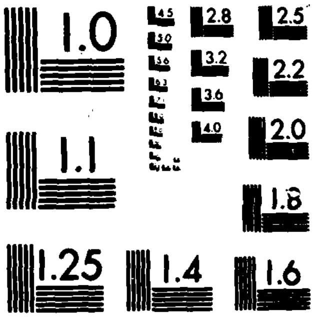
AD-A164 020 NUMERICAL GRID GENERATION FOR PARABOLIC PARTIAL 1/2
DIFFERENTIAL EQUATIONS US..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. S G MILLER
UNCLASSIFIED DEC 85 AFIT/GA/AA/85D-7 F/G 20/4 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

①

AD-A164 020

NUMERICAL GRID GENERATION FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS USING MARCHING TECHNIQUES

THESIS

Steven G. Miller
First Lieutenant, USAF
AFIT/GA/AA/85D-7

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DTIC
S ELECTE D
FEB 1 2 1986
B

DTIC FILE COPY

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

86 2 12 065

①

AFIT/GA/AA/85D-7

NUMERICAL GRID GENERATION FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS USING MARCHING TECHNIQUES

THESIS

Steven G. Miller
First Lieutenant, USAF
AFIT/GA/AA/85D-7

DTIC
S ELECTE D
FEB 1 2 1986
B

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

AFIT/GA/AA/85D-7

NUMERICAL GRID GENERATION FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS USING MARCHING TECHNIQUES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Astronautical Engineering

Steven G. Miller
First Lieutenant, USAF

December 1985

Approved for public release; distribution unlimited

Preface

This study develops a new technique for generating flow field grids around arbitrary configurations such as transatmospheric vehicles. missiles. advanced fighters etc. Since flight and wind tunnel testing are very expensive. it is advantageous to generate computational solutions first to minimize the wind tunnel testing. Therefore. it is crucial to construct grids for these computational solutions in such a manner that the flow field is accurately represented on the grid. At the present time representative grids are generated by elliptic partial differential equations at the cost of a great deal of computer resources. This study will demonstrate how representative grids can be generated by parabolic partial differential equations in a fraction of the time used by elliptic partial differential equations. It should be noted that the type of grid that should be utilized is dictated by the method used in the flow solver. The scope of this thesis is limited to the development of grid generation procedures and not in the generation of flow solutions. Flow solutions should be obtained with this grid procedure and compared with solutions using other grid procedures. Due to a lack of time and computer resources the flow solutions can not be accomplished.

I would like to thank Major James K. Hodge for suggesting the thesis topic and the many helpful discussions. I would also like to thank Dr. Joseph Shang and Dr. Sal Leone for serving as members of my thesis committee.

Steven G. Miller

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist Avail and/or Special
A-1

QUALITY INSPECTED 3

Table of Contents

Page

Preface ii

List of Figures v

List of Symbols vii

Abstract ix

I. Introduction 1

- 1.1 Background 1
- 1.2 Motivation 4
- 1.3 Problems 5
- 1.4 Objectives 5
- 1.5 Overview 6

II. Waverider Configuration 7

- 2.1 Definition of a Waverider 7
- 2.2 Construction of a Waverider 7
- 2.3 Equations to Generate Waverider Geometry 12

III. Parabolic Grid Equations 15

- 3.1 One-Dimensional Grid Generation Equation 15
- 3.2 Two-Dimensional Grid Generation Equations 16
- 3.3 Three-Dimensional Grid Generation Equations 19

IV. Numerical Procedure 22

- 4.1 Two-Dimensional Wrapped Grid Procedure 22
  - 4.1.1 Program Setup 22
  - 4.1.2 Boundary Points 22
  - 4.1.3 One-Dimensional Elliptic Equation With Q Distribution 26
  - 4.1.4 Plane-of-Symmetry Conditions 26
  - 4.1.5 Check on Convexity 27
  - 4.1.6 Orthogonal Grid Generation 30
  - 4.1.7 Grid Crossover Check 35
  - 4.1.8 Two-Dimensional Parabolic Equation 35
- 4.2 Three-Dimensional Wrapped Grid Procedure 37
  - 4.2.1 Boundary Points 38
  - 4.2.2 Three-Dimensional Starting and Stopping Procedures 41
  - 4.2.3 Orthogonal Grid Generation 43

V. Results 48

5.1 Two-Dimensional Wrapped Grid Problems and Results 48
  5.1.1 Boundary Generation Problems 48
  5.1.2 Wing Tip Grid Wrap Around Problem 51
  5.1.3 Projected Solution Convexity Problem 52
  5.1.4 Grid Crossover Problem 55
  5.1.5 Results 55
5.2 Three-Dimensional Wrapped Grid Problems and Results 60

