and y-coordinates of the grid. The coordinate system has its origin at the stop end of the runway, which may not necessarily equal XINIT and YINIT. Thus, the stop end of the runway has the coordinates (0,0). Further, the user also enters the following information in this section: type of the scattering aircraft (e.g., B-747, B-727, etc.) and orientation, the grid spacing (variable INC), the next point (XJS,YJS) to be used on the grid, and parameters to be input to the windowing routine. The description of these variables is given in detail in the corresponding program listing.

The user interaction section is followed by the section to modify sections 3 and 8 of the transmitter model input data file. First the data entered by the user is formatted using the FORMX exec. Then the data is concatenated to suit the input file format for the corresponding sections. Once this is done, the commands required to edit the file are stacked and the editor (XEDIT) is invoked. This completes the modification of the input data file.

After the modification of the input file, SPLMS calls the routines TRANS2, NNDM and
Figure A.2.1 Flow chart for SPLMS EXEC.
WINX in that order to run the transmitter model, the DME/P receiver model and the windowing routine. The control section of the routine monitors the coverage of the entire grid by moving the scatterer to the next point on the grid and repeating the same process until the entire grid is complete. The control section algorithm can be stated as

```
IF X.LT.XMAX THEN
    X = X + INC
ELSE
    X = XINIT
    Y = Y + INC
END IF
IF Y.GT.YMAX THEN
    stop program
ELSE
    reposition ground aircraft at current X and Y;
    run MLS model, store data
END IF
```

The flow chart for SPLMS exec is given in Figure A.2.1. The following system command is issued to run the contouring package:

```
SPLMS input_file_name input_file_type input_file_mode <return>
```
**SPLMS EXEC**

This exec modifies the input file and runs the contouring package until the entire grid is covered.

```
&TRACE
&IF .&1 = . &GOTO -INFO
&IF &1 = ? &GOTO -INFO
*THIS IS THE DONE FLAG USED TO CHECK IF THE ENTIRE GRID IS COMPLETE*
&DONE = FALSE
*
* USER INTERACTION AREA
* VARIOUS VARIABLES INCLUDE XPOS,YPOS ETC OF SCATTERING
* AIRCRAFT AND SHADOWING AIRCRAFT
*
*VERSION NUMBER*
&VERNUM = 1
*SCATTERER/SHADOWING AC NO*
&NO = 1
*INCREMENT FOR GRID SPACING*
&INC = 25
*VIRTUAL M/C TO RECEIVE FILES*
&RECVIRT = AEC35
*INITIAL XPOS*
&XINIT = -500
*INITIAL YPOS*
&YINIT = 200
*MAX VALUE OF X IN THE GRID*
&XMAX = 1000
*MAX VALUE OF Y IN THE GRID*
&YMAX = 800
*XJS AND YJS ARE THE VALUES OF X AND Y TO BE
*TO BE SET TO THE NEXT POINT ON THE GRID WHEN
*CONTOURING IS STOPPED AND RESUMED AFTER A ROUTINE
*CHECK. WHEN CONTOURING IS FIRST STARTED INITIATE
*THESE VALUES TO XINIT AND YINIT RESPECTIVELY.

&XJS = 175
&YJS = 775
*NO. OF POINTS REQUIRED TO MEET CMN SPECS FOR 10 SEC WINDOW
*THE 10 SEC WINDOW IS SPECIFIED FOR FA MODE. FOR IA MODE USE
*USE 40 SEC WINDOW. THE VALUES OF CTOL AND PTOL ARE 2 AND 10
*FOR FA MODE AND IA MODE RESPECTIVELY.
*NO. OF POINTS REQUIRED TO MEET CMN SPECS*
&CTOL = 02
*NO. OF POINTS REQUIRED TO MEET PFE SPECS*
&PTOL = 02
&PHIGHT = 41.0
&MPTOL = 02
```
*WIDTH OF THE WINDOW- NUMBER OF POINTS IN WINDOW
*50 FOR FA MODE AND 200 FOR IA MODE.*

&WID = 050

*ANGULAR ORIENTATION OF SCATTERING AC*
&PLANG = 0

*XPOS, YPOS AND ZPOS FOR SHADOWING AC*
&X = &XJS
&Y = &YJS
&Z = 0

*XINC = 0.5 * LENGTH OF AC * COS(PLANG)*
&XINC = 116

*YINC = 0.5 * LENGTH OF AC * SIN(PLANG)*
&YINC = 0

*AIRCRAFT TYPE '747' OR '727'*
&ACTYPE = 747

*SHADOWING AND SCATTERING AC TYPE*
&XAT = 01

END OF USER AREA

*ALT FOR SCATTERER*
&XALT = 0

*ANGLE FROM CENTER LINE*
&XANG = 0

*VELOCITY FOR SHADOWING AC*
&XVEL = 0

*GRCORR FOR SHADOWING AC*
&XGRCORR = 0

*LOOP FOR EDITING AND RUNNING MODEL*

**EXEC FORMX IS USED TO CONVERT INTEGERS TO FLOATING POINT**
EXEC FORMX &TAILX
&READ VARS &XTAIL
EXEC FORMX &TAILY
&READ VARS &YTAIL
EXEC FORMX &CKPTX
&READ VARS &XCKPT
EXEC FORMX &CKPTY
&READ VARS &YCKPT
EXEC FORMX &Z
&READ VARS &ZP
EXEC FORMX &XVEL
&READ VARS &VEL
EXEC FORMX &XANG
&READ VARS &ANG
EXEC FORMX &XALT
&READ VARS &ALT
EXEC FORMX &XGRCORR
&READ VARS &GRCORR
&TYPE THE GRCORR IS &GRCORR
EXEC FORMX &X
&READ VARS &X2
EXEC FORMX &Y
&READ VARS &Y2

******************************************************************************

&XNO = 01
&TYPE EDITING FILE &1 &2 &3
*TO DETERMINE THE ENTRY FOR SECTION 3 (SCATTERING ACFT) AND SECTION
*8 (SHADOWING ACFT). THESE MODIFICATIONS ARE MADE ON THE INPUT FILE
*TO MOVE THE ACFT TO THE NEXT GRID POINT.
&HT = &CONCAT OF &XNO &BLANK &XTAIL
&IT = &CONCAT OF &HT &BLANK &YTAIL
&JT = &CONCAT OF &IT &BLANK &XCKPT
&KT = &CONCAT OF &JT &BLANK &YCKPT
&LT = &CONCAT OF &KT &BLANK &ALT
&MT = &CONCAT OF &LT &BLANK &XAT
&NT = &CONCAT OF &MT &BLANK &GRCORR
&OT = &RIGHT OF &NT 67
&TYPE &OT
* THIS PART PREPARES THE PORTION TO BE EDITED--->SECTION 3
******************************************************************************

* THIS PART PREPARES THE PORTION TO BE EDITED--->SECTION8
*
&RC = &RIGHT OF &VEL 5
&SC = &RIGHT OF &ANG 5
&UC = &CONCAT OF &XNO &BLANK &XTAIL
&VC = &CONCAT OF &UC &BLANK &YTAIL
&WC = &CONCAT OF &VC &BLANK &ZP
&XC = &CONCAT OF &WC &BLANK &RC
&YC = &CONCAT OF &XC &BLANK &SC
&ZC = &CONCAT OF &YC &BLANK &XAT
&FC = &RIGHT OF &ZC 52
&TYPE &FC
&UB = &CONCAT OF &X2 &BLANK &Y2
&VB = &CONCAT OF &UB &BLANK &ZP
&WB = &RIGHT OF &VB 37
&TYPE &WB
&SECT = SECTION
&SEC3 = &CONCAT OF &SECT &BLANK &BLANK 3
&SEC8 = &CONCAT OF &SECT &BLANK &BLANK 8
*HERE WE STACK UP THE COMMANDS FOR XEDIT TO EDIT THE INPUT FILE
*WITHOUT USER INTERACTION. COMMANDS ARE INCLUDED TO LOOK FOR
*PARTicular SECTION. GO TO THE LINE TO BE MODIFIED. DELETE IT.
*PUT IN THE NEW LINE PREPARED FROM CONCAT-ING.
&STACK /&SEC3
&STACK D6
&STACK DEL
&STACK U
&STACK I
&STACK &OT
&STACK
&STACK
&STACK /&SEC8
&STACK D7
&STACK DEL 2
&STACK U
&STACK I
&STACK &FC
&STACK
&STACK
&STACK I
&STACK &WB
&STACK
&STACK
&STACK FILE
*CALLING XEDIT HERE &1 &2 &3 REPRESENT THE FILENAME FILETYPE FILEMODE
X &1 &2 &3
*CALL EXEC FOR BMLST
*TRANS2 EXEC SETS UP FILE DEFs FOR BMLS THE TRANSMITTER PORTION
EXEC TRANS2 &1 &2 &3
*CALL EXEC FOR DME/P RECEIVER MODULE
*NNDM EXEC SETS UP FILE DEFs AND RUNS NNDME (DME/P RECEIVER)
EXEC NNDM
*** STACK THE VALUES FOR WINDOW
* LIKE WINDOW HEIGHT ETC
****
&STACK &PTOL
&STACK &WID
&STACK &PHIGHT
&STACK &X2
&STACK &Y2
&STACK &PHIGHT
&STACK &MPTOL
*CALL EXEC FOR 95% WINDOW ROUTINE
*THIS CALLS WINK2 FORTRAN THE WINDOW ROUTINE. WINX CAN ALSO
*BE MODIFIED TO RUN WINK FORTRAN WHICH IS THE WINDOW ROUTINE
*WITH A LARGER RANGE OF ERROR VALUES BUT LOW RESOLUTION BETWEEN
*LEVELS. WINK2 CONCENTRATES ON THE REGION OF 0 TO 30 FT WITH
*HIGH RESOLUTION BETWEEN LEVELS. MODIFY WINX TO RUN WINK OR WINK2
*AS NECESSARY.
EXEC WINX
*
&BEGSTACK -ENDSTACK *
NOW WE MOVE TO THE NEXT GRID POINT
-ENDSTACK
&READ STRING &INP
&type &INP
*HERE THE GRID POINT IS CHECKED TO SEE IF IT GOES OUT OF THE GRID
*IF IT DOES NOT IT IS MOVED TO THE NEXT POINT. FIRST THE GRID MOVES
*ALONG A CONSTANT Y UNTIL X REACHES XMAX. WHEN X EXCEEDS XMAX, X IS
*RESET TO XINIT AND Y IS INCREMENTED TO THE NEXT Y VALUE. IF Y EXCEEDS
*YMAX THEN ENTIRE GRID IS COVERED AND THE DONE VARIABLE IS SET TO TRUE.
*ONCE DONE IS SET TO TRUE THE EXEC EXITS.

&IF &X LE &XMAX &X = &X + &INC
&IF &X GT &XMAX &X = &XINIT
&IF &X EQ &XINIT &Y = &Y + &INC
&IF &Y GT &YMAX &DONE = TRUE
&IF &DONE NE TRUE &GOTO -TOP
&TYPE END OF RUN
&EXIT

INFO

&TYPE TO DO CONTOURING USING THE BMLS MATH MODEL, DME/P RECEIVER MODEL
&TYPE WINDOWING ROUTINE. THE INPUT FILE TO BMLST IS MODIFIED TO INCREMENT
&TYPE SCATTERING AND SHADOWING ACFT POSITIONS OVER A GRID UNTIL THE ENTIRE
&TYPE GRID IS COVERED.

FIN
&EXIT
A.2.2 TRANS2 EXEC
This exec is used to run the BMLS Transmitter model.

This JCL routine (see Figure A.2.2) is called by the main contouring routine, SPLMS. This routine sets up the input and output file definitions for the transmitter routine. It also stacks up responses to queries that require user interaction. The transmitter model requests responses for the following questions:

(a) DO YOU WANT TO SEE A COPY OF THE INPUT DATA?
(b) PROCESS AZ DATA?
(c) PROCESS EL DATA?
(d) PROCESS DME/P UPLINK DATA?
(e) PROCESS DME/P DOWNLINK DATA?
(f) PERTURBATION SMOOTHING?

The user enters the responses in the TRANS2 EXEC prior to the execution of the package. Finally, the routine executes the transmitter model. The flowchart for this exec is given in Figure A.2.2. This exec is called by typing

TRANS2 input_file_name input_file_type input_file_mode <return>
Figure A.2.2 Flow chart for TRANS2 EXEC.
&TRACE
* CHECK FOR QUESTION
&IF . &1 EQ . ? &GOTO -HELP
** link to VS Fortran library
GLOBAL TXTLIB VLKMLIB VFORTLIB
** set up file defs
* INPUT DATA FILE TO PROPOGATION MODEL
FI 15 DISK &1 &2 &3
* SYSTEM/RECEIVER MODEL (BMLSR)
FI 8 DISK BMLSRIN DATA C(LRECL 132
* PROPOGATION MODEL (BMLST) GRAPHICAL OUTPUT DATA, INPUT TO BPLOTT
FI 14 DISK BMLST PLOT C (LRECL 132
* CREATE THE DATE/MONTH INPUT DATA FOR SINIT SUBROUTINE
Q TIME (FIFO
&READ VARS &DUM1 &DUM2 &TIME &DUM3 &DUM4 &DATE
&READ VARS &DUM1
&DAY = &SUBSTR OF &DATE 4 2
&MO = &SUBSTR OF &DATE 1 2
&YR = &SUBSTR OF &DATE 7 2
* STACK DATA FOR SINIT SUBROUTINE
&STACK &YR
&STACK &DAY
&STACK &MO
&STACK &TIME
* * STACK ANSWERS TO QUESTIONS FOR INPUT TO BMLST
* * DO YOU WANT TO SEE A COPY OF THE INPUT DATA?
&STACK Y
* PROCESS AZ DATA?
&STACK N
* PROCESS EL DATA?
&STACK N
* PROCESS DME/P UPLINK DATA?
&STACK Y
*PROCESS DME/P DOWNLINK DATA?
&STACK N
* PERTURBATION SMOOTHING?
&STACK N
*
FI 5 TERMINAL
FI 6 TERMINAL
FI 9 DISK MODINFO DATA C
LOAD STASH (START
LOAD BMLST (START CLEAR NOMAP
This exec sets up the file defs and input prompts to run the Transmitter portion of the MLS math model. It is used as TRANS2 FN FT FM, where FN FT FM refer to the input data file name, type and mode respectively.
A.2.3 NNDM EXEC
This exec runs the DME/P receiver model.

This routine (Figure A.2.3) sets up the input and output file definitions and runs the DME/P receiver routine. The input file for the DME/P receiver is defined as the output file of the transmitter model (i.e., BMSLRIN DATA C). The output file (RESULT DATA C) is the input to the windowing routine called by WINX. The output file contains the raw PFE along the flightpath for a particular position of the ground obstacle. This routine is called by typing

NNDM <return>
Figure A.2.3 Flow chart for NNDM EXEC.
NNDM EXEC

&TRACE
*CHECK FOR QUESTION
&IF ..&1 EQ .? &GOTO -HELP
*LINK TO VS FORTRAN LIBRARY
GLOBAL TXTLIB VLNKMLIB VFORTLIB
*
*SET UP FILE DEFS
* INPUT DATA FILE
FI 8 DISK BMLSRIN DATA C (LRECL 132
* OUTPUT FILE
FI 10 DISK RESULT DATA C
FI 5 TERMINAL
FI 6 TERMINAL
LOAD NNDME(START CLEAR NOMAP
&EXIT
&HELP
&BEGTYPE
THIS EXEC SETS UP FILE DEFS AND RUNS NNDME FORTRAN. NNDME IS THE DME/P RECEIVER ROUTINE. NNDME FORTRAN GIVES THE ERROR ALONG THE FLIGHT PATH.
INPUT TO NNDME COMES FROM BMLST. VARIABLES USED ARE MD RATIO, TIME DELAY AND PHASE INFORMATION.
&ENDTYPE
&EXIT
&EXIT
A.2.4 WINX EXEC
This exec runs the 95% windowing routine.

This routine first sets up the input and output file definitions for the windowing routine (WINK2). The input file to the windowing routine is the output of the DME/P receiver routine (RESULT DATA C) while the output of the windowing routine is written out to files for various decision heights. WINX also stacks up user responses for the windowing routine. These responses include the window width, window height, x and y position of scatterer and PFE tolerance as described in the listing. The windowing routine is then run and the output of the windowing routine is appended to contour data obtained from previous simulations. The final output is the contour data files for various decision heights. The flowchart for this routine is illustrated in Figure A.2.4. This routine is called by typing

WINX <return>
Figure A.2.4 Flow chart for WINX EXEC.
Winx Exec
This exec runs the 95% windowing routine.

&TRACE ON
*  WINX EXEC A
*THIS EXEC SETS UP FILE DEFS FOR 95% WINDOW ROUTINE
*  
*  AZIMUTH VERSION OF SPLMLS EXEC
*  AVIONICS ENGINEERING CENTER
*  OHIO UNIVERSITY
FI 5 TERMINAL
FI 6 TERMINAL
FI 31 DISK RESULT DATA C
FI 30 DISK P1K DATA C (RECFM V
FI 29 DISK P900 DATA C (RECFM V
FI 27 DISK P800 DATA C (RECFM V
FI 25 DISK P700 DATA C (RECFM V
FI 23 DISK P600 DATA C (RECFM V
FI 21 DISK P500 DATA C (RECFM V
FI 19 DISK P400 DATA C (RECFM V
FI 17 DISK P350 DATA C (RECFM V
FI 16 DISK P300 DATA C (RECFM V
FI 13 DISK P250 DATA C (RECFM V
FI 11 DISK P200 DATA C (RECFM V
FI 20 DISK P150 DATA C (RECFM V
FI 7 DISK P100 DATA C (RECFM V
FI 3 DISK P50 DATA C (RECFM V
FI 1 DISK P25 DATA C (RECFM V
*TYPE ENTER TOTAL NUMBER OF POINTS WHICH ARE ALLOWED TO BE LARGER THAN
*TYPE THE WINDOW HEIGHT PER WINDOW FOR CMN (I2 FORMAT)
&READ VARS &CTOL
*TYPE ENTER NUMBER OF POINTS FOR THE WINDOW WIDTH  (I3 FORMAT)
&READ VARS &WID
*TYPE ENTER CMN WINDOW HEIGHT (F7.5)
&READ VARS &CMNH
*TYPE ENTER X POSITION OF SCATTERER
&READ VARS &X
*TYPE ENTER Y POSITION OF SCATTERER
&READ VARS &Y
*TYPE ENTER PFE WINDOW HEIGHT (F7.5)
&READ VARS &PFEH
*TYPE ENTER TOTAL NUMBER OF POINTS WHICH ARE ALLOWED TO BE LARGER *THAN
THE WINDOW HEIGHT PER WINDOW FOR PFE (I2 FORMAT)
&READ VARS &PTOL
&STACK &CTOL
&STACK &WID
&STACK &CMNH
&STACK &X
&STACK &Y
&STACK &PFEH
&STACK &PTOL
LOAD WINK (START CLEAR NOMAP
COPY P1K DATA C PFE1K DATA A (APPEND
COPY P900 DATA C PFE900 DATA A (APPEND
COPY P800 DATA C PFE800 DATA A (APPEND
COPY P700 DATA C PFE700 DATA A (APPEND
COPY P600 DATA C PFE600 DATA A (APPEND
COPY P500 DATA C PFE500 DATA A (APPEND
COPY P400 DATA C PFE400 DATA A (APPEND
COPY P350 DATA C PFE350 DATA A (APPEND
COPY P300 DATA C PFE300 DATA A (APPEND
COPY P250 DATA C PFE250 DATA A (APPEND
COPY P200 DATA C PFE250 DATA A (APPEND
COPY P150 DATA C PFE150 DATA A (APPEND
COPY P100 DATA C PFE100 DATA A (APPEND
COPY P50 DATA C PFE50 DATA A (APPEND
COPY P25 DATA C PFE25 DATA A (APPEND
ERASE P1K DATA C
ERASE P900 DATA C
ERASE P800 DATA C
ERASE P700 DATA C
ERASE P600 DATA C
ERASE P500 DATA C
ERASE P400 DATA C
ERASE P350 DATA C
ERASE P300 DATA C
ERASE P250 DATA C
ERASE P200 DATA C
ERASE P150 DATA C
ERASE P100 DATA C
ERASE P50 DATA C
ERASE P25 DATA C
&EXIT
A.2.5 RUNDISC EXEC
This exec runs the contouring package in disconnect mode.