VI. Conclusion 72

VII. Recommendations 73

Bibliography 74

Appendix A: Two-Dimensional Slit Transformed Grid Procedure 76

A.1 Computational Domain 76
A.2 Boundary Points 76
A.3 Orthogonal Grid Generation 78

Appendix B: Two-Dimensional Wrapped Grid Computer Code 80

Appendix C: Three-Dimensional Wrapped Grid Computer Code 105

Vita 129

List of Figures

Figure **Title** **Page**

2-1. Cone Geometry Showing Streamlines Going Through Shock Wave and Asymptotically Approaching the Cone's Surface 9

2-2. Cone Geometry with Symmetrical Plane Stream Surfaces Inclined at Angle Phi 10

2-3. Pointed Nose Waverider with Infinitesimally Thin Wings 11

2-4. Rounded Nose Waverider Configuration 14

4-1. Criterion for Good and Bad Projected Solution Positions 29

4-2. Effect of Hyperbolic Tangent on the Generation of a Nearly Orthogonal Curve 33

4-3. Waverider Shock Intersections in X-Z Plane and Y-Z Plane 39

4-4. Finite Differencing Nodes Used in 3-D Parabolic Grid Generation 42

4-5. Angles Used in Calculating the 3-D Nearly Orthogonal Curves 44

5-1. Three Wing Tip Grid Lines Generated by Distributing Points Differently on the Waverider's Surface 50

5-2. Grid Lines Emanating from Wing Tip 53

5-3. Projected Solution at Maximum Position Causing Grid to Cross Boundaries 54

5-4. Convexity Criterion Backs off Maximum Projected Solution Position to an Acceptable Position with Maximum Smoothing 56

5-5. Minimum Projected Solution Position Gives Minimum Smoothing 57

5-6. Non-Orthogonal Waverider Grid (2-D Parabolic Grid) 59

5-7. Nearly Orthogonal Waverider Grid with Grid Crossover (2-D Parabolic Grid) 61

5-8. Nearly Orthogonal Waverider Grid with Grid Crossover Controlled by Re-Calculating Hyperbolic Tangent (2-D Parabolic Grid) 62

5-9. Nearly Orthogonal Waverider Grid with Grid Crossover Controlled by Clustering Points (2-D Parabolic Grid) 63

5-10. Nearly Orthogonal Waverider Grid with Automatic Grid Crossover Check (2-D Parabolic Grid) 64

5-11. 3-D Upper Surface Paneling of Waverider Configuration 67

5-12. 3-D Lower Surface Paneling of Waverider Configuration 68

5-13. Nearly Surface Orthogonal Waverider Grid in Plane-of-Symmetry (3-D Parabolic Grid) 69

5-14. Nearly Surface Orthogonal Waverider Grid with Cut-Away (3-D Parabolic Grid) 71

A-1. Transformation of Physical Space into Either a Rectangular or a Slit Computational Space 77

List of Symbols

$C_{k\ell}$	Signed Cofactor of a Matrix
I^-	Position of Projected Solution in ξ Direction
i	Position of Solution Station in ξ Direction
IMAX	Number of Grid Points in ξ Direction
j	Position of Solution Station in η Direction
JMAX	Number of Grid Points in η Direction
K^-	Position of Projected Solution in ζ Direction
k	Position of Solution Station in ζ Direction
KMAX	Number of Grid Points in ζ Direction
M_∞	Mach Number
n	Waverider Shape Factor
P	Source Term in ξ Direction
Q	Source Term in η Direction
R	Source Term in ζ Direction
s	Arc Length
x_1,y_1	Inner Boundary Point
x_2,y_2	Outer Boundary Point
x,y,z	Cartesian Coordinates in Physical Space
r,θ,z	Cylindrical Coordinates in Physical Space
r,θ,ϕ	Spherical Coordinates in Physical Space
ξ,η,ζ	Transformed Coordinates in Computational Space
$\alpha_{i,j}$	Coefficients of Elliptic and Parabolized Equations
β	Shock Wave Angle
γ	Specific-Heat Ratio

δ	Semi-Vertex Angle of a Cone
δ_1	Waverider's Lower Surface Plane-of-Symmetry Locator
ϕ	Dihedral Angle
ϕ_1	Angle of Straight Line Connecting Boundary Points in x-y Plane
ϕ_2	Angle of Normal Line to Surface in x-y Plane
ϕ_3	Angle of a Calculated Point on an Orthogonal Curve in x-y Plane
ψ_1	Angle of Straight Line Connecting Boundary Points in x-z Plane
ψ_2	Angle of Normal Line to Surface in x-z Plane
ψ_3	Angle of a Calculated Point on an Orthogonal Curve in x-z Plane
$\mu(S)$	Finite Differencing Switch that Depends on the Sign of the Source Term