The contour generation process takes extensive time (usually more than a week) to complete. Therefore, it is inconvenient to run the contour generation in a normal session and tie up the terminal for the whole session. Hence the program is run in the disconnect mode. This way a virtual machine runs the program but a terminal is not used.

In this routine, the SET AUTOREAD OFF command tells the mainframe not to look for terminal input, while CP DISCONN puts the virtual machine in disconnect mode. EXEC SPLMS PREPIN FIF C starts the contouring program by specifying the input file (PREPIN FIF C) to be used by the contouring package. This exec is run by typing its name:

RUNDISC <return>
&TRACE ALL
*THIS PROGRAM RUNS SPLMS EXEC
*PROGRAM IN DISCONNECT
SET AUTOREAD OFF
CP DISCONN
EXEC SPLMS PREPIN FIF C
CP LOGOFF
A.2.6 CONTOUR GENERATION

In this section, the steps involved in setting up and running the contouring package are described. For a given scenario, several parameters are determined. These include subsystem (AZ, EL or DME/P) characteristics, flightpath and velocity of approaching aircraft, type of ground aircraft and orientation, grid size and spacing. The subsystem characteristics include the location of the system, the transmitter parameters, the type of processing desired and the antenna pattern. A typical scenario may be in the following form:

Subsystem type : DME/P
Subsystem location (x,y,z): -1000.0,0.0,9.0 (phase center height)
Uplink frequency : 1072.498 MHz
Downlink frequency : 1072.498 MHz
Pulse shape : cos/cos2
Type of processing : DAC
Flightpath : 3° Glide path, centerline approach, starting at
5nmi from threshold, ending at threshold,
threshold crossing height = 50 feet.
Runway length : 9000 feet
Ground aircraft type : B-747
Aircraft orientation : Parallel to runway
Grid limits : -500 feet < x < 9000 feet; 0 feet < y < 1500 feet
Grid Spacing : 100 feet

These parameters for the given subsystem are entered into the corresponding sections of the input file. The following command is issued to edit the input file:

XEDIT file_name file_type file_mode <return>
DME/P parameters are entered in section 1 D. The ground aircraft type and location are entered into sections 3 and 8 which are used to simulate scattering and shadowing aircraft respectively. These sections are selected as desired by setting the flag appropriately (Yes/No) in the beginning of the section. The flightpath to be simulated is entered into section 12 of the input file. After setting up the input file, the main contouring exec SPLMS is edited by typing

\texttt{XEDIT SPLMS EXEC A <\texttt{return}>}

The user section of SPLMS is edited to modify aircraft type and orientation. The grid limits (XINIT, YINIT, XMAX and YMAX) and increment (INC) are also entered. When contouring is started for the first time, the variables XJS and YJS are set to be equal to XINIT and YINIT. Various parameters to be stacked for the windowing routine are also entered in the user section of SPLMS. Contour data for the scenario is then produced as follows. The ground aircraft is positioned at (XINIT, YINIT) (for this scenario: -500.0,0.0). The contouring package is run and the 95% peak error is determined and stored along with the position of the ground aircraft. The aircraft is then positioned at the next point in the current row and the process is repeated. Once the end of the row is reached, the next grid point is located at the start of the next row. This is determined by the following algorithm: For constant Y,

\texttt{IF X.LT.XMAX THEN}
\texttt{\hspace{1cm} X = X + INC}
\texttt{ELSE}
\texttt{\hspace{1cm} X = XINIT}
\texttt{\hspace{1cm} Y = Y + INC}
\texttt{END IF}
\texttt{IF Y.GT.YMAX THEN}
\texttt{\hspace{1cm} stop program}
\texttt{ELSE}
\texttt{\hspace{1cm} reposition ground aircraft at current X and Y;}
\texttt{\hspace{1cm} run MLS model, store data}
\texttt{END IF}

Once the above mentioned modifications are made the contouring package is run by typing:

\texttt{SPLMS file_name file_type file_mode <\texttt{return}>}

The contouring package can be run in disconnect mode by typing.
The contouring process needs to be checked periodically. This ensures that the process has not crashed or terminated. First logon to the virtual machine and when the display says reconnected type in the following:

\begin{verbatim}
ATTN <return>
HX <return>
SET AUTOREAD ON <return>
\end{verbatim}

The user may then check the data and reset the aircraft position to the next grid point by modifying the XJS and YJS variables. When ready to restart the program, the following command is typed:

\begin{verbatim}
RUNDISC <return>
\end{verbatim}
A.2.7 DME/P RECEIVER MODEL
This routine determines the raw PFE at each flightpath point.

This is the modified version of the DME/P signal model and is used as the DME/P receiver. This program reads the Multipath-to-Direct (M/D) ratio, phase and time delay of each signal component (both direct and multipath) at a point on the flightpath to determine the raw path following error (PFE). The output of the transmitter is used as the input to this program. The final output of this routine consists of the coordinates of each point on the flightpath and the error determined at that point.

The main routine reads the output of the transmitter model. First it skips the top portion of the input file until it reaches the section containing the multipath data. The program reads the flightpath coordinates and the number of signal components at that point. The M/D ratio, phase and time delay of each signal component are read and the combined perturbed pulse is generated. The combined pulse is then filtered using an IF filter to smoothen the envelope of the pulse. The filtered pulse is then processed using the delay-attenuate-compare (DAC) routine to obtain the DAC time. The error is then obtained from a measure of this time. The PFE value calculated is then written to the output file. The flowchart of this routine is given in Figure A.2.5. The program written in Fortran is also given.

The various subroutines called in this program and their functions are given below:

SUBROUTINE WINPUL: This subroutine is used to generate the combined perturbed pulse. The direct pulse generated is a cos/cos2 pulse.

Input: Pulse duration (pdur), rise time (rtime), multipath ratio (ampmp), time delay (delymp), phase (phase).

Return value: Combined pulse (sigtot)
SUBROUTINE FILT: This subroutine is used to filter the envelope of the perturbed pulse. This is a 1.75 MHz, 5-pole Butterworth IF filter.

Input : Amplitude of pulse to be filtered (sigamp) and number of points.

Return value: Filtered amplitude (sigamf)

SUBROUTINE DAC: This subroutine is used to determine the DAC detect time. Here the attenuated pulse and the delayed pulse are compared to determine the DAC detect time.

Input : Filtered pulse(sigamp), DAC delay time (delay), DAC attenuation constant (ATT) and number of points.

Return value: Flag to show if no decode or decode, and DAC detect time.

The following parameters need to be checked in the code before running the program:

(1) DME/P Frequency

(2) Pulse rise time
Figure A.2.5 Flow chart for DME/P receiver model.
NNDME.FOR
This program determines the raw PFE along a given flightpath.

PROGRAM NNDME
C PROGRAM TO DETERMINE DME/P ERROR OVER A RANGE OF M/D RATIOS AND
C TIME DELAYS FOR A GIVEN RELATIVE PHASE OF A MULTIPATH PULSE.
C THIS IS DONE FOR THE ENTIRE FLIGHTPATH. THE OUTPUT CONSISTS
C OF FLIGHTPATH COORDINATES AND THE ERROR.
C
C THIS IS A MODIFIED VERSION OF DME/P SIGNAL MODEL WRITTEN BY
C MICHAEL S. BRAASCH MARCH 1991. MODIFICATIONS MADE BY J. RAJENDRAN
C IN JULY 1991. MODIFICATIONS INVOLVED INTERFACING OUTPUT OF
C BMLST WITH THE SIGNAL MODEL.
C
C LAST MODIFIED IN APRIL 1992 TO INCLUDE THE PHASE CORRECTION.
C PRIOR TO MODIFICATION THE PHASE WAS DIRECTLY READ FROM INPUT FILE.
C BUT NOW THE PHASE IS THE SUM OF PHASE DUE TO TIME DELAY PLUS
C THE PHASE IN THE INPUT FILE.
C
C EXTERNAL DAC,WINPUL,FILT
C
C COMPLEX SIGTOT(8000),SIGDES(8000)
REAL SIGTIM(8000),SIGAMF(8000)
REAL*8 SIGAMP(8000)
REAL MPDB,FREQ
REAL RCVR(3),RVEL(3),AMP(30),TDEL(30),PH(30)
CHARACTER*132 BUFFER
LOGICAL STOP
INTEGER IANT(5)

C
C PI = ACOS(-1.0)
C = 9.8357120E8
C
C CHECK FREQUENCY BEFORE RUNNING RECEIVER
C OPERATIONAL FREQUENCY OF DME/P RECEIVER IN MHz
FREQ = 1072.498
C
C SET PULSE DURATION AND PULSE RISE TIME (NANOSECONDS)
PDIFF = 5800.
RTIME = 800.
C
C SKIP TOP SECTION OF INPUT FILE TILL MULTIPATH DATA IS REACHED
100 IF (BUFFER(1:8).NE.'END DATA') THEN
READ(8,10)BUFFER
10 FORMAT(A132)
GO TO 100
END IF
C READ MULTIPATH DATA FOR EACH FLIGHTPATH POINT
READ(8,*)((IANT(I),I=1,5)

73 READ(8,7)IPSS,INDEX,MTR,STOP,DIST,RCVR(1),RCVR(2),RCVR(3),
> (RVEL(I),IA = 1,3,NCOMP)
READ(8,9)(AMP(IX),PH(IX),AZ,EL,TDEL(IX),TDOP,
> RDOP,AZIN,ELIN,IX=1,NCOMP)
7 FORMAT(1X,I5,I5,I2,3X,L1,7(F10.3),I3)
9 FORMAT(1X,2E15.8,2E14.7,3E15.8,2E14.7)
C ZERO OUT THE DESIRED SIGNAL DATA VECTOR
   DO 150 I = 1,8000
       SIGDES(I) = (0.,0.)
   CONTINUE
C READ THE DESIRED SIGNAL INTO THE TOTAL SIGNAL VECTOR
   DO 163 I = 1,8000
       SIGTOT(I) = SIGDES(I)
   CONTINUE
C GENERATE THE PERTURBED PULSE. COMPONENT 1 IS THE DIRECT SIGNAL
C HERE WE COMBINE THE DIRECT AND MULTIPATH TO OBTAIN THE PERTURBED
PULSE
   DO 63 ID = 1,NCOMP
   C AMPLITUDE
       AMPMP = AMP(ID)
   C TIME DELAY
       DELYMP = TDEL(ID)/1E-9
   C PHASE - THIS WAS THE LAST MODIFIED PART. PREVIOUSLY IT WAS PHASE=PH(ID)
       PHASE = PH(ID)-(TDEL(ID)*2*PI*FREQ*1E06)
   C WINPUL IS CALLED TO GET THE COMBINED PULSE IN SIGTOT ARRAY
       CALL WINPUL(PDUR,RTIME,AMPMP,6500.+DELYMP,PHASE,SIGTOT)
   CONTINUE
C DO 200 J = 1,8000
   SIGTIM(J) = FLOAT(J)
   SIGAMP(J) = DBLE(CABS(SIGTOT(J))
200 CONTINUE
C FILTER SIGNAL AMPLITUDE ARRAY
   CALL FILT(SIGAMP,SIGAMF,8000)
C PERFORM DAC CALCULATIONS
   CALL DAC(SIGAMF,SIGTIM,8000,100.,0.5,YNDAC,TIMDAC)
   ERROR = (TIMDAC - 7006.)*(1.0E-9)*C
   IF(YNDAC.LT.(1.)) WRITE(*,*)'NO DECODE!'
       WRITE(10,51) RCVR(1),RCVR(2),RCVR(3),DIST,ERROR,STOP
C 51 FORMAT(5(1X,F11.4),1X,L4)
   IF (.NOT.(STOP)) GO TO 73
C CLOSE(10)
STOP
END
SUBROUTINE WINPUL(PULDUR,RISTN,AMP,PSTART,PHI,S)
C ROUTINE TO GENERATE A WINDOW OF SAMPLED DATA POINTS FOR A
C GIVEN SET OF PULSE CHARACTERISTICS
C
REAL AMP,PSTART,PULDUR,TINC,RISTN,OMEGM1,OMEGM2,PHI
COMPLEX S(8000)
C
IN-ONLY PARAMETER DESCRIPTIONS:
C AMP: PULSE PEAK AMPLITUDE
C PSTART: PULSE START TIME RELATIVE TO THE BEGINNING OF THE
C WINDOW, IN NANOSECONDS
C PULDUR: PULSE DURATION IN NANOSECONDS
C TINC: TIME SAMPLE INCREMENT IN NANOSECONDS
C RISTN: ZERO LEVEL TO PEAK PULSE RISE TIME IN NANOSECONDS
C OMEGM1: ANGULAR FREQUENCY OF THE COS PART OF THE PULSE
C OMEGM2: ANGULAR FREQUENCY OF THE COS2 PART OF THE PULSE
C PHI: PHASE ANGLE IN RADIANS (0 FOR THE DESIRED PULSE)
C
OUT-ONLY PARAMETER DESCRIPTION:
C S: VECTOR CONTAINING THE SUM OF THE PREVIOUS PLUS THE NEW
C SAMPLED PULSE DATA POINTS
C
DATA TINC /1.0E-9/
DATA OMEGM1 /1.0472E06/
DATA OMEGM2 /0.31416E06/
C
COSST = PSTART
COSSP = PSTART + RISTN
COS2ST = COSSP + TINC
COS2SP = PSTART + PULDUR
SIGST = PSTART
C
SKIP = 0.
IF(COSSP.GT.(8000.))THEN
  COSSP = 8000.
  SKIP = 1.
ELSEIF(COS2SP.GT.(8000.))THEN
  COS2SP = 8000.
ENDIF
C
RISET = RISTN/1.0E09
N = INT(SIGST)
PSTS = PSTART/1.0E09
C
SAMPLE AND STORE THE 'COS' PART OF THE PULSE
DO 50 T = COSST,COSSP,TINC
  TIME = T/1.0E09
  AK = AMP*SN(OMEGM1*(TIME-PSTS))
  S(N) = S(N) + CMPLX(COS(PHI)*AK,SIN(PHI)*AK)
  N = N + 1
50 CONTINUE
C
IF(SKIP.LT.(1.))THEN
C SAMPLE AND STORE THE "COS2" PART OF THE PULSE
DO 60 T = COS2ST,COS2SP,TINC
   TIME = T/1.0E09
   AK = AMP*(((COS(OMEGM2*(TIME-(PSTS+RISET))))**2)
   S(N) = S(N) + CMPLX(COS(PHI)*AK,SIN(PHI)*AK)
   N = N + 1
60 CONTINUE
ENDIF
N = N - 1
C
RETURN
END

SUBROUTINE DAC(SIGVID,VIDTIM,NPTS,DELAY,ATT,DACYN,DACTIM)
C ROUTINE TO DETERMINE THE DAC DETECT TIME
C
REAL SIGVID(8000),VIDTIM(8000)
REAL DELAY,ATT,DACYN,DACTIM
INTEGER NPTS
C
C DESCRIPTION OF IN-ONLY PARAMETERS:
C SIGVID: VECTOR CONTAINING THE ENVELOPE OF SAMPLED DATA
C POINTS OF THE TOTAL SIGNAL (DESIRED+MULTIPATH+GARBLE)
C DELAY: DAC DELAY TIME IN NANOSECONDS
C ATT: DAC ATTENUATION CONSTANT
C
C DESCRIPTION OF OUT-ONLY PARAMETERS:
C DACYN: CONTAINS A 0. IF NO DECODE, A 1. IF OTHERWISE
C DACTIM: IF DACYN = 1., THEN DACTIM = DAC DECODE TIME IN
C NANOSECONDS RELATIVE TO THE START OF THE WINDOW
C
C
YNOLD = 1.
DO 999 NT = 1,NPTS
   IF((NT-DELAY).LT.1)THEN
      D = 0.
   ELSE
      D = SIGVID(NT-IFIX(DELAY))
   ENDIF
   A = ATT*SIGVID(NT+1)
   IF((D.GE.A).AND.(YNOLD.LT.1))THEN
      DACYN = 1.
      DACTIM = VIDTIM(NT)
      GOTO 1000
   ENDIF
   IF(D.LT.A)THEN
      YNOLD = 0.
      DACYN = 0.
      DACTIM = 0.
   ENDIF
999 CONTINUE
SUBROUTINE FILT(X,Z,NPTS)
C
C FIVE POLE BUTTERWORTH FILTER
C 1.75 MHz IF FILTER
C
REAL*8 X(8000),Y(8000)
REAL*8 B1,B2,B3,B4,B5,B6
REAL*8 A2,A3,A4,A5,A6
REAL Z(8000)
C
C ASSIGN FILTER COEFFICIENTS
C
B1 = 0.04934405219360D-10
B2 = 0.24672026096802D-10
B3 = 0.49344052193604D-10
B4 = 0.49344052193604D-10
B5 = 0.24672026096802D-10
B6 = 0.04934405219360D-10
C
A2 = -4.96441762645639D0
A3 = 9.8583027368836700
A4 = -9.78839552031834D0
A5 = 4.8595533826767800
A6 = -0.96504297262782D0
C
C COMPUTE FIRST FIVE POINTS PASSED THROUGH FILTER
C
Y(1) = B1*X(1)
Y(2) = B1*X(2) + B2*X(1) - A2*Y(1)
Y(3) = B1*X(3) + B2*X(2) + B3*X(1) - A2*Y(2) - A3*Y(1)
Y(4) = B1*X(4) + B2*X(3) + B3*X(2) + B4*X(1) - A2*Y(3) - A3*Y(2)
> - A4*Y(1)
Y(5) = B1*X(5) + B2*X(4) + B3*X(3) + B4*X(2) + B5*X(1) - A2*Y(4)
> - A3*Y(3) - A4*Y(2) - A5*Y(1)
C
DO 65 N=1,N
Z(N) = SNGL(Y(N))
65 CONTINUE
C
C LOOP THROUGH FILTER EQUATION FOR REST OF ARRAY
C

DO 75 N=6,NPTS
    Y(N) = B1*X(N) + B2*X(N-1) + B3*X(N-2) + B4*X(N-3) + B5*X(N-4)
    > + B6*X(N-5) - A2*Y(N-1) - A3*Y(N-2) - A4*Y(N-3) - A5*Y(N-4)
    > - A6*Y(N-5)
    Z(N) = SNGL(Y(N))
75 CONTINUE
RETURN
END
A.2.8 WINDOWING ROUTINE
This routine determines the 95% peak PFE.

The windowing routine is used to determine the 95% peak PFE according to DME/P measurement methodology. This routine takes the output of the DME/P receiver (NNDME) and processes it to determine the 95% peak PFE.

First, the windowing routine reads in various parameters concerning window height, width and tolerances for PFE. The tolerances for PFE for FA mode and IA mode are 2 and 10 respectively. The next step involves filtering the PFE data using a low pass filter. The filtered PFE data is then copied into arrays for various decision heights. For each decision height, the portion of the flightpath from the starting point to that decision height are copied into the corresponding array. The next step involves the comparison of filtered PFE to the window height using the 95% criteria. This comparison involves determining how many points exceed the window height as the window is moved along the flight trace. If the number of points is less than the tolerance then the height of the window is reduced. The test is repeated until the comparison fails. If the test fails then the window height is incremented to previous window height and this gives the 95% peak PFE. If the test passes for all window heights then the lowest window height is written out as the 95% peak PFE. This process is repeated for arrays of all decision heights.

Figure A.2.6 illustrates the flowchart for the windowing routine. The Fortran code is also given.

SUBROUTINE WINDEX: This routine copies the filtered PFE into various arrays for different decision heights and runs the 95% window check.

Input : Filtered PFE and parameters for the window.
Output: Location of the obstacle (aircraft), 95% peak PFE, start and end of window in which the 95% criteria failed.
START

READ IN PARAMETERS

COPY PFE INTO ARRAYS FOR DECISION HEIGHTS

COMPARE WINDOW OF PFE DATA TO WINDOW HEIGHT IN HTOL ARRAY USING 95% CRITERIA

DID IT PASS?