Subscripts

b	Lower or Compression Surface of Waverider
fs	Freestream Surface of Waverider
$i \pm w$	Spatial Description at Distance w Away from Solution Station in ξ Direction
I^-	Position of Projected Solution in ξ Direction
$j \pm w$	Spatial Description at Distance w Away from Solution Station in η Direction
$k \pm w$	Spatial Description at Distance w Away from Solution Station in ς Direction
K^+	Position of Projected Solution in ς Direction
n	Recursion Relation
$x, y, z, \xi, \eta, \varsigma$	First Derivative with Respect to the Variable
$xx, yy, zz, \xi\xi, \eta\eta, \varsigma\varsigma, \xi\eta, \xi\varsigma, \eta\varsigma$	Second Derivative with Respect to the First Variable First then with Respect to the Second Variable

Abstract

Two- and three-dimensional surface normal grids are generated in Cartesian coordinates around a supersonic/hypersonic waverider configuration using parabolic partial differential equations. The elliptic partial differential equations for grid generation are parabolized in one direction for two dimensions, and in two directions for three dimensions. This is consistent with spatial marching flow solutions. The parabolized grid equations march in one direction for two dimensions and in two directions for three dimensions, without iteration. The following problems are investigated: description of the boundary points, grid generation around the waverider's wing tip, approximations to the elliptic grid generation equations around a convex corner, and grid crossover in concave regions when orthogonality is specified. The degree of grid smoothing in the marching directions is related to the positioning of the approximations to the elliptic grid generation equations. Highly stretched surface orthogonal grids are accurately generated without grid embedding for high Reynold's number flows, in less than one percent of the computer time required by elliptic grid generators.

I. Introduction

1.0 Background

Today in computational fluid dynamics, it is difficult to generate flow field grids around an arbitrary three-dimensional (3-D) aerodynamic configuration. There are many ways to generate grids, among these are conformal mapping procedures, algebraic methods and differential equation techniques[1,2]. The differential equation techniques require that elliptic, parabolic or hyperbolic partial differential equations (PDEs) be employed. The elliptic PDEs are the most widely developed. But, the elliptic grid generation equations are costly in terms of computer time due to the numerous iterations involved in solving these equations. The hyperbolic PDEs are the next most widely developed procedure. Although this procedure can produce grids quickly with its marching techniques, the outer boundary can not be specified. If the boundary contains a discontinuity, the discountinuity is propagated into the interior of the flow field and can produce a grid "shock." The parabolic PDEs that approximate the elliptic grid generation equations, are the most promising for rapidly generating a good solution. If one is not careful there is a danger of grid crossover, which will yield a zero Jacobian or violate the maximum principle.

Thompson[3] in one study, and Steger and Sorenson[4] in another study, used elliptic PDEs to generate flow field grids. The Poisson equation was used to generate highly-stretched grids by clustering points near a boundary. The elliptic equations, with Dirchlet boundary conditions, insure an unique solution. This guarantees a one-to-one mapping between the physical and computational planes, although the differenced form of the equation may not. With Neumann boundary conditions set on the boundaries, grid lines are orthogonal to the inner boundary surface, but may produce a zero Jacobian. The elliptic equations smooth out discountinuities from the boundaries without the danger of grid crossover. Because the domain of influence covers the whole computational plane, all points affect each other. The elliptic equations generate good grids, except numerous iterations are needed. The solution of the elliptic equations is inherently expensive and grid control is difficult.

Steger and Chaussee[5] used hyperbolic PDEs to reduce the computer time needed to generate a grid. The hyperbolic equations march in all directions, which is computationally efficient. If there is a discountinuity on the boundary, it is propagated into the interior of the flow field. To keep these equations stable, artificial viscosity must be included in the finite difference equations. The hyperbolic equations can generate grids which are nearly orthogonal, but the outer boundary points can not be specified. On a concave surface, orthogonal grid lines will coalesce, causing a grid "shock." This is also true for most grid generation procedures.