N

INCREASE WINDOW HEIGHT TO PREVIOUS VALUE OF HTOL

Y

OUTPUT: RUN PASSES FOR CURRENT VALUE OF WINDOW HEIGHT

MOVE WINDOW

N

IS WINDOW AT END OF DATA?

Y

HOLD ARRAYS FOR ALL DECISION HEIGHTS BEEN ANALYZED?

N

HAS RUN PASSED FOR ALL VALUES IN HTOL?

Y

INCREMENT ARRAY INDEX FOR HTOL

N

RETURN

Figure A.2.6 Flow chart for 95% windowing routine.
PROGRAM WINDOW
C
C PROGRAM TO READ RESULT DATA C AND OUTPUT THE LOWEST ERROR BUDGET
C (USING THE 95% CRITERIA) WHICH WAS PASSED BY THE RUN
C
DIMENSION DIST(1350)
DIMENSION IFLAG(3)
C
IFLAG : FLAG TO INDICATE IF ERROR DATA FOR ANY PARTICULAR SYSTEM
C EXIST
C 0 : DATA FILE EXIST
C 1 : NO DATA FILE
C
COMMON/ERRDTA/VERR(1350),X(1350),Y(1350),Z(1350),POSN(1350)
COMMON/CMXMT/ICONT(4,4),NFLTPT,ISYS,ITYPE
COMMON/HEADER(20),IANS,DECNM
COMMON/TITLE/TITLE1(18),TITLE2(18)
C
LOGICAL STOP
REAL NAUT
CHARACTER*80 BUFFER
C
DATA DEGREE/57.29577951/
DATA NAUT/6076.0/
557 DO 88889 I=1,1350
  VERR(I)=0.0
88889 CONTINUE
ISYS = 2
IDUM=0
C
99998 CONTINUE
C
C READ IN NOMINAL ERROR,DISTANCE AND RECEIVER POSITION.
C
C WRITE(6,345)
345 FORMAT(1X,'BEGIN READ STATEMENT')
   READ(31,3)BUFFER
   READ(31,3)BUFFER
   I=1
3 FORMAT(A80)
73 READ(31,36) X(I),Y(I),Z(I),DIST(I),VERR(I),STOP
36 FORMAT(1X,F11.4,1X,F11.4,1X,F11.4,1X,F11.4,1X,F11.4,1X,F11.4,1X,L4)
   POSN(I) = DIST(I)/NAUT
C WRITE(*,*) X(I),DIST(I),POSN(I)
I=I+1
IF(.NOT.(STOP)) GO TO 73
C WRITE(6,542)
542 FORMAT(1X,'END READ STATEMENT')
C 99997 CONTINUE
C 77776 CONTINUE
   IF(I.NE.0) GO TO 77777
   IFLAG(ISYS)=1
   GO TO 99999
77777 CONTINUE
   IFLAG(ISYS)=0
   NFLTPT=I-1
C THIS PART IS THE PFE FILTER FOR DME
C IT TAKES THE INPUT ERROR IN VERR AND FILTERS IT
TEMP1=VERR(1)
TEMP2=VERR(2)
VERR(1)=0.0357260902*TEMP1
VERR(2)=0.0357260902*(TEMP2+2.0*TEMP1)+1.24394614*VERR(1)
DO 333 I=3,NFLTPT
   TEMP3=VERR(I)
   VERR(I)=0.0357260902*(TEMP3+2.0*TEMP2+TEMP1)+1.24394614
      >*VERR(I-1)-0.386850499*VERR(I-2)
      TEMP1=TEMP2
      TEMP2=TEMP3
333 CONTINUE
C WINDEX CHECKS THE PFE AND CMN ERROR AGAINST ERROR BUDGETS
CALL WINDEX
C WRITE(6,761)
761 FORMAT(1X,'END OF Filt AND WINDEX SUBROUTINE')
C 99999 CONTINUE
INPLT=0
IF(IFLAG(ISYS).EQ.1) GO TO 55555
INPLT=INPLT+1
55555 CONTINUE
IF(INPLT.EQ.0) WRITE(6,1002)
1002 FORMAT(10X,'NO DATA/S ARE AVAILABLE')
C STOP
END
C SUBROUTINE WINDEX
C THE PROGRAM TAKES THE ERROR DATA FROM Filt ROUTINE AND
C CHECKS TO SEE IF IT MEETS ERROR BUDGET CRITERIA.
C THE PROGRAM FIRST CHECKS THE 200% BUDGET. IF THE BUDGET IS
C PASSED THEN THE DATA IS CHECKED ON THE 190% BUDGET. THIS CONTINUES
C UNTIL EITHER THE 10% BUDGET HAS BEEN PASSED OR THE DATA HAS FAILED
C TO PASS A CERTAIN BUDGET IN WHICH CASE THE LOWEST PASSED BUDGET
C IS PRINTED OUT.
'NFLPT' is the total number of data points for the run
'ZERR' contains CMN error
'VERR' contains PFE error

COMMON/ERRDTA/VERR(1350),X(1350),Y(1350),Z(1350),POSN(1350)
COMMON/CMXMT/ICONT(4,4),NFLPT,ISYS,ITYPE
DIMENSION PFE(1350,14),CMN(1350,14),POS(1350,14)

REAL HTOL(26)

INTEGER D,DD,E,U,FC,FP
INTEGER CNT(14)

HTOL is the array for window heights. The window height is
is gradually reduced from 30ft to 0.5ft

DATA HTOL/030.0,0.025.0,0.020.0,15.0,
> 11.0,10.5,10.0,09.5,
> 09.0,08.5,08.0,07.5,07.0,06.5,06.0,
> 05.5,05.0,04.5,04.0,03.5,03.0,2.5,2.0,1.5,1.0,0.5/
LINDX = 26

TOLER = total number of points per window which are
allowed to be outside the window height

READ(5,77445)MTOLER
77445 FORMAT(I2)

READ WINDOW WIDTH (NUMBER OF POINTS PER WINDOW)
READ(5,2134)NWID1
2134 FORMAT(I3)

READ WINDOW HEIGHT
READ(5,6754)HEIGHT
6754 FORMAT(F7.5)
READ(5,5024)XPOS
5024 FORMAT(F10.3)
READ(5,8047)YPOS
8047 FORMAT(F10.3)
READ(5,2947)PHIGHT
2947 FORMAT(F7.5)
READ(5,33544)MPTOL
33544 FORMAT(I2)

CHTE = HEIGHT*2.0

READ IN DATA

******************************
CREATE DATA ARRAY FOR VARIOUS DECISION HEIGHTS

DO 7889 K = 1,NFLPT
WRITE(*,*) POSN(K),VERR(K)
IF(POSN(K).GT.(2.013))GOTO 12
CMN(K,14) = ZERR(K)
PFE(K,14) = VERR(K)
POS(K,14) = POSN(K)
CNT(14) = K

12 CONTINUE
IF(POSN(K).GT.(2.327)) GOTO 13

CMN(K,13) = ZERR(K)
PFE(K,13) = VERR(K)
POS(K,13) = POSN(K)
CNT(13) = K

13 CONTINUE
IF(POSN(K).GT.(2.642)) GOTO 14

CMN(K,12) = ZERR(K)
PFE(K,12) = VERR(K)
POS(K,12) = POSN(K)
CNT(12) = K

14 CONTINUE
IF(POSN(K).GT.(2.956)) GOTO 15

CMN(K,11) = ZERR(K)
PFE(K,11) = VERR(K)
POS(K,11) = POSN(K)
CNT(11) = K

15 CONTINUE
IF(POSN(K).GT.(3.271)) GOTO 16

CMN(K,10) = ZERR(K)
PFE(K,10) = VERR(K)
POS(K,10) = POSN(K)
CNT(10) = K

16 CONTINUE
IF(POSN(K).GT.(3.585)) GOTO 17

CMN(K,9) = ZERR(K)
PFE(K,9) = VERR(K)
POS(K,9) = POSN(K)
CNT(9) = K

17 CONTINUE
IF(POSN(K).GT.(3.900)) GOTO 18

CMN(K,8) = ZERR(K)
PFE(K,8) = VERR(K)
POS(K,8) = POSN(K)
CNT(8) = K

18 CONTINUE
IF(POSN(K).GT.(4.057)) GOTO 19

CMN(K,7) = ZERR(K)
PFE(K,7) = VERR(K)
POS(K,7) = POSN(K)
CNT(7) = K

19 CONTINUE
IF(POSN(K).GT.(4.214)) GOTO 20

CMN(K,6) = ZERR(K)
PFE(K,6) = VERR(K)
POS(K,6) = POSN(K)
CNT(6)=K
20 CONTINUE
IF(POSN(K).GT.(4.371))GOTO 21
C C MN(K,5)=ZERR(K)
PFE(K,5)=VERR(K)
POS(K,5)=POSN(K)
CNT(5)=K
21 CONTINUE
IF(POSN(K).GT.(4.528))GOTO 22
C C MN(K,4)=ZERR(K)
PFE(K,4)=VERR(K)
POS(K,4)=POSN(K)
CNT(4)=K
22 CONTINUE
IF(POSN(K).GT.(4.686))GOTO 23
C C MN(K,3)=ZERR(K)
PFE(K,3)=VERR(K)
POS(K,3)=POSN(K)
CNT(3)=K
23 CONTINUE
IF(POSN(K).GT.(4.843))GOTO 24
C C MN(K,2)=ZERR(K)
PFE(K,2)=VERR(K)
POS(K,2)=POSN(K)
CNT(2)=K
24 CONTINUE
C C MN(K,1)=ZERR(K)
PFE(K,1)=VERR(K)
POS(K,1)=POSN(K)
CNT(1)=NFLTPT
C C LOOP THROUGH THE WINDOW FOR EACH CATEGORY
C
U = 0
LWINC = 1
LWINP = 1
DO 3902 E = 1,29,2
IF(E.EQ.5) GOTO 3902
D=E
IF (E.EQ.15) D=16
IF (E.EQ.9) D=20
DD=D+1
U = U + 1
FP = 0
FC = 0
NWIDTH = NWID1
IF((CNT(U)).LT.NWIDTH) NWIDTH = CNT(U)
C C 'K' IS THE INDEX OF FIRST POINT OF THE LAST WINDOW
77 K = CNT(U) - NWIDTH + 1
C START OF PFE CHECK LOOP
C
    IF(LWINP.EQ.LINDX) GOTO 104
144   DO 600 LL = LWINP,26
C
C BEGIN CHECK LOOP
400   DO 98 I = 1,K
      MERRPT = 0
      N = I + NWIDTH - 1
C
C CHECK THE WINDOW
    DO 121 J = I,N
      IF(ABS(PFE(J,U)).GT.HTOL(LL)) MERRPT = MERRPT + 1
      IF((MERRPT.GT.MPTOL).AND.(I.EQ.I)) FP = 1
      IF(MERRPT.GT.MPTOL) GOTO 576
121  CONTINUE
98    CONTINUE
    IF(LL.NE.LINDX) GOTO 600
104   CONTINUE
    WRITE(D,3)XPOS,YPOS,HTOL(LINDX),POSN(I),POSN(N),FP
C
    WRITE(6,*)XPOS,YPOS,HTOL(LINDX),POSN(I),POSN(N)
    LWINP = LINDX
    GOTO 3902
C
C INCREMENT BUDGET TO THE LOWEST BUDGET WHICH HAS BEEN PASSED
576   IF(LL.EQ.1) NL = 2
      IF(LL.GT.1) NL = LL
      NL = NL - 1
      WRITE(D,3)XPOS,YPOS,HTOL(NL),POSN(I),POSN(N),FP
C
    WRITE(6,*)XPOS,YPOS,HTOL(NL),POSN(I),POSN(N)
    LWINP = NL
    GOTO 3902
600   CONTINUE
3    FORMAT(F9.3,1X,F9.2,1X,F9.4,1X,F9.6,1X,F9.6,2X,l1)
3902  CONTINUE
    RETURN
END
APPENDIX A.3

ROUTINE FOR DETERMINATION OF GRID SPACING - GSPACE

This program is used to determine the pointwise error and contour deformation for various bandwidths of the 2-D rectangular low pass filter. The least bandwidth to satisfy the error criteria is used to determine the optimal grid spacing as illustrated in Section IV. The flowchart for this program is given in Figure A.3. The corresponding FORTRAN code is included subsequently.

First the program prompts the user to enter the name of the input contour data file, the grid limits, and the contour level to be examined. Then the program opens the corresponding input and output files. The program reads the PFE contour data into a two dimensional array. A binary contour is also obtained from the PFE contour data using the contour level as the threshold. The next step involves the determination of 2-D DFT of the contour data. The 2-D DFT is obtained using a row-column decomposition algorithm. The resulting 2-D Fourier spectrum is then filtered using a 2-D rectangular filter whose bandwidth is entered by the user at the request of the program. The reconstructed contour is obtained from this filtered spectrum using 2-D inverse DFT. The binary reconstructed contour is generated by means of thresholding.

The reconstructed contour is compared with the actual contour to obtain the pointwise reconstruction error. The pointwise error determined is written to the output file. The next step involves the comparison of the contours in the constant PFE planes using the binary contours obtained earlier. Edges in the binary contours are extracted using an edge detection algorithm. The Sobel templates are applied to the binary contours to determine the gradient at each edge point and the value of the gradient on the boundary of the contours is written out to an output file. The next step is to establish the connectivity of the boundary points on the reconstructed contour using a contour following algorithm. Once this is completed, the deformation is measured along the gradient at each point on the actual contour. This requires the determination of the point of intersection of the gradient with the reconstructed contour. Once the point of
intersection is determined, the deformation is calculated and entered in the output file. The pointwise reconstruction error and the deformation obtained are then checked to see if they exceed the tolerances. The least bandwidth of the filter that meets the tolerances is used to determine the grid spacing.

The various subroutines and their functions are listed below as the main routines of the software developed for the above operations.

**SUBROUTINE TWODFFT:** This routine is used to determine the 2-D DFT or 2-D Inverse DFT as desired. The choice of DFT or inverse DFT is specified using a flag. If the flag=1, then DFT; if flag=-1 then inverse DFT is obtained. If flag=-1 then the conjugate of the input array is determined before determining DFT. This routine determines the row-wise 1-D DFT's by passing each row of the contour to the DFT routine. The resulting 2-D array is passed columnwise to the DFT routine to determine the final output.

**Input:** 2-D array data for determination of 2-D DFT or inverse DFT.

**Flag to choose DFT or inverse DFT.**

**Output:** 2-D array after 2-D DFT or inverse DFT.

**SUBROUTINE PERCENT:** This routine determines the percentage error at each point of the contour data. It also obtains the error distribution.

**Input:** Actual contour, reconstructed contour, and contour level to examine.

**Output:** Percentage error, maximum error, mean error, and error distribution.

**SUBROUTINE PRELEDGE:** Determines the edge of the binary contour using binary edge detection templates.
Input : Binary contour and maximum array indices.

Output : Edge detected contour.

SUBROUTINE SOBEL: Determines the gradient at each point on the contour using the Sobel templates.

Input : Edge detected contour and maximum array indices.

Output : Magnitude and direction of gradient at each point on the contour.

SUBROUTINE SIMPOSE: Superimposes edge detected contours (actual and reconstructed contours) which helps visualize the deformation.

Input : Actual and reconstructed edge detected contours.

Output : Superimposed contour boundaries.

SUBROUTINE EDFOLLOW: Determines the connectivity on the boundary of the reconstructed contour. Uses a contour following algorithm.

Input : Detected edges of the reconstructed contour and maximum array indices.

Output : 2-D array of connected points and number of points.

SUBROUTINE MINSTART: Determines the starting point of the contour to be followed.

Input : Detected edges of the reconstructed contour and maximum array indices.

Output : Coordinates of starting point of contour.

SUBROUTINE ZEROOUT: Zeroes out contour points that have already been located on the contour.

Input : Detected edges of the reconstructed contour and 2-D array of points that have been already processed by contour following.

Output : Detected edges of the new reconstructed contour with previously
detected contour points set to zero.

SUBROUTINE FFT : Determines the 1-D FFT of the input array.

   Input : Input 1-D array and the number of points in FFT.
   Output : 1-D FFT or inverse FFT of input array.

SUBROUTINE CONJUGATE: Returns the complex conjugate of the input 2-D array. Called by
   TWODFFT in order to obtain inverse DFT.

   Input : 2-D array whose complex conjugate is required.
   Output : Complex conjugate of input array.

SUBROUTINE BLIMIT: This is the 2-D rectangular low pass filter.

   Input : 2-D Spectrum and bandwidth.
   Output : Filtered spectrum.
Figure A.3 Flow chart for grid spacing determination (GSPACE).
PROGRAM TO DETERMINE GRID SPACING-RETURNS POINTWISE RECONSTRUCTION ERROR AND DEFORMATION OF CONTOUR AFTER RECONSTRUCTION FOR DIFFERENT BANDWIDTHS OF FILTER.

PROGRAM GSPACE
COMPLEX ERR(128,128),OERR(128,128)
COMPLEX RERR(128,128),BRER(128,128)
INTEGER BINERR(128,128),B12ERR(128,128)
INTEGER CHR(128,128),CHR2(128,128)
INTEGER CHRAR(128,128),CHRAC2(128,128)
INTEGER INX(256),IJX(256),IN(256),IJ(256)
REAL BTA(256),B2A(256),XERR(4096,2)
CHARACTER*15 NAME

C WRITE(*,*)'ENTER INPUT FILENAME'
READ(*,5)NAME
FORMAT(A)

WRITE(*,*)'ENTER SPACING ALONG X-DIRECTION'
READ(*,*)XSP

WRITE(*,*)'ENTER SPACING ALONG Y-DIRECTION'
READ(*,*)YSP

WRITE(*,*)'ENTER THRESHOLD'
READ(*,*)THRESH

WRITE(*,*)'ENTER STARTING VALUE IN X'
READ(*,*)XINIT

WRITE(*,*)'ENTER STARTING VALUE IN Y'
READ(*,*)YINIT

WRITE(*,*)'ENTER X-RANGE'
WRITE(*,*)' MINIMUM X-VALUE'
READ(*,*)XMIN

WRITE(*,*)' MAXIMUM X-VALUE'
READ(*,*)XMAX

WRITE(*,*)'ENTER Y-RANGE'
WRITE(*,*)' MINIMUM Y-VALUE'
READ(*,*)YMIN

WRITE(*,*)' MAXIMUM Y-VALUE'
READ(*,*)YMAX

C OPEN INPUT AND OUTPUT FILES
OPEN(UNIT=12,IOSTAT=IOS,FILE=NAME,STATUS="OLD")
OPEN(UNIT=21,IOSTAT=IOS,FILE="O",FORMAT="NEW")
OPEN(UNIT=31,IOSTAT=IOS,FILE=*GRAD*,STATUS="NEW")
C INPUT DATA FROM CONTOUR FILE IS READ IN HERE
10 READ(12,*,END=20)X,Y,E,SW1,SW2,FL
   IF (((X.GE.XMIN).AND.(X.LE.XMAX)).AND.((Y.GE.YMIN)
   .AND.(Y.LE.YMAX))) THEN
      IX=(X-XMIN)/XSP + 2
      IY=(Y-YMIN)/YSP + 2
C IX AND IY ARE INDICES AND ERR IS THE ERROR AT EACH POINT ON THE GRID
   ERR(IX,IY)=E
C THE ARRAY BINERR CONTAINS THE BINARY IMAGE. HERE WE THRESHOLD
C THE ERROR BASED ON THE THRESHOLD VALUE - THRESH.
   IF(E .GE. THRESH) THEN
      BINERR(IX,IY)=1
   ELSE
      BINERR(IX,IY)=0
   END IF
   END IF
   GO TO 10
20 CONTINUE
C HERE WE PAD ZEROS BEFORE TAKING THE FFT.
C
XMAX=X
C
YMAX=Y
   DO 30 I=IX+1,128
      DO 30 J=IY+1,128
         ERR(I,J)=0.0
      END IF
      END IF
   GO TO 10
30 CONTINUE
C HERE WE TAKE THE TWODFFT OF THE ERROR CONTOUR
C INPUT ARRAY IS ERR. OUTPUT ARRAY IS OERR
C IRS IS A FLAG TO INDICATE IF FFT OR INVERSE FFT IS REQUIRED
C IRS = 1 MEANS FFT WHILE IRS=-1 MEANS INVERSE FFT
C
CALL TWODFFT(ERR,IRS,OERR)
C
WRITE(*,*)'ENTER VALUE FOR BANDWIDTH'
READ(*,*)JB
C
DO 102 JB=24,128,2
C INSERT VALUES FOR BANDWIDTH
   BW1=FLOAT(JB)
   BW2=FLOAT(JB)
WRITE(21,*)'BANDWIDTH = ',BW1
C LOW PASS FILTER ROUTINE. TRUNCATION OF TERMS OUTSIDE BW1
   CALL BLIMIT(OERR,BRER,BW1,BW2)
C INVERSE TWO-D FFT
   IRS=-1
   CALL TWODFFT(BRER,IRS,RERR)
   DO 50 I=1,IX
      DO 51 J=1,IY
         IF(CABS(RERR(I,J)).GE.THRESH) THEN
            BI2ERR(I,J)=1
   END IF
   END IF
   END IF
   END IF
   GO TO 10
50 CONTINUE
51 CONTINUE
ELSE
    B12ERR(I,J)=0
END IF
51 CONTINUE
50 CONTINUE
C
C DETERMINE POINTWISE RECONSTRUCTION ERROR
    CALL PERCENT(ERR,RERR,THRESH)
C DETERMINE EDGE POINTS OF ACTUAL CONTOUR
    CALL PRELEDGE(BINERR,CHR,IX,IY)
C DETERMINE GRADIENT AT EACH EDGE POINT ON ACTUAL CONTOUR
    CALL SOBEL(BINERR,CHR,IX,IY)
C DETERMINE EDGE POINTS ON RECONSTRUCTED CONTOUR
    CALL PRELEDGE(B12ERR,CHR2,IX,IY)
C DETERMINE GRADIENT AT EACH POINT ON RECONSTRUCTED CONTOUR
    CALL SOBEL(B12ERR,CHR2,IX,IY)
C SUPERIMPOSE THE TWO BOUNDARIES. HELPS USER VISUALIZE.
    CALL SIMPOSE(CHR,CHR2,IX,IY)
C
    CALL EDFOLLOW(CHR2,IX,IY,XERR,JCOUNT)
    JCOUNT=JCOUNT+1
    XERR(JCOUNT,1)=300.0
    XERR(JCOUNT,1)=300.0
    REWIND(31)
    READ(31,*)NAME2
    ICOUNT=0
363 ICOUNT=ICOUNT+1
    READ(31,19)INX(ICOUNT),IJX(ICOUNT),GX,BX,BY,B2A(ICOUNT)
19 FORMAT(13,W,13,W,4(F9.3,2X))
    IF(B2A(ICOUNT).NE.-1.111) GOTO 363
C DETERMINE DEFORMATION BETWEEN ACTUAL AND RECONSTRUCTED CONTOURS
    DO 565 I=1,ICOUNT-1
        DISTEMP=3000.0
        DISTEMP2=3000.0
        CCT=0.0
        IF(B2A(I).NE.90.0) THEN
            SLOPE1 = TAN(B2A(I)*ACOS(-1.0)/180.0)
        END IF
        DO 567 J=1,JCOUNT-1
            IDX=INX(I)-XERR(J,1)
            IDY=IJX(I)-XERR(J,2)
            DIST=SQRT(FLOAT(IDY**2+IDX**2))
            IF(DISTEMP2.GE.DIST) THEN
                DISTEMP2=DIST
                DX1=XERR(J,1)
                DY1=XERR(J,2)
                BA1=BTA(J)
            END IF
            IF(IDX.NE.0) THEN
                TEMPAN = ATAN(FLOAT(IDY)/FLOAT(IDX))
TEMPAN = TEMPAN * 180.0 / (ACOS(-1.0))
    IF (IDX.GT.0) THEN
      TEMPAN = 180.0 + TEMPAN
    END IF
ELSE
    IF (IDX.LT.0) THEN
      TEMPAN = 90.
    END IF
    IF (IDX.GT.0) THEN
      TEMPAN = 270.
    END IF
    IF (IDX.EQ.0) THEN
      TEMPAN = B2A(I)
    END IF
ENDIF
ENDIF
  CCT = CCT + 1.0
ENDIF
ELSE
    IDX2 = INX(I) - XERR(J+1,1)
    IDY2 = IXJ(I) - XERR(J+1,2)
    IF (IDX2.NE.0) THEN
      TEMPAN2 = ATAN(FLOAT(IDY2)/FLOAT(IDX2))
      TEMPAN2 = TEMPAN2 * 180.0 / (ACOS(-1.0))
      IF (IDX2.GT.0) THEN
        TEMPAN2 = 180.0 + TEMPAN2
      END IF
      IF (IDX2.LT.0) THEN
        TEMPAN2 = 90.
      END IF
      IF (IDX2.EQ.0) THEN
        TEMPAN2 = B2A(I)
      END IF
    ELSE
      IF (IDX2.LT.0) THEN
        TEMPAN2 = 90.
      END IF
      IF (IDX2.GT.0) THEN
        TEMPAN2 = 270.
      END IF
      IF (IDX2.EQ.0) THEN
        TEMPAN2 = B2A(I)
      END IF
    END IF
ENDIF
SUB1 = B2A(I) - TEMPAN
SUB2 = B2A(I) - TEMPAN2
PROD = SUB1 * SUB2
SUB3 = B2A(I) - TEMPAN + 180.
SUB4 = B2A(I) - TEMPAN2 + 180.
PROD2 = SUB3 * SUB4
IF ((PROD.LT.0.0).OR.(PROD2.LT.0.0)) THEN
DIFFX1 = (XERR(J,1) - XERR(J+1,1))
DIFFY1 = (XERR(J,2) - XERR(J+1,2))
IF(DIFFX1.EQ.0.0) THEN
  XED = XERR(J,1)
  YED = SLOPE1*(X-XED) + IJX(I)
ELSE
  SLOPE2 = DIFFY1 / DIFFX1
  XED = (-1)*INX(I)*SLOPE1 + IJX(I) - XERR(J,2) + SLOPE2*XERR(J,1)
  IF(SLOPE1.NE.SLOPE1) THEN
    XED = XED / (SLOPE2 - SLOPE1)
  END IF
END IF

DIST = SQRT(((INX(I) - XED)**2 + (IJX(I) - YED)**2)
CCT = CCT + 1
IF(DISTEMP.GE.DIST) THEN
  DX = XED
  DY = YED
  BA = B2A(I)
  DISTEMP = DIST
END IF
END IF
END IF
CONTINUE
IF((CCT.EQ.0.0).OR.(DISTEMP.GT.12.0)) THEN
  DISTEMP = DISTEMP2
  DX = DX1
  DY = DY1
  BA = BA1
END IF
IF(DISTEMP.NE.3000.) THEN
  WRITE(21,9) INX(I), IJX(I), B2A(I), DX, DY, BA, DISTEMP
END IF
9 FORMAT(13, 2X, 13, 2X, 5(F8.3, 2X), 2X, F6.2)
CONTINUE
CONTINUE
CLOSE(21)
CLOSE(12)
STOP
END

SUBROUTINE ROUND(RX, IX)
C ROUTINE TO PROVIDE FLOATS ROUNDED TO NEAREST INTEGER
  CRX = RX - IFIX(RX)
  IF(CRX.LT. 0.5) THEN
    IX = IFIX(RX)
  ELSE
    IX = IFIX(RX) + 1
  END IF
  RETURN
SUBROUTINE PREEDGE(BR,BR2,IX,IY)
C ROUTINE TO DETERMINE THE EDGE POINTS ON INPUT CONTOUR
INTEGER BR(128,128),BR2(128,128)
C WRITE(21,'*IMAGE*')
DO 10 I = 1,IX
   DO 20 J = 1,IY
      IF(BR(I,J).EQ.1)
         IF((I.EQ.1).OR.(I.EQ.IX).OR.(J.EQ.1).OR.(J.EQ.IY))
            GOTO 30
         ELSE
            IF((BR(I,J-1)*BR(I,J+1)*BR(I-1,J)*BR(I+1,J)).EQ.0)
               GOTO 30
         END IF
      END IF
   GOTO 40
10 CONTINUE
   CONTINUE
   WRITE(21,'*')(BR2(1,J),J=1,IY)
 CONTINUE
RETURN
END

SUBROUTINE MINSTART(NERR,IX,IY,XST,YST)
C ROUTINE TO DETERMINE THE STARTING POINT FOR CONTOUR
INTEGER NERR(128,128)
COMMON /SPACE/XSP,YSP,XSP2,YSP2
COMMON /INITIAL/XINIT,YINIT,XINIT2,YINIT2
COMMON /INT/XMIN,XMAX,YMIN,YMAX
DO 10 I=1,IX
   DO 20 J=1,IY
      IF(NERR(I,J).EQ.1)
         XST = FLOAT(I)
         YST = FLOAT(J)
         GO TO 30
      END IF
   CONTINUE
20 CONTINUE
10 CONTINUE
XST=0.0
YST=0.0
30 CONTINUE
RETURN
END

SUBROUTINE EDFOLLOW(CHR,IX,IY,XERR,ICTR)
C ROUTINE TO DETERMINE THE CONNECTEDNESS ON RECONSTRUCTED CONTOUR

INTEGER CHR(128,128)
REAL XERR(4096,2)
CHARACTER CHRAC

COMMON /SPACE/XSP,YSP,XSP2,YSP2
COMMON /INITIAL/XINIT,YINIT,XINIT2,YINIT2
COMMON /INT/XMIN,XMAX,YMIN,YMAX

DIMENSION CHRAC(128,128)
ICTR=0

12 CALL MINSTART(CHR,IX,IY,XST,YST)
   DX=0
   DY=0
   XMK=0
   YMK=0
   DIST=0
   IF(XST.EQ.0.0 .AND. YST.EQ.0.0) THEN
     GOTO 210
   ELSE
     I=XST
     J=YST
   END IF
   CHRAC(I,J)= 'S'
   ICTR=ICTR+1
10   ICOUNT = 0
20   IF(XMK.EQ.0.0 .AND. YMK.EQ.0.0) THEN
       XD=-1
       YD=-1
   ELSE
       DX=I-XMK
       DY=J-YMK
   END IF
   IF(DX.EQ.0.0) THEN
     IF(DY.EQ.1.0) THEN
       XD=-1
       YD=-1
     ELSE
       IF(DY.EQ.-1.0) THEN
         XD=1
         YD=1
       END IF
     END IF
   END IF
   IF(DX.EQ.1.0) THEN
     IF(DY.EQ.1.0) THEN
       XD=-1
       YD=0
     ELSE
       IF(DY.EQ.-1.0) THEN
         XD=0
         YD=1
       ELSE
         IF(DY.EQ.0.0) THEN
           XMK=I
           YMK=J
         ELSE
           IF(DY.EQ.1.0) THEN
             XD=-1
             YD=0
           ELSE
             IF(DY.EQ.-1.0) THEN
               XD=0
               YD=1
             ELSE
               IF(DY.EQ.0.0) THEN
                 XMK=I
                 YMK=J
               ELSE
                 XMK=I-1
                 YMK=J
               END IF
             END IF
           END IF
         END IF
       END IF
     END IF
   END IF
   ICTR=ICTR+1
210  RETURN
XD=-1
YD=1
END IF
END IF
IF(DX.EQ.-1.0)
THEN
IF(DY.EQ.-1.0)
THEN
XD=1
YD=0
ELSE IF(DY.EQ.0.0)
THEN
XD=1
YD=-1
ELSE IF(DY.EQ.1.0)
THEN
XD=0
YD=-1.0
END IF
END IF
ICOUNT=ICOUNT+1
XMK=I+XD
YMK=J+YD
IF(CHR(I,J).EQ.1)
THEN
XERR(CTR,1)=I
XERR(CTR,2)=J
END IF
IF(CHR(XMK,YMK).EQ.1)
THEN
IF(CHRAC(XMK,YMK).NE.'S')
THEN
CHRAC(XMK,YMK)='I'
END IF
ELSE
IF(ICOUNT.EQ.8)
THEN
GO TO 70
ELSE
GOTO 20
END IF
END IF
XPREV=I
YPREV=J
I=XMK
J=YMK
XMK=XPREV
YMK=YPREV
C WRITE(*,*)I,J
IF(CHRAC(I,J).EQ.'S')
THEN
C WRITE(21,*)'CURVE ENDS'
GO TO 205
END IF
GO TO 10
70 WRITE(*,*)'RM POINT',I,J,XMK,YMK
205 CALL ZEROUT(CHR,CHRAC)
C WRITE(*,*)'COUNTER =',ICTR
GOTO 12
210 CONTINUE
RETURN
END

SUBROUTINE ZEROUT(CHR,CHARAC)
INTEGER CHR(128,128)
CHARACTER CHARAC
DIMENSION CHARAC(128,128)
DO 10 I=1,128
DO 20 J=1,128
IF(CHARAC(I,J).EQ.'I' .OR. CHARAC(I,J).EQ.'S') THEN
   CHR(I,J)=0.0
END IF
10 CONTINUE
20 CONTINUE
10 CONTINUE
RETURN
END

SUBROUTINE TWODFFT(XERR,IZ,ZERR)
COMPLEX XERR(128,128),X(128),ZERR(128,128)
IF(IZ.EQ.-1) THEN
   CALL CONJUGATE(XERR)
END IF
DO 30 I=1,128
DO 10 J=1,128
X(J)=XERR(I,J)
10 CONTINUE
CALL FFT(X,7)
DO 20 J=1,128
ZERR(I,J)=X(J)
20 CONTINUE
30 CONTINUE
DO 60 J=1,128
DO 40 I=1,128
X(I)=128.0*ZERR(I,J)
40 CONTINUE
CALL FFT(X,7)
DO 50 I=1,128
ZERR(I,J)=X(I)
50 CONTINUE
IF((IZ.EQ.-1).AND.(CABS(ZERR(I,J)).LT.0.1)) ZERR(I,J)=0.0
60 CONTINUE
RETURN
END

SUBROUTINE FFT(F,LN)
COMPLEX F(128),U,W,T,CMPLX
PI=3.141593
N=2**LN
NV2=N/2
NM1=N-1
J=1
DO 3 I = 1,NM1
IF(I.GE.J) GO TO 1
T=F(J)
F(J)=F(I)
F(I)=T
1  K=NV2
2  IF (K.GE.J) GO TO 3
J=J-K
K=K/2
GO TO 2
3  J=J+K
DO 5 L=1,LN
LE=2**L
LE1=LE/2
U=(1.0,0.0)
W=CMPLX(COS(PI/LE1),SIN(PI/LE1))
DO 5 J =1,LE1
DO 4 I=J,N,LE
IP=I+LE1
T=F(IP)*U
F(IP)=F(I)-T
4  F(I)=F(I)+T
5  U=U*W
DO 6 I=1,N
F(I)=F(I)/FLOAT(N)
6  CONTINUE
CONTINUE
RETURN
END

SUBROUTINE CONJUGATE(Y)
C ROUTINE TO DETERMINE CONJUGATE OF COMPLEX ARRAY
C REQUIRED BEFORE INVERSE FFT
COMPLEX Y(128,128)
DO 10 I=1,128
DO 20 J=1,128
Y(I,J)=CONJG(Y(I,J))
20  CONTINUE
10  CONTINUE
RETURN
END

SUBROUTINE BLIMIT(BLERR,BRERR,BW1,BW2)
C ROUTINE TO BANDLIMIT
C RECTANGULAR FILTER IMPLEMENTED HERE.
COMPLEX BLERR(128,128),BRERR(128,128)
TEMP2=BW2/2.0
TEMP = BW1/2.0
DO 10 I=1,128
DO 10 J=1,128
IF((I.GT.IFIX(TEMP)).AND.(I.LT.IFIX(128.0-TEMP)))THEN
BRERR(I,J)=CMPLX(0.,0.)
ELSE
BRERR(I,J)=BLERR(I,J)
END IF
IF((J.GT.IFIX(TEMP2)).AND.(J.LT.IFIX(128.0-TEMP2)))THEN
BRERR(I,J)=CMPLX(0.,0.)
END IF
10 CONTINUE
RETURN
END

SUBROUTINE PERCENT(PERR,PRERR,THRESH)
C RECONSTRUCTION ERROR IS DETERMINED IN THIS ROUTINE
C ERROR DISTRIBUTIONS ARE ALSO OBTAINED
COMPLEX PERR(128,128),PRERR(128,128)
INTEGER HIST(64)
PMAXERR=0.0
WRITE(21,*)' PFE',' PFE REC',' PERCENT ERROR'
DO 10 I = 1, 128
  DO 20 J = 1, 128
    IF(CABS(PERR(I,J)).GT.THRESH) THEN
      DIFF = CABS(PERR(I,J)-PRERR(I,J))
      CALL ROUND(DIFF,IDIFF)
      HIST(IDIFF+1)=HIST(IDIFF+1)+1
      WRITE(21,19)CABS(PERR(I,J)),CABS(PRERR(I,J)),DIFF/CABS(PERR(I,J))
19    FORMAT(3(F10.4,2X))
      IF(DIFF.GT.PMAXERR) THEN
        PMAXERR=DIFF
        ERR=CABS(PERR(I,J))
      END IF
    END IF
20 CONTINUE
10 CONTINUE
QERR=PMAXERR/ERR
WRITE(21,9)PMAXERR,ERR,QERR
9  FORMAT(13HMAX. ERROR = ,F5.2,2X,8HERROR = ,F5.2,2X,
<16HPERCENT ERROR = ,F6.2)
    WRITE(21,*)'ERROR VALUE ',' NUMBER OF POINTS'
    DO 30 L = 1,32
      WRITE(21,29)L-1,HIST(L)
29    FORMAT(3X,I3,9X,I4)
      AVRG=AVRG+(L-1)*HIST(L)
      TOT=TOT+HIST(L)
30 CONTINUE
WRITE(21,*)'MEAN ERROR = ', AVRG/TOT
RETURN
END
SUBROUTINE SOBEL(BERR,CHR,IX,IY)
C ROUTINE TO DETERMINE GRADIENT
INTEGER BERR(128,128),CHR(128,128)
WRITE(31,*)'EDGE GRADIENT'
DO 10 I = 1,IX
DO 20 J = 1,IY
IF(CHR(I,J).EQ.1) THEN
  IF(J.NE.1) THEN
    BX=(BERR(I+1,J+1)-BERR(I-1,J+1)+2*(BERR(I+1,J)-BERR(I-1,J)) +
    < BERR(I,J+1)-BERR(I,J-1))
    BY=(BERR(I+1,J+1)-BERR(I+1,J-1)+2*(BERR(I,J+1)-BERR(I,J-1)) +
    < BERR(I-1,J+1)-BERR(I-1,J-1))
  ELSE
    BX=(BERR(I+1,J+1)-BERR(I-1,J+1)+2*(BERR(I+1,J)-BERR(I-1,J)))
    BY=(BERR(I+1,J+1)+2*(BERR(I,J+1)) +
    < BERR(I-1,J+1))
  END IF
  G=SQRT(BX**2 + BY**2)
  IF(BX.NE.0) THEN
    BTA=(180*ATAN(BY/BX))/(ACOS(-1.0))
    BK=(180*ASIN(BY/G))/ACOS(-1.0)
  ELSE
    IF(BY.NE.0) THEN
      BTA=(180*ASIN(BY/G))/(ACOS(-1.0))
    ELSE
      BTA=0.0
    END IF
  END IF
  WRITE(31,9)I,J,G,BX,BY,BTA
9 FORMAT(3X,13,F9.3,1X,13,F9.3,1X)
END IF
20 CONTINUE
10 CONTINUE
BTA=-1.111
L=111
WRITE(31,9)L,L,BTA,BTA,BTA,BTA
RETURN
END

SUBROUTINE SIMPOSE(BINERR,B12ERR,IX,IY)
C SUPERIMPOSED VERSION OF ACTUAL AND RECONSTRUCTED CONTOUR
C BOUNDARIES. USED AS AN AID TO VISUALIZE
INTEGER BINERR(128,128),B12ERR(128,128)
INTEGER X(128)
DO 10 I = 1,IX
DO 20 J = 1,IY
  IF(B12ERR(I,J).EQ.1) THEN
    X(J)=2
  ELSE
    X(J)=0
  END IF
20 CONTINUE
10 CONTINUE
RETURN
END
END IF

20    CONTINUE

C    WRITE(21,*)(BINERR(I,J)+X(J),J=1,IY)

10    CONTINUE

RETURN

END
APPENDIX A.4

ROUTINE FOR VERIFICATION OF GRID SPACING - VERIFY

This program is used to verify the optimal grid spacing obtained using GSPACE. In this program, the high resolution contour and optimal grid spacing contour are compared and the contour deformation is measured. If the deformation is below 100 feet, then the optimal grid spacing chosen is verified; otherwise the bandwidth is increased and a new grid spacing is determined using GSPACE. The determination and verification process should proceed iteratively until the optimal grid spacing satisfying the tolerances is determined.