To specify the outer boundary, reduce grid generation time and enhance grid control, Nakamura[6] used parabolic PDEs. The parabolic equations are an initial value and boundary value problem and can take advantage of marching techniques. By parabolizing the elliptic equations, the parabolic equations reproduce most of the properties of elliptic equations. The parabolized equations reproduce the diffusion effect which smooths out discontinuities that may be present on the inner boundary. The parabolic equations also allow the outer boundary surface to be specified. The elliptic equations are parabolized in the η direction and the solution is marched from the inner boundary to the outer boundary. The parabolic equations require that an approximation to the elliptic equations be made either upstream or downstream of the solution position. The outer boundary is used for this approximation in this case. Since the outer boundary is generally distant from the inner boundary, there could be too much smoothing near the inner boundary for a highly-curved complex geometry. The Laplace equation is parabolized instead of the Poisson equation, therfore grid points have to be clustered by some other means than specifying the source terms. Points are clustered near a boundary by using linear relationships. Near orthogonality at the inner boundary is also accomplished by linear relationships and must be re-calculated after each marching step. When *generating grids in three dimensions*. *Nakamura*[6] *marches in both the* η *and* ς directions. Since the 3-D grid, in this case, marches from the inner boundary to the outer boundary in one of its directions, the whole grid has to be generated and stored before there is enough information for a flow solver to be initiated.

Edwards[7] also used the Laplace equation and parabolized it in the η and ς directions. Points were clustered and orthogonality was imposed in much the same way that Nakamura[6] did. Grid embedding was used to enhance the grid density near the inner boundary. This option causes problems when coupled to a flow solver. Boundary conditions between the regular grid and the embedded grid have to be matched. If grid stretching and the outer boundary is selected correctly, grid embedding can be advoided. In physical space, a rectangular boundary is used. This makes it difficult to disperse points near the outer boundary where there are no viscous effects.

Noack[8] also solved the parabolic PDEs to generate two-dimensional (2-D) grids in each cross plane of a 3-D axis normal grid. The Laplace equation is parabolized in the η direction as Nakamura[6] and Edwards[7]. Both Nakamura[6] and Edwards[7] used the outer boundary to approximate the elliptic equations. As stated before, this can cause too much smoothing near the inner boundary if the geometry has a highly-curved surface. The elliptic equations are approximated at the next station out beyond the solution boundary. This corrects the problem of a possibility of too much smoothing, but in some cases it may not generate enough smoothing.

Finally, Hodge, Leone and McCarty[9] used parabolic PDEs to generate 3-D grids.

Instead of marching in the conventional directions η and ζ, the elliptic equations are parabolized in the ξ and ζ directions. This has great significance when grids are generated in three dimensions because only three 2-D planes or surfaces need to be stored at one time. Since these solution procedures also march downstream the whole grid is not stored at one time, decreasing memory storage requirements. Hodge, et al[9] parabolized the Poisson equations. Specifying the source terms in the Poisson equations allows the grid to be highly-stretched for high Reynold's number flows, without using grid embedding. The approximation of the elliptic equations can be placed anywhere before or after the present solution. Hodge, et al[9] did not extend the grids to be orthogonal near the inner boundary, but were generated normal to the axis only. Since the 3-D grid equations are used, changes from one cross plane to another are smoothed. In addition, the grid can be adapted while marching.

1.2 Motivation

Missiles with non-circular lifting body cross-sections are of current interest as a means for obtaining high-performance at supersonic/hypersonic speeds. Missiles need to achieve higher performance in order to outmaneuver advanced fighters. A waverider configuration will generate these high-performance characteristcs. In the development of a waverider configuration, designs must be subjected to wind tunnel tests and numerical simulation. Computational methods such as Navier-Stokes or Parabolizied Navier-Stokes must be used to take into account the high-speed thermal effects, viscosity and flow separation. These flow solvers need a representative flow field grid.

A procedure needs to be developed to generate 3-D nearly orthogonal grids that use minimal computer resources and attain some degree of smoothing. Although elliptic PDEs generate good grids, they use excessive computer resources. Hyperbolic

PDEs give efficient solutions, but the outer boundary surface can not be specified. Discontinuities on the boundaries may be propagated into the interior of the flow field causing a grid "shock." The desired qualities can be generated from parabolic PDEs. Efficient grids can be generated from the parabolic grid generation equations that will have the desired characteristics of the elliptic grid generation equations. The parabolic grids can be generated to be highly-stretched and orthogonal to the inner boundary with minimal computer resources.

1.3 Problems

Generating a grid around the waverider configuration is challenging. The generalized waverider used in this study has concave, convex and thin lifting body surfaces. The waverider configuration is a stringent test for any grid generation scheme.

In order to generate a valid grid, the inner boundary points must be able to see its corresponding outer boundary point without the geometry's surface getting in the way. Since the waverider is highly-curved, generating the boundary points is not a trivial problem. Wrapping the grid around the waverider's wing tip without grid lines crossing the geometry's surface is another problem. Specifying an orthogonal grid on a concave surface may violate the maximum principle, because all the grid lines will tend to coalesce.