First the program prompts the user to enter the names of the input contour data files - both the high resolution contour data and the optimal grid spacing contour data, the grid limits, and the contour level to be examined. Then the program opens the corresponding input and output files and reads the PFE contour data into two 2-D arrays. A binary contour is also obtained from the high resolution PFE contour data using the contour level as the threshold. The next step involves the interpolation of the optimal grid spacing contour to the high resolution grid using a bilinear interpolation scheme. The binary interpolated contour is also obtained for further use. The resultant binary contours are then compared in the constant PFE planes using the method described earlier. The deformation thus obtained should be less than 100 feet. The flowchart for this routine is given in Figure A.4. The corresponding FORTRAN code is given subsequently.

This routine uses the subroutines preledge, sobel, edfollow, minstart and zeroout which were explained in Appendix A.3. It also uses the interpolation routine INTERPOL.

SUBROUTINE INTERPOL: 

This routine interpolates the optimal grid spacing contour on to the high resolution contour using bilinear interpolation.

Input : Optimal grid spacing contour data, array indices, optimal grid spacing, and high resolution grid spacing.

Output : Interpolated optimal grid spacing contour.
Figure A.4 Illustration of grid spacing verification (VERIFY).
VERIFY
Program to verify the grid spacing determined using GSPACE
This program compares the optimal grid spacing contour and
the high resolution contour.

PROGRAM VERIFY
REAL ERR(128,128),ERR2(128,128)
REAL NERR(128,128)
INTEGER BINERR(128,128),BIIERR(128,128)
INTEGER CHR(128,128),CHR2(128,128)
INTEGER INX(256),IJX(256),IN(256),IJ(256)
REAL BTA(256),B2A(256),XERR(4096,2)
CHARACTER*15 NAME,NAME2

COMMON /SPACE/XSP,YSP,XSP2,YSP2
COMMON /INITIAL/XINIT,YINIT,XINIT2,YINIT2
COMMON /INT/XMIN,XMAX,YMIN,YMAX
COMMON /DATA/ERR(128,128),ERR2(128,128)

USER INTERACTION
WRITE(*,*)'ENTER INPUT FILENAME'
READ(*,5)NAME
FORMAT(A)

WRITE(*,*)'ENTER INPUT FILENAME (FILE TO BE INTERPOLATED)'
READ(*,5)NAME2

WRITE(*,*)'ENTER SPACING ALONG X-DIRECTION'
READ(*,*)XSP
WRITE(*,*)'ENTER SPACING ALONG Y-DIRECTION'
READ(*,*)YSP
WRITE(*,*)'ENTER THRESHOLD'
READ(*,*)THRESH
WRITE(*,*)'ENTER STARTING VALUE IN X'
READ(*,*)XINIT

WRITE(*,*)'ENTER STARTING VALUE IN Y'
READ(*,*)YINIT

WRITE(*,*)'ENTER SPACING ALONG X-DIRECTION FOR ',NAME2
READ(*,*)XSP2

WRITE(*,*)'ENTER SPACING ALONG Y-DIRECTION FOR ',NAME2
READ(*,*)YSP2

WRITE(*,*)'ENTER STARTING VALUE IN X FOR ',NAME2
READ(*,*)XINIT2
WRITE(*,*)'ENTER STARTING VALUE IN Y ',NAME2
READ(*,*)YINIT2
WRITE(*,*)'ENTER X-RANGE'
WRITE(*,*)' MINIMUM X-VALUE'
READ(*,*)XMIN
WRITE(*,*)' MAXIMUM X-VALUE'
READ(*,*)XMAX
WRITE(*,*)'ENTER Y-RANGE'
WRITE(*,*)' MINIMUM Y-VALUE'
READ(*,*)YMIN
WRITE(*,*)' MAXIMUM Y-VALUE'
READ(*,*)YMAX

C OPEN INPUT AND OUTPUT FILES
OPEN(UNIT=12,IOSTAT=IOS,FILE=NAME,STATUS='OLD')
OPEN(UNIT=30,IOSTAT=IOS,FILE=NAME2,STATUS='OLD')
OPEN(UNIT=21,IOSTAT=IOS,FILE='OA',STATUS='NEW')
OPEN(UNIT=31,IOSTAT=IOS,FILE='GRAD',STATUS='NEW')

C INPUT DATA FROM THE CONTOUR FILES IS READ IN HERE
READ(12,*),X,Y,E,SW,STW,FL
IF(((X.GE.XMIN).AND.(X.LE.XMAX)).AND.((Y.GE.YMIN).AND.(Y.LE.YMAX))) THEN
IX=(X-XMIN)/XSP + 2
IY=(Y-YMIN)/YSP + 2
C IX AND IY ARE INDICES AND ERR IS THE ERROR AT EACH POINT ON THE GRID
ERR(IX,IY)=E
C THE ARRAY BINERR CONTAINS THE BINARY IMAGE. HERE WE THRESHOLD
C THE ERROR BASED ON THE THRESHOLD VALUE - THRESH.
IF(E.GE.THRESH) THEN
BINERR(IX,IY)=1
ELSE
BINERR(IX,IY)=0
END IF
END IF
GO TO 10

CONTINUE
C READ LOW RESOLUTION DATA FILE
READ(30,*),X,Y,E,SW,STW,FL
IF(((X.GE.XMIN).AND.(X.LE.XMAX)).AND.((Y.GE.YMIN).AND.(Y.LE.YMAX))) THEN
IX2=(X-XMIN)/XSP2 + 2
IY2=(Y-YMIN)/YSP2 + 2
ERR2(IX2,IY2)=E
END IF
GO TO 31
CONTINUE
C THE FOLLOWING ROUTINE IS USED TO ACCOMPLISH THE TWO DIMENSIONAL
C BILINEAR INTERPOLATION.
CALL INTERPOL(IX,IY,IX2,IY2,NERR)
C
C THE INTERPOLATED ARRAY IS THRESHOLDED TO OBTAIN THE BINARY CONTOURS
C BY USING THE CONTOUR LEVEL TO BE EXAMINED AS THE THRESHOLD.
DO 50 I=1,IX
DO 60 J=1,IY
IF(NERR(I,J).GE.THRESH) THEN
   BI2ERR(I,J)=1
ELSE
   BI2ERR(I,J)=0
END IF
60 CONTINUE
50 CONTINUE
CALL PRELEDGE(BINERR,CHR,IX,IY)
CALL SOBEL(BINERR,CHR,IX,IY)
CALL PRELEDGE(BI2ERR,CHR2,IX,IY)
CALL SOBEL(BI2ERR,CHR2,IX,IY)
CALL SIMPOSE(CHR,CHR2,IX,IY)
CALL EDIPOSE(CHR2,IX,IY,XERR,JCOUNT)
JCOUNT=JCOUNT+1
XERR(JCOUNT,1)=300.0
XERR(JCOUNT,2)=300.0
REWIND(31)
READ(31,*)NAME2
ICOUNT=0
363 ICOUNT=ICOUNT+1
READ(31,19)INX(ICOUNT),IJX(ICOUNT),GX,BX,BY,B2A(ICOUNT)
19 FORMAT(I3,2X,I3,2X,4(F9.3,2X))
IF(B2A(ICOUNT).NE.-1.111) GOTO 363
DO 565 I=1,ICOUNT-1
DISTEMP=3000.0
DISTEMP2=3000.0
CCT=0.0
IF(B2A(I).NE.90.0) THEN
   SLOPE1=TAN(B2A(I)*ACOS(-1.0)/180.0)
END IF
DO 567 J=1,JCOUNT-1
IDX=INX(I)-XERR(J,1)
IDY=IJX(I)-XERR(J,2)
DIST=SQRT(FLOAT(IDY**2+IDX**2))
IF(DISTEMP2.GE.DIST) THEN
   DISTEMP2=DIST
   DX1=XERR(J,1)
   DY1=XERR(J,2)
   BA1=BTA(J)
END IF
IF(IDX.NE.0) THEN
   TEMPAN=ATAN(FLOAT(IDY)/FLOAT(IDX))
   TEMPAN=TEMPAN*180.0/(ACOS(-1.0))
   IF(IDX.GT.0) THEN
      TEMPAN=180.0+TEMPAN
   END IF
   ELSE
      IF(IDY.LT.0) THEN
         TEMPAN=90.
END IF
IF(IDY.GT.0) THEN
  TEMPAN=270.
END IF
IF(IDY.EQ.0) THEN
  TEMPAN=B2A(I)
END IF
END IF
  CCT=CCT+1.0
ENDIF
DISTMP=DIST
DX=XERR(J,1)
DY=XERR(J,2)
BA=BTA(J)
GO TO 567
ENDIF
ELSE
IDX2=INX(I)-XERR(J+1,1)
IDY2=IJX(I)-XERR(J+1,2)
IF(IDX2.NE.0)THEN
  TEMPAN2=ATAN(FLOAT(IDY2)/FLOAT(IDX2))
  TEMPAN2=TEMPAN2*180.0/(ACOS(-1.0))
  IF(IDX2.GT.0)
    TEMPAN2=180.0+TEMPAN2
  END IF
ELSE
  IF(IDY2.NE.0)
    TEMPAN2=90.
  END IF
  IF(IDY2.GT.0)
    TEMPAN2=270.
  END IF
  IF(IDY2.EQ.0)
    TEMPAN2=B2A(I)
  END IF
ENDIF
END IF
SUB1=B2A(I)-TEMPAN
SUB2=B2A(I)-TEMPAN2
PROD=SUB1*SUB2
PROD2=SUB3*SUB4
IF((PROD.LT.0.0).OR.(PROD2.LT.0.0)) THEN
  DIFFX1=(XERR(J,1)-XERR(J+1,1))
  DIFFY1=(XERR(J,2)-XERR(J+1,2))
  IF(DIFFX1.EQ.0.0)THEN
    XED=XERR(J,1)
    YED=SLOPE1*(X-XED) + IJX(I)
  ELSE
    SLOPE2=DIFFY1/DIFFX1
  END IF
ENDIF
XED = (-1)*INX(I)*SLOPE1 + IJX(I)-XERR(J,2) + SLOPE2*XERR(J,1)
IF(SLOPE1.NE.SLOPE2) THEN
  XED=XED/(SLOPE2-SLOPE1)
  YED=SLOPE2*(XED-XERR(J,1)) + XERR(J,2)
END IF
END IF

DIST=SQRT(((INX(I)-XED)**2 + (IJX(I)-YED)**2)
CCT=CCT+1
IF(DISTEMP.GE.DIST) THEN
  DX=XED
  DY=YED
  BA=B2A(I)
  DISTEMP=DIST
  END IF
END IF

567 CONTINUE
IF((CCT.EQ.0.0).OR.(DISTEMP.GT.12.0)) THEN
  DISTEMP=DISTEMP2
  DX=DX1
  DY=DY1
  BA=BA1
END IF
IF(DISTEMP.NE.3000.) THEN
  WRITE(21,9)INX(I),IJX(I),B2A(I),DX,DY,BA,DISTEMP
END IF
9 FORMAT(I3,2X,I3,2X,5(F8.3,2X),2X,F6.2)
565 CONTINUE
102 CONTINUE
333 CONTINUE
CLOSE(21)
CLOSE(12)
STOP
END

SUBROUTINE ROUND(RX,IX)
CRX = RX - IFIX(RX)
IF(CRX.LT. 0.5) THEN
  IX=IFIX(RX)
ELSE
  IX=IFIX(RX) + 1
END IF
RETURN
END

SUBROUTINE PRELEDGE(BR,BR2,IX,IY)
INTEGER BR(128,128),BR2(128,128)
WRITE(21,*)"IMAGE"
DO 10 I = 1,IX
  DO 20 J = 1,IY
IF(BR(I,J).EQ.1) THEN
  IF((I.EQ.1).OR.(I.EQ.IX).OR.(J.EQ.1).OR.(J.EQ.IY)) THEN
    GOTO 30
  ELSE
    IF((BR(I-1,J)*BR(I,J+1)*BR(I-1,J)*BR(I+1,J)).EQ. 0) THEN
      GOTO 30
    END IF
  END IF
END IF
GOTO 40
30   BR2(I,J) = 1
40   CONTINUE
20   CONTINUE
C   WRITE(21,*) (BR2(I,J), J=1,IY)
10   CONTINUE
RETURN
END

SUBROUTINE SOBEL(BERR,CHR,IX,IY)
INTEGER BERR(128,128),CHR(128,128)
WRITE(31,*)'EDGE GRADIENT'
DO 10 I=1,IX
  DO 20 J = 1, IY
    IF(CHR(I,J).EQ.1) THEN
      IF(J.NE.1) THEN
        BX=(BERR(I+1,J+1)-BERR(I-1,J+1)+2*(BERR(I+1,J)-BERR(I-1,J)) + < BERR(I+1,J-1)-BERR(I-1,J-1))
      ELSE
        BX=(BERR(I+1,J+1)-BERR(I-1,J+1)+2*(BERR(I+1,J)-BERR(I-1,J)) + < BERR(I-1,J+1)-BERR(I-1,J-1))
      END IF
      BY=(BERR(I+1,J+1)-BERR(I+1,J-1)+2*(BERR(I,J+1)-BERR(I,J-1)) + < BERR(I-1,J+1)-BERR(I-1,J-1))
    ELSE
      BX=(BERR(I+1,J+1)-BERR(I-1,J+1)+2*(BERR(I+1,J)-BERR(I-1,J)) + < BERR(I+1,J+1)+2*(BERR(I,J+1)) + < BERR(I-1,J+1))
    END IF
    G=SNRT(BX**2 + BY**2)
    IF(BX.NE.0) THEN
      BTA=(180*ATAN(BY/BX))/(ACOS(-1.0))
    ELSE
      IF(BY.NE.0) THEN
        BTA=(180*ASIN(BY/G))/(ACOS(-1.0))
      ELSE
        BTA=0.0
      END IF
    END IF
  END IF
WRITE(31,9)I,J,G,BX,BY,BTA
9   FORMAT(I3,2X,I3,2X,4(F9.3,2X))
END IF
20   CONTINUE
10   BTA=-1.111
SUBROUTINE SIMPOSE(BINERR,B12ERR,IX,ITY)
INTEGER BINERR(128,128),B12ERR(128,128)
INTEGER X(128)
DO 10 I = 1,IX
  DO 20 J = 1,ITY
    IF(B12ERR(I,J).EQ.1) THEN
      X(J) = 2
    ELSE
      X(J) = 0
    END IF
  CONTINUE
WRITE(21,*)(BINERR(I,J) + X(J),J = 1,ITY)
10 CONTINUE
RETURN
END

SUBROUTINE INTERPOL(IX1,IY1,IX2,IY2,NERR)
C INTERPOLATES THE OPTIMAL GRID SPACING CONTOUR TO THE
C HIGH RESOLUTION CONTOUR.
REAL NERR(128,128)
COMMON /DATA/ERR(128,128),ERR2(128,128)
COMMON /SPACE/XSP,YSP,XSP2,YSP2
COMMON /INITIAL/XINIT,YINIT,XINIT2,YINIT2
COMMON /INT/XMIN,XMAX,YMIN,YMAX
C
DO 10 I = 1,IX1
  DO 10 J = 1,IY1
    X = (I-1)*XSP + XINIT
    Y = (J-1)*YSP + YINIT
    RJ = (X-XMIN)/XSP2
    RJ2 = (Y-YMIN)/YSP2
    CH1 = RJ-IFIX(RJ)
    CH2 = RJ2-IFIX(RJ2)
    INX = IFIX((X-XMIN)/XSP2) + 1
    INY = IFIX((Y-YMIN)/YSP2) + 1
    IF((CH1.EQ.0.0).AND.(CH2.EQ.0.0)) THEN
      NERR(I,J) = ERR2(INX,INY)
    ELSE
      X2 = (INX-1)*XSP2 + XMIN
      Y2 = (INY-1)*YSP2 + YMIN
      DELX = (X-X2)/XSP2
      DELY = (Y-Y2)/YSP2
      DX = 1-DELX
SUBROUTINE INTERP2(ERR, NERR)
C THIS ROUTINE INTERPOLATES THE TWO CONTOURS ON TO A VERY FINE GRID
C
REAL ERR(128,128), NERR(128,128)
COMMON /SPACE/XSP, YSP, XSP2, YSP2
COMMON /INITIAL/XINIT, YINIT, XINIT2, YINIT2
COMMON /INT/XMIN, XMAX, YMIN, YMAX

DO 10 I=IFIX(XMIN), IFIX(XMAX-XSP), 10
   DO 10 J=IFIX(YMIN), IFIX(YMAX-YSP), 10
      RJ=(I-XMIN)/XSP + 1.0
      RJ2=(J-YMIN)/YSP + 1.0
      CH1=RJ-IFIX(RJ)
      CH2=RJ2-IFIX(RJ2)
      INX = IFIX((I-XMIN)/10.0) + 1
      INY = IFIX((J-YMIN)/10.0) + 1
      IRJ=RJ
      IRJ2=RJ2
      IF((CH1.EQ.0.0).AND.(CH2.EQ.0.0)) THEN
         NERR(INX,INY)=ERR(IRJ,IRJ2)
      ELSE
         X2=(IRJ-1)*XSP + XMIN
         Y2=(IRJ2-1)*YSP + YMIN
         DELX = (I-X2)/XSP
         DELY = (J-Y2)/YSP
         DX=1-DELX
         DY=1-DELY
         NERR(INX,INY)=DX*DY*ERR(IRJ,IRJ2) + DX*DELY*ERR(IRJ,IRJ2+1) +
         < DELX*DY*ERR(IRJ+1,IRJ2) + DELX*DELY*ERR(IRJ+1,IRJ2+1)
      END IF
10 CONTINUE
RETURN
END

SUBROUTINE MINSTART(NERR, IX, IY, XST, YST)
INTEGER NERR(128,128)
   COMMON /SPACE/XSP, YSP, XSP2, YSP2
   COMMON /INITIAL/XINIT, YINIT, XINIT2, YINIT2
   COMMON /INT/XMIN, XMAX, YMIN, YMAX
   DO 10 I=1,IX
      DO 20 J=1, IY
IF(NERR(I,J).EQ.1)THEN
  XST = FLOAT(I)
  YST = FLOAT(J)
  GO TO 30
END IF

20 CONTINUE
10 CONTINUE
  XST=0.0
  YST=0.0
30 CONTINUE
RETURN
END

SUBROUTINE EDFOLLOW(CHR,IX,IY,XERR,ICTR)
  INTEGER CHR(128,128)
  REAL XERR(4096,2)
  CHARACTER CHRAC
  COMMON /SPACE/XSP,XSP2,YSP2
  COMMON /INITIAL/XINIT,XINIT2,YINIT,YINIT2
  COMMON /INT/XMIN,XMAX,YMIN,YMAX
  DIMENSION CHRAC(128,128)
  ICTR=0
12 CALL MINSTART(CHR,IX,IY,XST,YST)
  DX=0
  DY=0
  XMK=0
  YMK=0
  DIST=0
  IF(XST.EQ.0.0 .AND. YST.EQ.0.0) THEN
    GOTO 210
  ELSE
    I=XST
    J=YST
  END IF
  CHRAC(I,J)='S'
10  ICOUNT = 0
20  IF(XMK.EQ.0.0 .AND. YMK.EQ.0.0) THEN
    XD=-1
    YD=-1
  ELSE
    DX=I-XMK
    DY=J-YMK
  END IF
  IF(DX.EQ.0.0) THEN
    IF(DY.EQ.0.0) THEN
      IF(DY.EQ.-1) THEN
    XD=-1
    YD=-1
    ELSE
      IF(DY.EQ.-1) THEN
XD=1
YD=1
END IF
END IF
END IF
IF(DX.EQ.1.0) THEN
IF(DY.EQ.1.0) THEN
XD=-1
YD=0
ELSE IF(DY.EQ.-1.0) THEN
XD=0
YD=1
ELSE IF(DY.EQ.0.0) THEN
XD=-1
YD=1
END IF
ENDIF
END IF
IF(DX.EQ.-1.0) THEN
IF(DY.EQ.-1.0) THEN
XD=1
YM=0
ELSE IF(DY.EQ.0.0) THEN
XD=1
YD=-1
ELSE IF(DY.EQ.1.0) THEN
XD=0
YD=-1.0
END IF
ENDIF
ICOUNT=ICOUNT+1
XMK=I+XD
YMK=J+YD
IF((CHR(I,J).EQ.1).AND.(ICOUNT.EQ.1)) THEN
ICTR=ICTR+1
XERR(ICTR,1)=I
XERR(ICTR,2)=J
END IF
ENDIF
IF(CHR(XMK,YMK).EQ.1) THEN
IF(CHRAC(XMK,YMK).NE.'S') THEN
CHRAC(XMK,YMK)='I'
ENDIF
ELSE
IF(ICOUNT.EQ.8) THEN
GO TO 70
ELSE
GO TO 20
ENDIF
ENDIF
XPREV=I
YPREV=J
I=XMK
J=YMKE
XMK=XPREV
YMK=YPREV
IF(CHRAC(I,J).EQ.'S') THEN
  WRITE(21,*)'CURVE ENDS'
  GO TO 205
END IF
GO TO 10
70 WRITE(*,*)'RM POINT',I,J,XMK,YMK
205 CALL ZEROOUT(CHR,CHRAC)
GOTO 12
210 CONTINUE
RETURN
END

SUBROUTINE ZEROOUT(CHR,CHRAC)
INTEGER CHR(128,128)
CHARACTER CHARAC
DIMENSION CHARAC(128,128)
DO 10 I=1,128
DO 20 J=1,128
IF(CHRAC(I,J).EQ.'I' .OR. CHARAC(I,J).EQ.'S') THEN
  CHR(I,J)=0.0
END IF
20 CONTINUE
10 CONTINUE
RETURN
END
APPENDIX B.

SIMULATION RESULTS

This section includes the computer simulation results obtained using test data generated which are not collected from a physical system. First, the grid size estimates for combinations of runway lengths and obstacle type are described in Appendix B.1. Various contour plots obtained for the parallel orientation of a B-747 obstacle are presented in Appendix B.2. Determination of grid spacing for the contours in Appendix B.2 involves spatial frequency analysis. The results of this analysis are given in Appendix B.3. Various plots provided in this section include error distribution plots for different bandwidths of the filter and 2-D Fourier spectrum of PFE contours for different decision heights. Results of the verification process are given in Appendix B.4.
APPENDIX B.1 RESULTS OF GRID SIZE DETERMINATION

Grid sizes were determined for combinations of different runway lengths (5000 ft, 9000 ft, and 12000 ft) and obstacle types (B-747 and B-727 aircrafts). These results were presented in Table I of Section VI. Figures B.1.1 - B.1.3 illustrate the grid sizes obtained for a B-747 obstacle for runway lengths of 5000 ft, 9000 ft and 12000 ft respectively. Figures B.1.4 - B.1.6 demonstrate the grid sizes obtained for B-727 obstacle for runway lengths of 5000 ft, 9000 ft and 12000 ft.
Figure B.1.1

Grid Area for B-747 for 50000ft runway

Distance From Antenna (in Feet)

Distance From Centerline (in Feet)

x - scatterer location
Grid Area for B-747 for 12000ft runway

- scatterer location

Distance From Centerline (in Feet)

Distance From Antenna (in Feet)

Figure B.1.3
Grid Area for B-727 for 5000ft runway

Transponder location : -1000.0, 0.0, 0.0
5 nmi 3 degree centerline approach

Figure B.1.4
APPENDIX B.2 CONTOUR PLOTS

In this section, error contours generated for a B-747 scatterer are provided. This scenario involved a 3°, 5 nautical miles centerline approach for a runway of length 9000 ft. The scatterer was oriented parallel to the runway. Figures B.2.1 - B.2.12 are the low resolution contour plots obtained for various decision heights. The maximum value of PFE seen in these plots is about 7ft. The maximum value from previous studies and flight measurements is close to 60 feet. This shows that the DME/P portion of the MLS math model needs to be validated. Figure B.2.13 - B.2.24 are the high resolution contours obtained in the local window for the low resolution contours. These plots are used as sample data to demonstrate the methodology for optimal grid spacing determination.
DME/P SUBSYSTEM -1000,0,0,0,9,00
PFE CONTOURS FOR 8-747 (900FT DH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFC900.PLT
AEC MLS MODELING

Figure B.2.1
DME/P SUBSYSTEM: -1000.0, 0.0, 9.00
PFE CONTOURS FOR B-747 (800FT DH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFG800.PLT
AEC MLS MODELING

Figure B.2.2
Figure B.2.3

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT

PLOT STORED ON DISK FILE: PFC700.PLT

REC M.S. MODELING

DME/P SUBSYSTEM: 0.0, 0.9, 0.0
PFE CONTOURS FOR A-KM ANGLE: 0
3 DEGREE C.C.L.
APPROACH: 9000 FT. RUNWAY

DISTANCE FROM RUNWAY CENTERLINE IN FEET

DISTANCE IN FRONT OF THE ANTENNA IN FEET

0.5 1.5 2.5 3.5 4.5

8.5 7.5 6.5 5.5 4.5

0.5 1.5 2.5 3.5 4.5

9.5 x 10^3
DME/P SUBSYSTEM -1000.0,0.0,9.00
PFE CONTOURS FOR 8-747 (600FT OH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFE600.PLT
AEC MLS MODELING

Figure B.2.4
Figure B.2.5

Distance from runway centerline in feet

Note: Contour lines are in increments of 2.5 ft.

Plot stored on disk file: PFE5000.PLT

REC ML5 Modeling
DME/P SUBSYSTEM: -1000.0, 0.0, 0.0, 9.00
PFE CONTOURS FOR B-747 (350FT DH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

DISTANCE FROM RUNWAY CENTERLINE IN FEET

DISTANCE IN FRONT OF THE ANTENNA IN FEET

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFE350.PLT
AEC MLS MODELING

Figure B.2.7
DME/P SUBSYSTEM: -1000.0,0.0,0.0,9.00
PFE CONTOURS FOR B-747 (300FT OH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFE300.PLI
ACC MLS MODELING

Figure B.2.8
DME/P SUBSYSTEM -1000.0,0.0,9.00
PPE CONTOURS FOR B-747 (250FT DH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFC250.PLT
AEC MLS MODELING

Figure B.2.9
DME/P SUBSYSTEM -1000.0,0.0,9.00
PTE CONTOURS FOR B-747 (CAT 1D1)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFEIDH.PLT
RCE MLS MODELING

Figure B.2.10
DME/P SUBSYSTEM -1000.0,0.0,9.00
PPE CONTOURS FOR B-747 (500FT DH)
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFE150.FLT
REC MLS MODELING

Figure B.2.11
DME/P SUBSYSTEM -1000.0,0.0,9.00
PFE CONTOURS FOR B-747 (CAT IIIA)
AIRCRAFT ROTATION ANGLE: 0°
3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 2.5 FT
PLOT STORED ON DISK FILE: PFE110M.PLT 3 DEGREE C.C.L. APPROACH - 9000 FT. RUNWAY
ACC MLS MODELING

Figure B.2.12
DME/P SUBSYSTEM -1000.0,0.0,9.00
700 FOOT DH - PFD CONTOURS FOR B-747
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.L. APPROACH - 9000 FT. RUNWAY

DISTANCE FROM RUNWAY CENTERLINE IN FEET

500.0 700.0 900.0 1100.0 1300.0 1500.0 1700.0 1900.0 2100.0

DISTANCE IN FRONT OF THE ANTENNA IN FEET

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 3.0 FT
PLOT STORED ON DISK FILE: DCON3.PLT
REC MLS MODELING

Figure B.2.15
DME/P SUBSYSTEM -1000.0,0.0,9.00
600 FOOT DH - PFE CONTOURS FOR B-747
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 3.0 FT
PLOT STORED ON DISK FILE: DCON4.PLT
ACC MLS MODELING

Figure B.2.16
DME/P SUBSYSTEM  -1000.0,0.0,9.00
400 FOOT DH - PFE CONTOURS FOR B-747
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.L. APPROACH - 9000 FT. RUNWAY

DISTANCE FROM RUNWAY CENTERLINE IN FEET

DISTANCE IN FRONT OF THE ANTENNA IN FEET

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 3.0 FT
PLOT STORED ON DISK FILE: DCON6.PLT
ACC MLS MODELING

Figure B.2.18
DME/P SUBSYSTEM -1000.0, 0.0, 0.9, 0.0
350 FOOT DN - PFC CONTOURS FOR B-747
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 3.0 FT
PLOT STORED ON DISK FILE: DCON7.PLT
AEC MLS MODELING

Figure B.2.19
DME/P SUBSYSTEM -1000.0,0.0,9.00
CAT-I DH - PFE CONTOURS FOR B-747
AIRCRAFT ROTATION ANGLE: 0
3 DEGREE C.L. APPROACH - 9000 FT. RUNWAY

NOTE: CONTOUR LINES ARE IN INCREMENTS OF 3.0 FT
PLOT STORED ON DISK FILE: DCON10.PLT
AEC MLS MODELING

Figure B.2.22
APPENDIX B.3

RESULTS OF GRID SPACING DETERMINATION

This section describes results obtained for the grid spacing determination of the contours shown in Appendix B.2. The spatial frequency analysis. In this section the results of this analysis are presented. Spectral plots for each of the decision heights are given in Figures B.3.1 - B.3.12. It is noted from the 2-D Fourier spectrum that the frequency content of the contours is distributed mainly in the low frequency range. Further, distribution plots (Figures B.3.13 - B.3.44) of reconstruction error were also obtained for different bandwidths of the 2-D filter and error contour values.
Fourier spectrum for 900ft DH
Fourier spectrum for 800ft DH
Fourier spectrum for 700ft DH

Figure B.3.3
Fourier spectrum for 600ft DH

Figure B.3.4
Fourier spectrum for 500ft DH

Figure B.3.5
Fourier spectrum for 400ft DH

Figure B.3.6
Fourier spectrum for 300ft DH
Fourier spectrum for 250ft DH

Figure B.3.9
Fourier spectrum for CAT-I DH

Figure B.3.10
Fourier spectrum for CAT-II DH

Figure B.3.12
Histogram of reconstruction error for 900 ft DH contour

3ft PFE contour
Bandwidth = 0.011 cycle/ft
Total number of points = 227
Maximum percentage reconstruction error = 0.447
Mean reconstruction error = 3.139 ft
Maximum deformation = 88.4 ft
Mean deformation = 25.32 ft

Figure B.3.13
Histogram of reconstruction error for 900 ft DH contour

3 ft PFE contour
Bandwidth = 0.01 cycle/ft
Total number of points = 227
Maximum percentage reconstruction error = 0.466
Mean reconstruction error = 3.362 ft
Maximum deformation = 103.1 ft
Mean deformation = 31.45 ft

Figure B.3.14

Pointwise Reconstruction Error (ft)
4ft PFE contour
Bandwidth = 0.02 cycle/ft
Total number of points = 119
Maximum percentage reconstruction error = 0.357
Mean reconstruction error = 2.313 ft
Maximum deformation = 90.15 ft
Mean deformation = 16.28 ft

Figure B.3.15
Histogram of reconstruction error for 900 ft DH contour

- Bandwidth = 0.019 cycle/ft
- Total number of points = 119
- Maximum percentage reconstruction error = 0.364
- Mean reconstruction error = 2.369 ft
- Maximum deformation = 111.8 ft
- Mean deformation = 19.94 ft

Figure B.3.16
Histogram of reconstruction error for 900 ft DH contour

- 5ft PFE contour
- Bandwidth = 0.011 cycle/ft
- Total number of points = 52
- Maximum percentage reconstruction error = 0.437
- Mean reconstruction error = 3.529 ft
- Maximum deformation = 55.9 ft
- Mean deformation = 24.07 ft

Figure B.3.17
Histogram of reconstruction error for 900 ft DH contour

- 5 ft PFE contour
- Bandwidth = 0.011 cycle/ft
- Total number of points = 52
- Maximum percentage reconstruction error = 0.447
- Mean reconstruction error = 3.649 ft
- Maximum deformation = 111.8 ft
- Mean deformation = 30.04 ft

Figure B.3.18
Histogram of reconstruction error for 900 ft DH contour

6ft PFE contour
Bandwidth = 0.011 cycle/ft
Total number of points = 19
Maximum percentage reconstruction error = 0.343
Mean reconstruction error = 3.087 ft
Maximum deformation = 50 ft
Mean deformation = 13.88 ft

Pointwise Reconstruction Error (ft)

Figure B.3.19
Histogram of reconstruction error for 900 ft DH contour

6ft PFE contour
Bandwidth = 0.018 cycle/ft
Total number of points = 19
Maximum percentage reconstruction error = 0.349
Mean reconstruction error = 3.169 ft
Maximum deformation = 103.1 ft
Mean deformation = 30.33 ft

Figure B.3.20
Histogram of Deformation for 900ft DH contour

3ft PFE contour
Bandwidth = 0.01 cycle/ft
Total number of points = 72
Maximum deformation = 88.4 ft
Mean deformation = 25.86 ft

Figure B.3.21
Figure B.3.22

Histogram of Deformation for 900ft DH contour

- 3ft PFE contour
- Bandwidth = 0.01 cycle/ft
- Total number of points = 72
- Maximum deformation = 103.1 ft
- Mean deformation = 31.45 ft
Histogram of Deformation for 900ft DH contour

- 4ft PFE contour
- Bandwidth = 0.02 cycle/ft
- Total number of points = 46
- Maximum deformation = 90.15 ft
- Mean deformation = 16.28 ft

Figure B.3.23
Histogram of Deformation for 900ft DH contour

4ft PFE contour

Bandwidth = 0.019 cycle/ft

Total number of points = 46

Maximum deformation = 111.8 ft

Mean deformation = 19.94 ft

Figure B.3.24
Histagram of Deformation for 900 ft DH contour

- 6 ft PPE contour
- Bandwidth = 0.019 cycle/ft
- Total number of points = 18
- Maximum deformation = 50 ft
- Mean deformation = 13.88 ft

Figure 3.27
Histogram of Deformation for 900ft DH contour

6ft PFE contour
Bandwidth = 0.018 cycle/ft
Total number of points = 18
Maximum deformation = 103.1 ft
Mean deformation = 30.33 ft

Figure B.3.28
Figure B.3.29

Histogram of reconstruction error for CAT-II DH contour

3ft PFE contour
Bandwidth = 0.01 cycle/ft
Total number of points = 373
Maximum percentage reconstruction error = 0.414
Mean reconstruction error = 2.119 ft
Maximum deformation = 79.05 ft
Mean deformation = 28.85 ft
Histogram of reconstruction error for CAT-II DH contour

3ft PFE contour
Bandwidth = 0.009 cycle/ft
Total number of points = 373
Maximum percentage reconstruction error = 0.433
Mean reconstruction error = 2.178 ft
Maximum deformation = 103.1 ft
Mean deformation = 30.57 ft

Figure B.3.30
Histogram of reconstruction error for CAT-II DH contour

4ft PFE contour
Bandwidth = 0.018 cycle/ft
Total number of points = 139
Maximum percentage reconstruction error = 0.243
Mean reconstruction error = 1.542 ft
Maximum deformation = 111.8 ft
Mean deformation = 17.62 ft

Figure B.3.32
Histogram of reconstruction error for CAT-II DH contour

5 Hz PFE contour
Bandwidth = 0.01 cycle/ft
Total number of points = 53

Maximum percentage reconstruction error = 0.4
Mean reconstruction error = 2.832 ft
Maximum deformation = 55.9 ft
Mean deformation = 23.14 ft

Pointwise Reconstruction Error (ft)

Figure B.3.33
Histogram of reconstruction error for CAT-II DH contour

5 ft PFE contour
Bandwidth = 0.007 cycle/ft
Total number of points = 53
Maximum percentage reconstruction error = 0.445
Mean reconstruction error = 3.5 ft
Maximum deformation = 111.8 ft
Mean deformation = 40.25 ft
Pointwise Reconstruction Error (ft)

Histogram of reconstruction error for CAT-II DH contour

- 6 ft PFE contour
- Bandwidth = 0.016 cycle/ft
- Total number of points = 19
- Maximum percentage reconstruction error = 0.267
- Mean reconstruction error = 2.166 ft
- Maximum deformation = 81.83 ft
- Mean deformation = 30.12 ft

Figure B.3.35
Histogram of reconstruction error for CAT-II DH contour

6ft PFE contour
Bandwidth = 0.016 cycle/ft
Total number of points = 19
Maximum percentage reconstruction error = 0.284
Mean reconstruction error = 2.254 ft
Maximum deformation = 147.9 ft
Mean deformation = 36.32 ft

Figure B.3.36
Histogram of Deformation for CAT-II DH contour

3ft PFE contour
Bandwidth = 0.01 cycle/ft
Total number of points = 93
Maximum deformation = 79.05 ft
Mean deformation = 28.85 ft

Figure B.3.37
Histogram of Deformation for CAT-II DH contour

3ft PFE contour
Bandwidth = 0.009 cycle/ft
Total number of points = 93
Maximum deformation = 103.1 ft
Mean deformation = 30.57 ft

Figure B.3.38
Histogram of Deformation for CAT-II DH contour

4ft PFE contour
Bandwidth = 0.02 cycle/ft
Total number of points = 49
Maximum deformation = 90.15 ft
Mean deformation = 12.62 ft

Figure B.3.39
Histogram of Deformation for CAT-II DH contour

4ft PE contour

Bandwidth = 0.018 cycle/ft

Total number of points = 49

Maximum deformation = 111.8 ft

Mean deformation = 17.62 ft

Figure B.3.40
Histogram of Deformation for CAT-II DH contour

5ft PFE contour
Bandwidth = 0.01 cycle/ft
Total number of points = 30
Maximum deformation = 55.9 ft
Mean deformation = 23.14 ft

Deformation (ft)
Figure B.3.41

Number of Points

16 14 12 10 8 6 4 2 0
0 20 40 60 80 100 120 140 160
Histogram of Deformation for CAT-II DH contour

- SH PPE contour
- Bandwidth = 0.007 cycle/ft
- Total number of points = 30
- Maximum deformation = 111.8 ft
- Mean deformation = 40.25 ft

Figure B.3.42

Number of Points

Deformation (ft)
 Histogram of Deformation for CAT-II DH contour

6ft PFE contour
Bandwidth = 0.016 cycle/ft
Total number of points = 18
Maximum deformation = 81.83 ft
Mean deformation = 30.12 ft

Number of Points

Deformation (ft)

Figure B.3.43
Histogram of Deformation for CAT-II DH contour

6ft PFE contour
Bandwidth = 0.015 cycle/ft
Total number of points = 18
Maximum deformation = 147.9 ft
Mean deformation = 36.32 ft

Figure B.3.44
Appendix B.4

Results of Verification of Grid Spacing

In this section, the results of the verification process are given. This verification process was conducted by comparing contours with grid spacing of 50ft and 75ft with the high resolution contours. Figures B.4.1-B.4.8 are the deformation obtained for the 3, 4, 5 and 6ft contours.
Histogram of deformation for 900 ft DH

3 ft contour
Grid spacing = 50 ft

Figure B.4.1
Histogram of deformation for 900 ft DH

4 ft contour
Grid spacing = 50 ft

Figure B.4.3
Histogram of deformation for 900 ft DH

4 ft contour

Grid spacing = 75 ft

Deformation (ft)

Number of points

Figure B.4.4
Histogram of deformation for 900 ft DH

5 ft contour

Grid spacing = 75 ft

Figure B.4.6
APPENDIX C.