1.4 Objectives

The first objective is to generate a wrapped 2-D nearly orthogonal grid. The first step is to properly distribute points on the boundaries. Next a procedure to generate the grid around the wing tip is developed. One option is for the grids to proceed from the inner to outer boundary in almost a straight line. The other option is for the grid lines to be orthogonal to the waverider's surface. Finally, the orthogonality criterion needs to be relaxed when the inner boundary becomes too concave.

The second and last objective is to generate a wrapped 3-D nearly surface orthogonal grid. This is a simple extension of the 2-D grid procedure. Most of the problems with the waverider's geometry are dealt with in the 2-D grid generation procedure. For a waverider configuration or conical structure, the cross-sections are invariant along a generated ray, so there will not be problems with generating the grid as in the 2-D wrapped grid procedure.

1.5 Overview

The following is a brief summary of what will be discussed in each chapter. Chapter II demonstrates how a waverider is constructed, and how it generates high-performance characteristics. Chapter III gives the elliptic Poisson equations in one, two and three dimensions. The 2-D and 3-D equations are parabolized. The finite difference form for each set of parabolized equations are generated with a procedure on how to use them. Chapter IV explains how the 2-D and 3-D wrapped grids are produced. Chapter V displays the results, and discusses how problems were solved. Chapter VI states the concluding remarks. Chapter VII gives the recommendations. Appendix A discusses a 2-D slit transformed grid procedure. Appendix B is the 2-D wrapped grid computer code. Finally, Appendix C is the 3-D wrapped grid computer code.

II. Waverider Configuration

This Chapter defines a waverider. demonstrates how an infinitesmally thin wing waverider is generated from a known flow field. and develops the 2-D equations of a thick wing waverider in the base plane for extension into three dimensions.

2.1 Definition of a Waverider

Starting from a cone. it is possible to construct high-performance configurations. These configurations are generated by identifying special stream surfaces appropriately as solid surfaces from known flow fields[10]. If steady inviscid flow equations are used to generate flow fields. a stream surface has no flow across it and can be used as a solid surface. The rest of the flow field is inviscid flow past the newly constructed solid surface. Numerous aerodynamic configurations can be constructed by this method. If the resulting upper surface is aligned with the freestream flow. it will generate a freestream pressure on this surface. If the resulting lower surface is bounded by an attached shock. it will generate a pressure associated with the shock on this surface. The combination of these two surfaces will generate a high-performance vehicle. These vehicles appear to ride on a shock wave attached beneath them. therefore they are called waveriders.

2.2 Construction of a Waverider

The waverider used in this study is constructed from a known flow field around an axisymmetric cone. The centerline of the cone must first be aligned with the freestream. The known flow field is constructed by taking the freestream Mach number along with the cone's semi-vertex angle to calculate the inclination of the shock. A streamline starting in the freestream flow will pass through a ray on the shock and remain in a plane as it proceeds asymptotically towards the surface of the cone. Other streamlines that pass through this same ray on the shock will proceed downstream in the same plane. The combination of these streamlines make a plane stream surface that is perpendicular to the cone and the shock boundary as shown in Fig. (2-1).

There are an infinite number of these stream surface planes that pass through the centerline of the cone. Two of these stream surface planes oriented at a positive and negative angle ϕ. form a symmetric pair of lifting surface planes as shown in Fig. (2-2). If the portion of the cone above the lifting surface planes is discarded. the whole upper surface of the waverider is aligned with the freestream. The only place where the shock is attached is along the wing tip of the waverider as shown in Fig. (2-3). In reality. there will still be a shock wave on the upper surface due to boundary layer iteraction. but this surface will still be close to freestream pressure. Thus the waverider has higher pressures on the lower surface and lower pressures on the upper surface. which will generate a high-performance vehicle.

The configuration just developed has infinitesimally thin wings. which is impossible for a realistic configuration. This configuration has a unique design condition for the Mach number and orientation specified. At a different Mach number and orientation the configuration will be off design conditions. where it's performance will be decreased. These configurations are only applicable in the supersonic and hypersonic flow regimes. The sharp nosed. sharp wing tipped waverider is also impractical. because in a hypersonic flow they will melt or oblate. The waverider used in this study was developed through the hypersonic small disturbance theory which idealizes the configuration. as well as making it difficult to analyze at off-design conditions. Therefore. using an computational method on the waverider will allow a more realistic configuration to be analyzed at various flow conditions.