PARALLEL IMPLEMENTATION SOFTWARE

In this section, the various modules of the parallel implementation are presented. The various modules in this implementation are the parent process, the child process and the parallel implementation of the 2-D discrete Fourier transform. The parent process (SCRPT) given in Appendix C.1 prepares the input and output files for the parallel implementation, spawns the process, then sorts the results into files for different decision heights. The parent process is similar in implementation to SPLMS exec described in Appendix A.2.1. The child process (PARBMLS) described in Appendix C.2 is a combined program of the transmitter model, DME/P receiver model, and the windowing routine. The parallel implementation of the 2-D DFT (PARFFT) is presented in Appendix C.3.
C.1 PARENT PROCESS

The parent process given here consists of three parts. The main section called Scrpt calls the other sections to accomplish the function of the parent. Scrpt calls the program Edit to prepare 16 input files required for one execution of the simulation package on the parallel processor. It also calls the program Sort to sort out the results of the implementation onto various disk files. The program scrpt also takes care of the file manipulations like creating, renaming and erasing files. The listings for these three sections are given here.
This shell script is the main controlling unit of the parallel implementation. Sets up input files, spawns and terminates process as required for contour generation.

%Variable a is set to END when grid is done.
% a='begin'
echo $a
%
%mesg2 is the file which is checked continuously until the current process is completed on S-2010. The contents of mesg2 is initialized to 'begin'.
% echo begin > mesg2
%
%The edit routine prepares sections 3 and 8 of the input file. The entire input file is divided into 5 files. File 1 is called npid and contains sections 0 through 2. File 2 is SECT3 and contains section 3. File 3 is called npid2 and contains sections 4 through 7. File 4 contains section 8. File 5 called npid3 contains the remaining sections.
%edit prepares 32 files - 2 each for the 16 nodes.
% edit.exe
%Here the 16 input files are made from the output of edit.
%
for i in A B C D E F G H I J K L M N O P
  do
    cat npid SECT3$i npid2 SECT8$i npid3 > PID_$i
    echo OUTPUT > X$i
    rm SECT3$i SECT8$i
  done
%
%Here the variable a is checked to see if entire grid is done
% until test $a = END
  do
%Here the machine is acquire first
    getmc s2010 16
    b='begin'
%
The contouring package is spawned on the nodes
    echo Spawning Process
    spawn shakdme -1
%
%Here variable bis checked to see if the spawned process is completed
  until test $b = end
    do
%
%mesg2 is the file into which the corner node writes after process is terminated. This file is checked to see if process is over.
  b='cat mesg2'
  done
The data that is uploaded contains the peak PFE for all the different decision heights. These are sorted out to various files for different decision heights using the sort routine.

Remove all unwanted files

The decision height files are appended to existing files

Free S-2010

End of Process
EDIT.F
This routine prepares Sections 3 and 8 of the 16 input files for contouring.

PROGRAM EDIT
C DECLARATIONS
CHARACTER*5 FNAME
CHARACTER*6 NAME
CHARACTER*6 SNAME
CHARACTER*5 UNAME
CHARACTER*4 NR
INTEGER FLAG
C
C THE GRID LIMITS ARE ENTERED HERE.
C MAXIMUM AND MINIMUM EXTREMES OF THE GRID ARE ENTERED
C ALSO THE LENGTH OF ACFT IN X AND Y DIRECTIONS ARE ENTERED
C
XMAX=8000
YMAX=1000
XINIT=0
YINIT=0
XD=116
YD=0
C GRID SPACINGS ARE ENTERED HERE
XINC=100
YINC=100
C OPEN INPUT FILES
OPEN(7,IOSTAT=IOS,FILE='SECT3',STATUS='OLD')
OPEN(8,IOSTAT=IOS,FILE='SECT8',STATUS='OLD')
C
C READ CURRENT LOCATION OF SCATTERER
READ(7,*),INO,XTAIL,YTAIL,XCKPT,YCKPT,ALT,IAT,GCORR
READ(8,*),KNO,XVAL,YVAL,ZVAL,VEL,ANG,KAT
READ(8,*),XVAL2,YVAL2,ZVAL2
XPOS=XTAIL+XD
YPOS=YTAIL-YD
C
FNAME='SECT3'
UNAME='SECT8'
FLAG=0
C PREPARE SECTIONS 3 AND 8 BASED ON NODE NUMBER
C AND WRITE TO OUTPUT FILE. NAME OUTPUT FILE ALSO
C BASED ON NODE NUMBER. I IS THE INDEX THAT IS USED
C
AS THE NODE NUMBER.
DO 10 I=1,16
NR=CHAR(I-1+ICHAR('A'))
NAME=FNAME/NR
SNAME=UNAME/NR
10 CONTINUE
OPEN(21,IOSTAT=IOS,FILE=NAME,STATUS='NEW')
OPEN(30,IOSTAT=IOS,FILE=SNAME,STATUS='NEW')

WYPOS=YPOS+YINC
WXPOS=XPOS+XINC*(I-1)
IF(WYPOS.GT.YMAX) THEN
  WYPOS=YINIT
  WXPOS=WXPOS+XINC*16
END IF

IF(WXPOS.GT.XMAX) THEN
  FLAG=FLAG+1
ELSE
  XTAIL=WXPOS-XD
  YTAIL=WYPOS+YD
  XCKPT=WXPOS+XD
  YCKPT=WYPOS-YD
  XVAL=XTAIL
  YVAL=YTAIL
  XVAL2=XTAIL+20
  YVAL2=YTAIL
END IF

C
3 FORMAT(8X,I2,1X,F8.0,1X,F8.0,1X,F8.0,1X,F8.0,1X,F8.0,1X,F8.0,
*1X,I2,1X,F8.0)
WRITE(21,3)INO,XTAIL,YTAIL,XCKPT,YCKPT,ALT,IAT,GCORR
WRITE(30,5)KNO,XVAL,YVAL,ZVAL,VEL,ANG,KAT
5 FORMAT(8X,I2,1X,F8.0,1X,F8.0,1X,F8.0,1X,F5.0,1X,F5.0,
*1X,I2)
7 FORMAT(11X,F8.0,1X,F8.0,1X,F8.0)
WRITE(30,7)XVAL2,YVAL2,ZVAL2
IF(I.EQ.1) THEN
  CLOSE(7)
C
C ALSO WRITE NEW LOCATION OF SCATTERER FOR NODE 0 INTO SECT3
C AND SECT8.
OPEN(8,IOSTAT=IOS,FILE='SECT8',STATUS='OLD')
OPEN(7,IOSTAT=IOS,FILE='SECT3',STATUS='OLD')
WRITE(8,5)KNO,XVAL,YVAL,ZVAL,VEL,ANG,KAT
WRITE(7,7)XVAL2,YVAL2,ZVAL2
WRITE(7,3)INO,XTAIL,YTAIL,XCKPT,YCKPT,ALT,IAT,GCORR
END IF
10 CONTINUE
CLOSE(7)
CLOSE(8)
C
C CHECK IF ENTIRE GRID IS DONE.
IF(FLAG.EQ.16) THEN
  WRITE(*,*)'END'
ELSE
  WRITE(*,*)'BEGIN'
END IF
STOP
END
SORT
This program sorts out the output from nodes into various contour data based on decision heights.

PROGRAM SORT
CHARACTER*3 FNAME
CHARACTER*4 SNAME
CHARACTER*1 ENAME
CHARACTER*2 DNAME
CHARACTER*72 BUFFER
ENAME = 'X'
FNAME = 'ERR'
DO 10 I = 1,16
   SNAME = FNAME //(CHAR(ICHAR('A') + I - 1))
   OPEN(UNIT=9, IOSTAT=IOS, FILE=SNAME, STATUS='OLD')
10    CONTINUE
DO 20 J = 1,14
   NNO = J + 10
   DNAME = ENAME //(CHAR(ICHAR('A') + J - 1))
   OPEN(UNIT = NNO, IOSTAT = IOS, FILE = DNAME, STATUS = 'UNKNOWN')
20    CONTINUE
DO 30 J = 1,14
   READ(9,*) a, b, c, d, e, ik, ki
   NNO = J + 10
   WRITE(NNO,3) a, b, c, d, e, ik
3    FORMAT(F9.3,1X,F9.3,1X,F8.4,1X,F9.6,1X,F9.6,1X,F9.6,2X,I2)
30    CONTINUE
CLOSE(9)
10    CONTINUE
   DO 40 I = 1,14
   CLOSE(I+10)
40    CONTINUE
STOP
END
C.2 CHILD PROCESS

The child program is the combine version of the transmitter, receiver, and windowing routines. Intermediate files have not been eliminated here. The files to read from and write to are determined based on the logical node number, using a call to mynode() function. Further, once all nodes complete the task synchronization is achieved by informing the corner node of the completion. This is done in the gsum routine.
PARBMLST - S-2010 VERSION OF CONTOURING ROUTINE
This is the child program that is executed on the nodes of the
S-2010 message passing multicomputer system. This includes the
DME/P receiver, windowing routine and gsum routine to ensure
proper termination of the process.

PROGRAM BMLST
CHARACTER*132 BUFF
CHARACTER*80 FPCTL
CHARACTER*60 RTITLE
CHARACTER*40 IFILE, OTFILE, ORFILE, BLANK, NAME, ADDR, FFILE
CHARACTER*14 ANTYP
CHARACTER*9 RDATE
CHARACTER*8 RTIME
CHARACTER*4 IRID
CHARACTER*3 VERS
CHARACTER*4 FNAME
CHARACTER*1 INAME
CHARACTER*5 SNAME
LOGICAL STOP, NEWSEG, WSTOP, SHADOW, HUMP, ANSFL, MENSURD
LOGICAL AZPLZ, ELPLZ, DMEPLZ, PLZ
INTEGER PID, PPID, PPOD, RID, RDO, FILE
INTEGER BLDID, ACID
COMPLEX ERG, DICSTB, HUMPLX, SHADLX
DIMENSION INDAMP(147)
DIMENSION AMPX(4), PHASEX(4), ICONT(4)

C The variable timer and timer2 are used for timing measurements.
INTEGER TIMER, TIMER2

4 FORMAT(' NUMBER OF DIFFUSE GROUND COMPONENTS IS ', I2, ',
  * ONLY 25 COMPONENTS WILL BE USED.', )
100 FORMAT(1X, 'ENTER FULL NAME OF PROPAGATION MODEL INPUT',
  * 'DATA FILE', ', I2, ',
  */', ' (UP TO 40 CHARACTERS): ')
101 FORMAT(A40)
102 FORMAT(1X, 'ENTER FULL NAME OF PROPAGATION MODEL OUTPUT',
  * 'DATA FILE', ', I2, ',
  */', ' (UP TO 40 CHARACTERS): ')
103 FORMAT(1X, 'ENTER FULL NAME OF PROPAGATION MODEL OUTPUT',
  * 'DATA FILE', ', I2, ',
  */', ' (UP TO 40 CHARACTERS): ')
104 FORMAT(1X, 'FILE: ', I2, ', NAME: ', A40)
106 FORMAT(1X, 'FILE: ', I2, ', NAME: ', A40)
107 FORMAT(1X, 'ARE THESE FILE NAMES CORRECT? ')
108 FORMAT(1X, 'ENTER NUMBER OF FILE WHOSE NAME MUST BE MODIFIED: ')
110 FORMAT(I2)
120 FORMAT(' INVALID FILE NUMBER')
121 FORMAT(1X, 'FILE: ', I2, ', NAME: SYSTEM DEFAULT')
986 FORMAT(A132)
988 FORMAT(1X, A132)
59 FORMAT(//' PERTURBATION SMOOTHING? ')  
126 FORMAT(//' MLST SIMULATION PERFORMED BY: ')  
127 FORMAT(A40,3X,A9,3X,A8)  
128 FORMAT(//' DO YOU WANT TO SEE A COPY OF THE INPUT DATA? ')  
129 FORMAT(1X,5I2)  
131 FORMAT(1X,A5)  
134 FORMAT(//' PROCESS ',A14,' DATA? ')  
9995 FORMAT(//' EMPTY MEASURED FLIGHTPATH DATA FILE. ')  
9996 FORMAT(1X,A80)  
9997 FORMAT(A80)  
9998 FORMAT(1X,'IS THIS FILE NAME CORRECT? ')  
9999 FORMAT(//' ENTER NAME OF FILE CONTAINING MEASURED FLIGHTPATH DATA. ')  
7777 FORMAT(//' MLS MATHEMATICAL MODEL VERSION ',A3,' PROGRAM BMLST')  
7778 FORMAT(//' BEGINNING PROCESSING AT ',A8,/)  
7779 FORMAT(//' PROCESSING COMPLETED AT ',A8,/)  
       ANTTYP(1)=' AZIMUTH '  
       ANTTYP(2)=' ELEVATION '  
       ANTTYP(3)=' DME/P UPLINK '  
       ANTTYP(4)=' DME/P DOWNLINK '  
       TIMER=ICLOCK()  
       CALL SINIT  
       OPEN(ISIU,IOPSTAT=IOS,FILE='USRINT.DAT',STATUS='OLD')  
       OPEN(ISOU,IOPSTAT=IOS,FILE='ERR.DAT',STATUS='NEW')  
       WRITE(ISOU,100)PID  
       READ(ISIU,101)ITFILE  
       WRITE(ISOU,102)PPID  
       READ(ISIU,101)OTFILE  
       WRITE(ISOU,103)RID  
       READ(ISIU,101)ORFILE  
122 WRITE(ISOU,104)  
       IF(ITFILE.EQ.BLANK)THEN  
            WRITE(ISOU,121)PID  
       ELSE  
            WRITE(ISOU,106)PID,ITFILE  
       END IF  
       IF(OTFILE.EQ.BLANK)THEN  
            WRITE(ISOU,121)PPID  
       ELSE  
            WRITE(ISOU,106)PPID,OTFILE  
       END IF  
       IF(ORFILE.EQ.BLANK)THEN  
            WRITE(ISOU,121)RID  
       ELSE  
            WRITE(ISOU,106)RID,ORFILE  
       END IF  
       WRITE(ISOU,107)  
       CALL YESNO(ANSFL)  
       FIN=10  
       NODE=mynode()  
       PID=FIN+NODE  
       PPID=3*FIN+NODE
RID = 5*FIN + NODE
INAME = CHAR(ICHAR('A') + NODE)
C HERE THE VARIOUS FILES TO BE READ AND WRITTEN TO ARE HANDLED BASED
C ON NODE NUMBER
IF (ANSFL) THEN
  FNAME = 'PID_'
  SNAME = FNAME // INAME
  CALL SOPEN(PID, SNAME, 'OLD', 132)
  FNAME = 'PPID'
  SNAME = FNAME // INAME
  CALL SOPEN(PID, SNAME, 'NEW', 132)
ELSE
  WRITE (ISOU, 109)
  READ (ISIU, 110) FILE
    DO 90 ID = 1, NDIF
      K = K + 1
      IF (INDAMP(INY) .EQ. K) THEN
        CALL SETVAL (INY, VA(ID), PHAS(ID),
                      AZA(ID), ELA(ID), TD(ID), DOPF(ID), RDOPF(ID),
                      AZID(ID), ELID(ID))
        INY = INY + 1
      END IF
    END IF
DO 90
90 CONTINUE
END IF
IF (IPSS .EQ. 1) THEN
  DATAMP(IXMTR, IMDIAG) = AMAX1(0., AMPMAX)
  CALL CONCZL(IXMTR, YY, Z, W, CONAN(IXMTR, IMDIAG))
  IMDIAG = IMDIAG + 1
END IF
NCOMP = K
IF (IPSS .EQ. 1) THEN
  CALL MAXMLT(IXMTR, INDEX)
END IF
CALL EDTMLT(IXMTR, INDAMP)
IF (STOP .AND. IXMTR.EQ.IXSTOP .AND. IPSS.EQ.KPSS) THEN
  WSTOP = .TRUE.
ELSE
  WSTOP = .FALSE.
END IF
CALL WRTMLT(RID, IPSS, INDX, IXMTR, WSTOP, DIST)
NEWPT = 0
CONTINUE
END IF
20 CONTINUE
IF (IFIRST.EQ.0) THEN
  DSTMIN=DST
  IFIRST=1
END IF
CALL DIGOUT(STOP, NEWSEG, ICONT)
GO TO 11
10 REWIND PPID
CALL SINIT
WRITE (ISOU,7779) RTIME
CLOSE(RID)
CLOSE(ISIU)
CALL SIMDME(RID)
CLOSE(PID)
WRITE(ISOU,*)'RETURNED FROM WINDOW'
TIMER2 = TIMER-iclock()
CALL GSUM
STOP
500 WRITE (ISOU,9995)
STOP
END

SUBROUTINE SIMDME(IRID)
EXTERNAL DAC,WINPUL2,DFILT
COMMON/ACS/NAC,ACID(10),AC(4,10),NACTYP(10),ALT(10),
  * COSA(10),SINA(10),ACENT(3,10),ARCVR(3,10),ARVEL(3,10)
COMPLEX SIGTOT(8000),SIGDES(8000)
REAL SIGTIM(8000),SIGAMF(8000)
REAL*8 SIGAMP(8000)
REAL MPDB
CHARACTER*4 FIL
CHARACTER*5 FILOUT
REAL RCVR(3),RVEL(3),AMP(30),TDEL(30),PH(30)
CHARACTER*132 BUFFER
CHARACTER*1 STOP
INTEGER IANT(5),P
PI = ACOS(-1.0)
C = 9.8357120E8
PDUR = 5800.
RTIME = 800.
C SETTING UP FILENAMES AND NUMBERS FOR THIS ROUTINE BASED ON
C NODE NUMBER
  P=MYNODE()
  INP=P+16
  FIL='RlD_'
  FILOUT=FIL//(CHAR(ICHAR('A')+P))
  JUP=IRID
  OPEN(UNIT=JUP,IOSTAT=IOS,FILE=FILOUT,STATUS='OLD')
  FIL= 'RES_'
  FILOUT=FIL//(CHAR(ICHAR('A')+P))
OPEN(UNIT=INP, FILE=FILOUT, STATUS='UNKNOWN')
WRITE(INP,113)

113 FORMAT(1X,'RCVR XPOS ',4X,'RCVR YPOS',3X,'RCVR ZPOS ',3X,
   >' DIST ',3X,'ERROR IN FT',3X,' STOP ',/)

100 IF (BUFFER(1:8).NE.'END DATA') THEN
   READ(JUP,10) BUFFER
   GO TO 100
END IF

READ(JUP,*) (IANT(I), I=1,5)

73 READ(JUP,*) IPSS,INDX,XMTR,STOP,DIST,(RCVR(IA),IA = 1,3),
   >(RVEL(IA),IA = 1,3), NCOMP
READ(JUP,*) (AMP(IX),PH(IX),AZ,EL,TDEL(IX),TDOP,
   >RDOP, AZIN, ELIN, IX=1,NCOMP)
DO 150 I = 1,8000
   SIGDES(I) = (0.,0.)
150 CONTINUE
DO 163 I = 1,8000
   SIGTOT(I) = SIGDES(I)
163 CONTINUE
DO 63 ID = 1,NCOMP
   AMPMP = AMP(ID)
   DELYMP = TDEL(ID)/1E-09
   PHASE = PH(ID)
   CALL WINPUL2(PDUR,RTIME,AMPMP,6500.+DELYMP,PHASE,SIGTOT)
63 CONTINUE
DO 200 J = 1,8000
   SIGTIM(J) = FLOAT(J)
   SIGAMP(J) = DBLE(CABS(SIGTOT(J)))
200 CONTINUE
CALL DFILT(SIGAMP,SIGAMF,8000)
CALL DAC(SIGAMF,SIGTIM,8000,100.,0.5,YNDAC,TIMDAC)
ERROR = (TIMDAC - 7006.)*(1.0E-9)*C
IF(YNDAC.LT.(1.)) WRITE(*,*)'NO DECODE!'
WRITE(INP,51) RCVR(1),RCVR(2),RCVR(3),DIST,ERROR,STOP

51 FORMAT(5(1X, F11.4),5X, A1)
IF (STOP.EQ.'F') GO TO 73
CLOSE(INP)
CLOSE(JUP)
CALL WINDOW(INP, FILOUT)
RETURN

SUBROUTINE WINDOW(MJP,NNAME)
DIMENSION DIST(1350)
DIMENSION IFLAG(3)
COMMON/ERRDTA/VERR(1350), X(1350), Y(1350), Z(1350), POSN(1350)
COMMON/CMMXT/ICONT(4,4), NFLTPT, I SYS, ITYPE
COMMON/HEADER(20), I ANS, DEC NM
COMMON/TITLE/ TITLE1(18), TITLE2(18)
COMMON/ACS/NAC, ACID(10), AC(4,10), NACTYP(10), ALT(10),
   * COSA(10), SINA(10), ACENT(3,10), ARCVR(3,10), ARVEL(3,10)
CHARACTER*1 STOP
REAL NAUT
CHARACTER*80 BUFFER
CHARACTER*5 NNAME
DATA DEGREE/57.29577951/
DATA NAUT/6076.0/
557 DO 88889 I=1,1350
   VERR(I)=0.0
88889 CONTINUE
   OPEN(MJP,IOSTAT=IOS,FILE=NNAME,STATUS= 'OLD')
   ISYS = 2
   IDUM=0
99998 CONTINUE
345 FORMAT(1X,'BEGIN READ STATEMENT')
       READ(MJPI3)BUFFER
       READ(MJPI3)BUFFER
       I=1
3 FORMAT(A80)
73 READ(MJPI3) X(I),Y(I),Z(I),DIST(I),VERR(I),STOP
36 FORMAT(1X,F11.4,1X,F11.4,1X,F11.4,1X,F11.4,1X,F11.4,5X,A1)
       POSN(I) = DIST(I)/NAUT
       I=I+1
       IF(STOP.EQ.'F') GO TO 73
542 FORMAT(1X,'END READ STATEMENT')
99997 CONTINUE
77776 CONTINUE
       IF(I.NE.0) GO TO 77777
       IFLAG(ISYS)=1
       GO TO 99999
77777 CONTINUE
       IFLAG(ISYS)=0
       NFLPT=I-1
       TEMP1=VERR(1)
       TEMP2=VERR(2)
       VERR(1)=0.0357260902*TEMP1
       VERR(2)=0.0357260902*(TEMP2+2.0*TEMP1)+1.24394614*VERR(1)
       DO 333 I=3,NFLPT
         TEMP3=VERR(I)
         VERR(I)=0.0357260902*(TEMP3+2.0*TEMP2+TEMP1)+1.24394614
         >*VERR(I-1)-0.386850499*VERR(I-2)
         TEMP1=TEMP2
         TEMP2=TEMP3
333 CONTINUE
     CALL WINDEX
761 FORMAT(1X,'END OF FILT AND WINDEX SUBROUTINE')
99999 CONTINUE
     INPLT=0
     IF(IFLAG(ISYS).EQ.1) GO TO 55555
     INPLT=INPLT+1
55555 CONTINUE
     IF(INPLT.EQ.0) WRITE(6,1002)
1002 FORMAT(10X,'NO DATA/S ARE AVAILABLE')
SUBROUTINE WINDEX
COMMON/ERRDTA/VERR(1350),X(1350),Y(1350),Z(1350),POSN(1350)
COMMON/CXMT/CONT(4,4),NFLTPT,ISYS,ITYPE
COMMON/AC/SAC,ACID(10),AC(4,10),NACTYP(10),ALT(10),
* COSA(10),SINA(10),AENT(3,10),ARCVR(3,10),ARVEL(3,10)
DIMENSION PFE(1350,14),CMN(1350,14),POS(1350,14)
REAL HTOL(26)
CHARACTER*3 FNAME
CHARACTER*5 SNAME
INTEGER D,DD,E,U,FC,FP
INTEGER CNT(14)
DATA HTOU200.0,125.0,
> 100.0,75.0,50.0,
> 47.5,45.0,42.5,41.0,37.5,36.0,
> 33.5,30.0,27.5,25.0,22.5,20.0,17.5,15.0,
> 12.0,10.0,7.5,5.0,3.0,1.0,0.5/
LINDX = 26
MTOLER=2
77445 FORMAT(I2)
NWD1=50
2134 FORMAT(I3)
HEIGHT=41.0
6754 FORMAT(F7.5)
IF (NACTYP(1) .EQ. 1) THEN
   PLAWID = 116.0
ELSE
   IF (NACTYP(1) .EQ. 3) PLAWID = 66.0
END IF
IF(AC(1,1).EQ.AC(3,1)) THEN
   YPOS = AC(2,1)+PLAWID
ELSE
   YPOS = AC(2,1)
END IF
5024 FORMAT(F10.3)
   IF(AC(2,1).EQ.AC(4,1)) THEN
      XPOS = AC(1,1)+PLAWID
   ELSE
      XPOS = AC(1,1)
   END IF
8047 FORMAT(F10.3)
   PHIGHT=41.0
2947 FORMAT(F7.5)
   MPTOL=2
33544 FORMAT(I2)
   CHITE = HEIGHT*2.0
DO 7889 K = 1,NFLTPT
   IF(POSN(K).GT.(2.013))GOTO 12
   PFE(K,14)=VERR(K)
   POS(K,14)=POSN(K)
CNT(14)=K
12 CONTINUE
   IF(POSN(K).GT.(2.327))GOTO 13
   PFE(K,13)=VERR(K)
   POS(K,13)=POSN(K)
   CNT(13)=K
13 CONTINUE
   IF(POSN(K).GT.(2.642))GOTO 14
   PFE(K,12)=VERR(K)
   POS(K,12)=POSN(K)
   CNT(12)=K
14 CONTINUE
   IF(POSN(K).GT.(2.956))GOTO 15
   PFE(K,11)=VERR(K)
   POS(K,11)=POSN(K)
   CNT(11)=K
15 CONTINUE
   IF(POSN(K).GT.(3.271))GOTO 16
   PFE(K,10)=VERR(K)
   POS(K,10)=POSN(K)
   CNT(10)=K
16 CONTINUE
   IF(POSN(K).GT.(3.585))GOTO 17
   PFE(K,9)=VERR(K)
   POS(K,9)=POSN(K)
   CNT(9)=K
17 CONTINUE
   IF(POSN(K).GT.(3.900))GOTO 18
   PFE(K,8)=VERR(K)
   POS(K,8)=POSN(K)
   CNT(8)=K
18 CONTINUE
   IF(POSN(K).GT.(4.057))GOTO 19
   PFE(K,7)=VERR(K)
   POS(K,7)=POSN(K)
   CNT(7)=K
19 CONTINUE
   IF(POSN(K).GT.(4.214))GOTO 20
   PFE(K,6)=VERR(K)
   POS(K,6)=POSN(K)
   CNT(6)=K
20 CONTINUE
   IF(POSN(K).GT.(4.371))GOTO 21
   PFE(K,5)=VERR(K)
   POS(K,5)=POSN(K)
   CNT(5)=K
21 CONTINUE
   IF(POSN(K).GT.(4.528))GOTO 22
   PFE(K,4)=VERR(K)
   POS(K,4)=POSN(K)
   CNT(4)=K
22 CONTINUE
IF(POSN(K).GT.(4.686))GOTO 23
PFE(K,3)=VERR(K)
POS(K,3)=POSN(K)
CNT(3)=K
23 CONTINUE
IF(POSN(K).GT.(4.843))GOTO 24
PFE(K,2)=VERR(K)
POS(K,2)=POSN(K)
CNT(2)=K
24 CONTINUE
PFE(K,1)=VERR(K)
POS(K,1)=POSN(K)
7889 CONTINUE
CNT(1)=NFLTPT
U = 0
LWING = 1
LWINP = 1
DO 3902 E = 1,29,2
   IF(E.EQ.5) GOTO 3902
P=MYNODE()
NNO=P+9
DD=E
FNAME='ERR1
SNAME=FNAME//(CHAR(ICHAR('A')
OPEN(NNO,IOSTAT=IOS,FILE=SNAME,STATUS='NEW')
U=U+1
FP = 0
FC = 0
NWIDTH = NWID1
IF((CNT(U)).LT.NWIDTH) NWIDTH = CNT(U)
77  K = CNT(U) - NWIDTH + 1
IF(LWINP.EQ.LINDX) GOTO 104
144  DO 600 LL = LWINP,26
400  DO 981 = 1,K
   MERRPT = 0
   N = I + NWIDTH - 1
   DO 121 J = I,N
      IF(ABS(PFE(J,U)).GT.HTOL(LL)) MERRPT = MERRPT + 1
      IF((MERRPT.GT.MPTOL).AND.(I.EQ.1)) FP = 1
      IF(MERRPT.GT.MPTOL) GOTO 576
   121 CONTINUE
   98 CONTINUE
   IF(LL.NE.LINDX) GOTO 600
104 CONTINUE
WRITE(NNO,3)XPOS,YPOS,HTOL(LINDX),POSN(I),POSN(N),FP,DD
LWINP = LINDX
GOTO 3902
576 IF(LL.EQ.1) NL = 2
   IF(LL.GT.1) NL = LL
   NL = NL - 1
   WRITE(NNO,3)XPOS,YPOS,HTOL(NL),POSN(I),POSN(N),FP,DD
   LWINP = NL
SUBROUTINE GSUM
C THIS ROUTINE IS USED TO SYNCHRONIZE ALL THE NODES AFTER A SINGLE EXECUTION
C NODE 0 IS INFORMED THAT ALL THE TASKS HAVE BEEN COMPLETED AND HENCE
C NEW GRID POINTS CAN BE ASSIGNED.
    INTEGER ALL, ANY, FALSE, HOST, TRUE
    PARAMETER(FALSE = 0, TRUE = 1)
    PARAMETER(ALL = -1, ANY = -2)
    PARAMETER(HOST = -1)
    REAL SUM
    REAL MSGVAL
    INTEGER NODE
    INTEGER MSGTYP, MSGLEN, MSGPID, MSGNOD
    DATA MSGTYP, MSGLEN / 600, 4 / 
    MSGPID = MYPID
    SUM = 4.0
    IF (MYNODE() .EQ. 0) THEN
        OPEN(UNIT=75, IOSTAT=IOS, FILE='MESG2', STATUS='UNKNOWN')
        DO 10 NODE = 1, NNODES - 1
            MSGNOD = ANY
            CALL Frecv(MSGVAL, MSGLEN, MSGTYP, MSGNOD, MSGPID)
            SUM = SUM + MSGVAL
        10    CONTINUE
    ELSE
        CALL Fsend(SUM, MSGLEN, MSGTYP, 0, MSGPID)
        MSGNOD = 0
    ENDIF
    IF (MYNODE() .EQ. 0) THEN
        WRITE(75, *)'END'
        CLOSE(75)
    END IF
    WRITE(*,*) IN GSUM
RETURN
END
C.3 PARFFT

In this routine parallel computation of the 2-D DFT is decomposed into number of 1-D FFT's using the row-column based decomposition. This routine requires interprocessor communication to complete the transfer of intermediate results. Each node calculates the 1-D FFT's on 1/16 th of the entire grid in parallel. The parallel computation of the 2-D DFT is described in great detail in Chapter VI.
PARFFT

This routine determines the 2-D FFT using the row-column decomposition algorithm.

PROGRAM PARFFT

C DECLARATIONS
REAL*4 ERR,FR(64),FX(64),Y(64),Z(64)
REAL*4 FC(64),XR(64),XC(64)
COMPLEX F(64,64),X(64),F1(64,64)
CHARACTER*3 FNAME
C PARAMETERS FOR MESSAGE PASSING
MSGLEN1=256
MSGLEN2 = 128
MSGTYP = 600
MSGTYP1 = 601
MSGTYP2 = 602
MSGTYP3 = 603
MSGTYP4 = 604
MNOD = MYNODE() 
MPID = MYPID()
NNOD = NNODES()
C OUTPUT FILE TO BE WRITTEN BY NODE 0
IF(MNOD.EQ.0) THEN
C INPUT FILE WITH CONTOUR DATA
OPEN(UNIT=12,IOSTAT=IOS,FILE='XA',STATUS='OLD')
OPEN(UNIT=21,IOSTAT=IOS,FILE='F2A',STATUS='NEW')
END IF
C INPUT FILE WITH CONTOUR DATA
C NOW THE SECTION TO READ DATA Follows
ICOL=16 
IROW=8
IF(MNOD.EQ.0) THEN
DO 15 I =1,IROW
DO 10 J=1,ICOL
READ(12,*)XPOS,YPOS,ERR,SWIN,STWIN,L
F(I,J)=CMPLX(ERR,0.0)
10 CONTINUE
15 CONTINUE
C THE ABOVE CONDITION HELPS SKIP ROWS
C EACH NODE READS ONLY ONE SECTION OF THE DATA FILE
CLOSE(12)
DO 25 I=IROW+1,64
DO 20 J=ICOL+1,64
F(I,J)=CMPLX(0.0,0.0)
20 CONTINUE
25 CONTINUE
DO 30 J = 1,NNOD-1
DO 40 K =1,64,16
DO 50 L=1,64
   Y(L)=REAL(F(J+K,L))
   X(L)=F(J+K,L)
50    CONTINUE
   CALL FSEND(Y,MSGLEN1,MSGTYP,J,MPID)
40    CONTINUE
30    CONTINUE
END IF
DO 70 K=1,64,16
IF (MNOD.EQ.0) THEN
   DO 80 L=1,64
      X(L)=F(K,L)
80    CONTINUE
ELSE
   CALL FRECV(Y,MSGLEN1,MSGTYP,0,MPID)
   DO 81 L=1,64
      X(L)=CMPLX(Y(L),0.0)
81    CONTINUE
END IF
CALL FFT(X,6)
DO 90 I = 1,64
   XR(I)=REAL(X(I))
   XC(I)=AIMAG(X(I))
90    CONTINUE
C
C SEND TO AND RECEIVE THE 1-D ROW TRANSFORMED ARRAYS
C
DO 100 I = 1,NNOD
DO 105 L = 1,64
   FR(L)=XR(L)
   FC(L)=XC(L)
105    CONTINUE
   J=I-1
   CALL FSEND(FR,MSGLEN1,MSGTYP1,J,MPID)
   CALL FSEND(FC,MSGLEN1,MSGTYP2,J,MPID)
   CALL FRECV(FR,MSGLEN1,MSGTYP1,J,MPID)
   CALL FRECV(FC,MSGLEN1,MSGTYP2,J,MPID)
   DO 110 L = 1,64
      F1(J+K,L)=CMPLX(FR(L),FC(L))
110    CONTINUE
100    CONTINUE
70    CONTINUE
DO 120 K =1,64,16
   IMNO=MNOD+K
   DO 130 J = 1,64
      X(J)=F1(J,IMNO)
130    CONTINUE
CALL FFT(X,6)
DO 140 L = 1,64
  FR(L)=REAL(X(L))
  FC(L)=AIMAG(X(L))
140  CONTINUE
IF (MNOD.NE.0) THEN
  CALL FSEND(FR,MSGLEN1,MSGTYP3,0,MPID)
  CALL FSEND(FC,MSGLEN1,MSGTYP4,0,MPID)
ELSE
  DO 150 L =1,64
    F1(L,IMNO)=0.0
    F1(L,IMNO)=X(L)
 150  CONTINUE
END IF
IF(MNOD.EQ.0) THEN
  DO 160 M = 1,NNOD-1
    CALL FRECV(FR,MSGLEN1,MSGTYP3,M,MPID)
    CALL FRECV(FC,MSGLEN1,MSGTYP4,M,MPID)
    IND = IMNO+M
    DO 170 J = 1,64
      F1(J,IND)=0.0
      F1(J,IND)=CMPLX(FR(J),FC(J))
    170  CONTINUE
  160  CONTINUE
END IF
IF(MNOD.EQ.0) THEN
  DO 230 J = 1,NNOD-1
    DO 240 K = 1,64,16
      Y(L)=REAL(F1(J+K,L))
      Z(L)=AIMAG(F1(J+K,L))
    240  CONTINUE
    CALL FSEND(Y,MSGLEN1,MSGTYP1,J,MPID)
    CALL FSEND(Z,MSGLEN1,MSGTYP,J,MPID)
  230  CONTINUE
END IF
K=1
DO 270 K = 1,64,16
IF (MNOD.NE.0) THEN
  CALL FRECV(Y,MSGLEN1,MSGTYP1,0,MPID)
  CALL FRECV(Z,MSGLEN1,MSGTYP,0,MPID)
  DO 381 L = 1,64
    X(L)=CMPLX(Y(L),-Z(L))
  381  CONTINUE
ELSE
  DO 255 L =1,64
    X(L)=CMPLX(REAL(F1(K,L)),-AIMAG(F1(K,L)))
  255  CONTINUE
END IF
CALL FFT(X,6)
DO 290 I = 1,64
   XR(I) = 64.0*REAL(X(I))
   XC(I) = 64.0*AIMAG(X(I))
290 CONTINUE
C
SEND TO AND RECEIVE THE 1-D ROW TRANSFORMED ARRAYS
C
DO 500 I = 1,NNOD
DO 505 L = 1,64
   FR(L) = XR(L)
   FC(L) = XC(L)
505 CONTINUE
J = I-1
CALL FSEND(FR,MSGLEN1,MSGTYP2,J,MPID)
CALL FSEND(FC,MSGLEN1,MSGTYP3,J,MPID)
CALL FRECV(FR,MSGLEN1,MSGTYP2,J,MPID)
CALL FRECV(FC,MSGLEN1,MSGTYP3,J,MPID)
DO 510 L = 1,64
   F1(J+K,L) = CMPLX(FR(L),-FC(L))
510 CONTINUE
500 CONTINUE
C
IND = 0
DO 520 K = 1,64,16
   IMNO = MNOD + K
   DO 530 J = 1,64
      X(J) = F1(J,IMNO)
   530 CONTINUE
C
CALL FFT(X,6)
DO 540 L = 1,64
   FX(L) = 64.0*CABS(X(L))
540 CONTINUE
CALL FSEND(FX,MSGLEN1,MSGTYP4,0,MPID)
C
IF(MNOD.EQ.0) THEN
DO 560 M = 1,NNOD
   MN = M-1
   CALL FRECV(FX,MSGLEN1,MSGTYP4,MN,MPID)
   WRITE(21,*)(FX(L),L = 1,64)
   IND = IMNO + M-1
   DO 570 J = 1,64
      F1(J,IND) = FX(J)
   570 CONTINUE
   560 CONTINUE
END IF
520 CONTINUE
IF(MNOD.EQ.0) THEN
CLOSE(12)
CLOSE(21)
END IF
STOP
END

SUBROUTINE FFT(F,LN)
COMPLEX F(64),U,W,T,CMPLX
PI=3.141593
N=2**LN
NV2=N/2
NM1=N-1
J=1
DO 3 I = 1,NM1
IF(I.GE.J) GO TO 1
T=F(J)
F(J)=F(I)
F(I)=T
K=NV2
IF (K.GE.J) GO TO 3
J=J-K
K=K/2
GO TO 2
3 J=J+K
DO 5 L=1,LN
LE=2**L
LE1=LE/2
U=(1.0,0.0)
W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
DO 5 J = 1,LE1
DO 4 I=J,N,LE
IP=I+LE1
T=F(IP)*U
F(IP)=F(I)-T
F(I)=F(I)+T
U=U*W
DO 6 I=1,N
F(I)=F(I)/FLOAT(N)
CONTINUE
RETURN
END