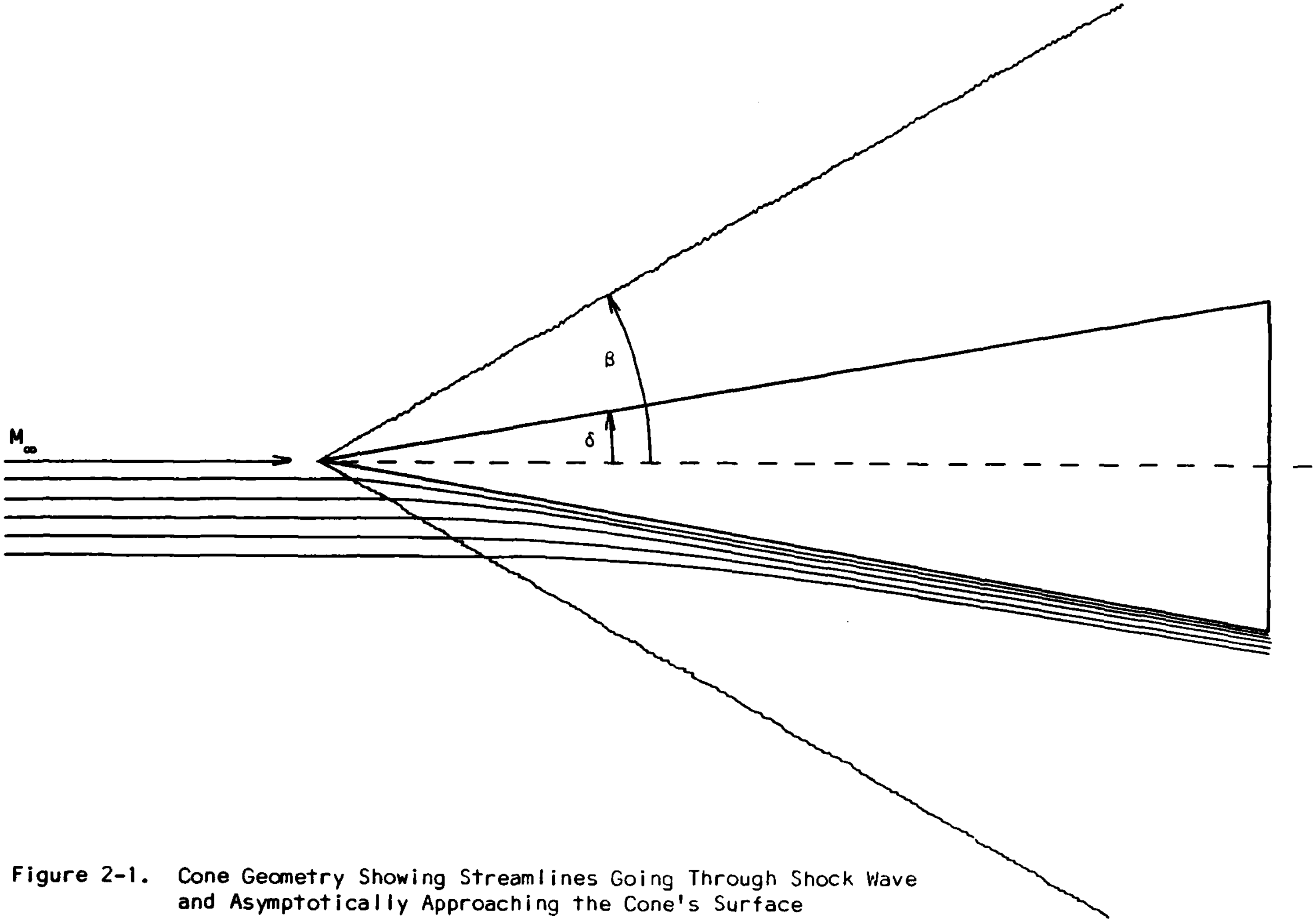


Figure 2-1. Cone Geometry Showing Streamlines Going Through Shock Wave and Asymptotically Approaching the Cone's Surface

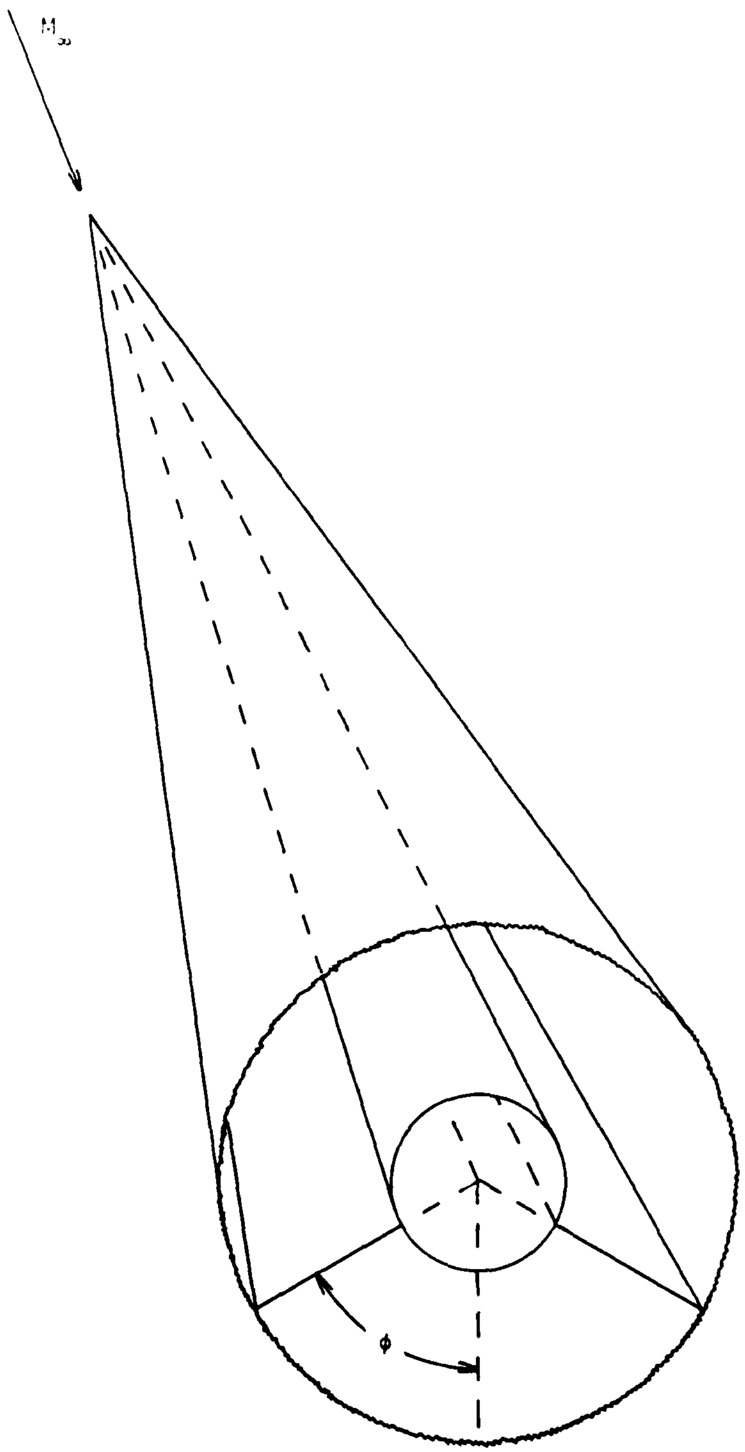


Figure 2-2. Cone Geometry with Symmetrical Plane Stream Surfaces Inclined at Angle Phi

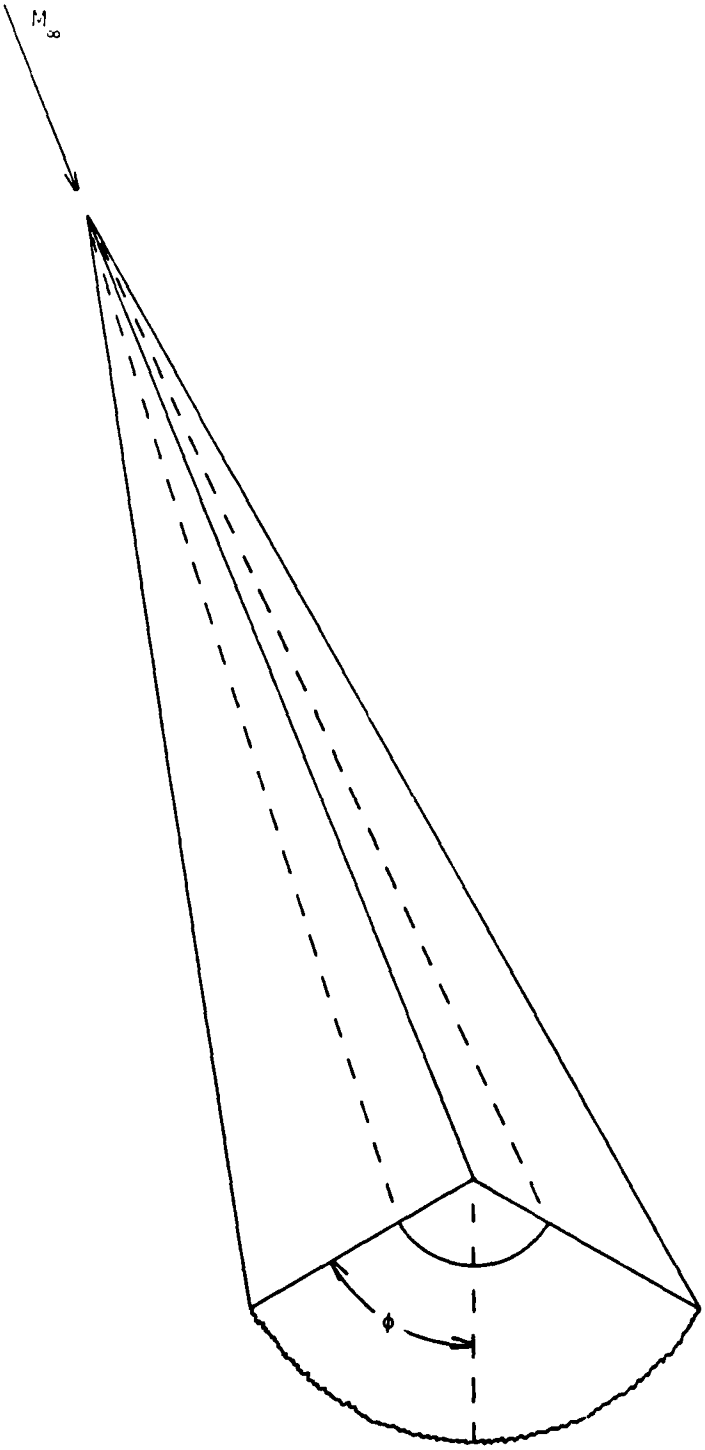


Figure 2-3. Pointed Nose Waverider with Infinitesimally Thin Wings

2.3 Equations to Generate Waverider Geometry

In general, infinitesimally thin wings are not possible, therefore a wing with volume needs to be generated so fuel, propulsive systems, avionics, etc. can fit inside. This problem was studied by Kim, Rasmussen and Jischke[11] where equations were developed to generate the lower and upper surfaces of the waverider in the base plane. The wings are generated with thickness and still retained most of the properties of a waverider. The non-dimensionalized equation that generates the lower or compression surface is given by

$$r_b = \sqrt{\left[\frac{\beta^2}{\delta^2} - \delta_1^2\right]\left[\frac{\phi}{\phi_\ell}\right]^n + \delta_1^2} \qquad (2-1)$$

The non-dimensionalized equation that generates the freestream surface is given by

$$r_{fs} = \frac{\beta}{\delta}\sqrt{\left[\frac{\frac{\beta^2}{\delta^2} - \delta_1^2}{\frac{\beta^2}{\delta^2} - 1}\right]\left[\frac{\phi}{\phi_\ell}\right]^n + \frac{\delta_1^2 - 1}{\frac{\beta^2}{\delta^2} - 1}} \qquad (2-2)$$

The non-dimensionalized 3-D surface of the waverider is generated by scaling the 2-D base plane geometry with respect to cone characteristics. There are two types of waveriders, conical and non-conical. The conical waverider has a sharp nose. A straight line can be drawn from the base plane at the wing tip to the cone's apex. This is where the waverider intersects the shock wave. In both the x-z plane and the y-z plane this intersection is a straight line. A straight line can be drawn from the apex of a cone to any other point on the cone. The x-y plane cross-sections are then scaled according to the distance downstream from the apex. The x and y coordinates are scaled equally for a conical waverider.

For a non-conical waverider the location of the nose has to be determined first. The nose is no longer at the original cone's apex, but at some other position on the shock in the lower plane-of-symmetry. Since the upper surface is aligned with the freestream, the x coordinate of the nose is at the same x coordinate as the upper surface plane-of-symmetry point in the base plane. The waverider is symmetrical, therefore the y coordinate of the nose is equal to zero. The z location of the nose is determined by placing the x and y coordinates into the equation of a cone. A line can be drawn from the base plane at the wing tip to the nose. This line projected into the x-z plane is a straight line. The same line projected into the y-z plane is a parabola. This is shown in Fig. (2-4) and in Fig. (4-3). Since the z locations of the cross-sections are specified by the grid control term R, the x location of these cross-sections can be determined from the straight projected line in the x-z plane. The y coordinate of the shock intersection is determined by substituting the x and z coordinates into the cone equation. The x and y cross section coordinates are determined by scaling the shock intersection coordinates with respect to the base plane coordinates. For a non-conical waverider, the y coordinates are stretched more than the x coordinates near the nose.

In general the symmetric configurations generated by the above equations will not be conical, even though the shock and the flow in the shock layer are asymptotically conical. The waverider will be conical only if the compression surface touches the original cone. This waverider will have a pointed nose and a sharp ridge on the upper surface. The waverider will not be conical if the compression surface does not intersect the original cone. This waverider will have a rounded nose and a rounded corner will replace the sharp ridge on the upper surface. Although the waverider will have a rounded nose in horizontal projected plane, the nose will be sharp in any vertical plane.

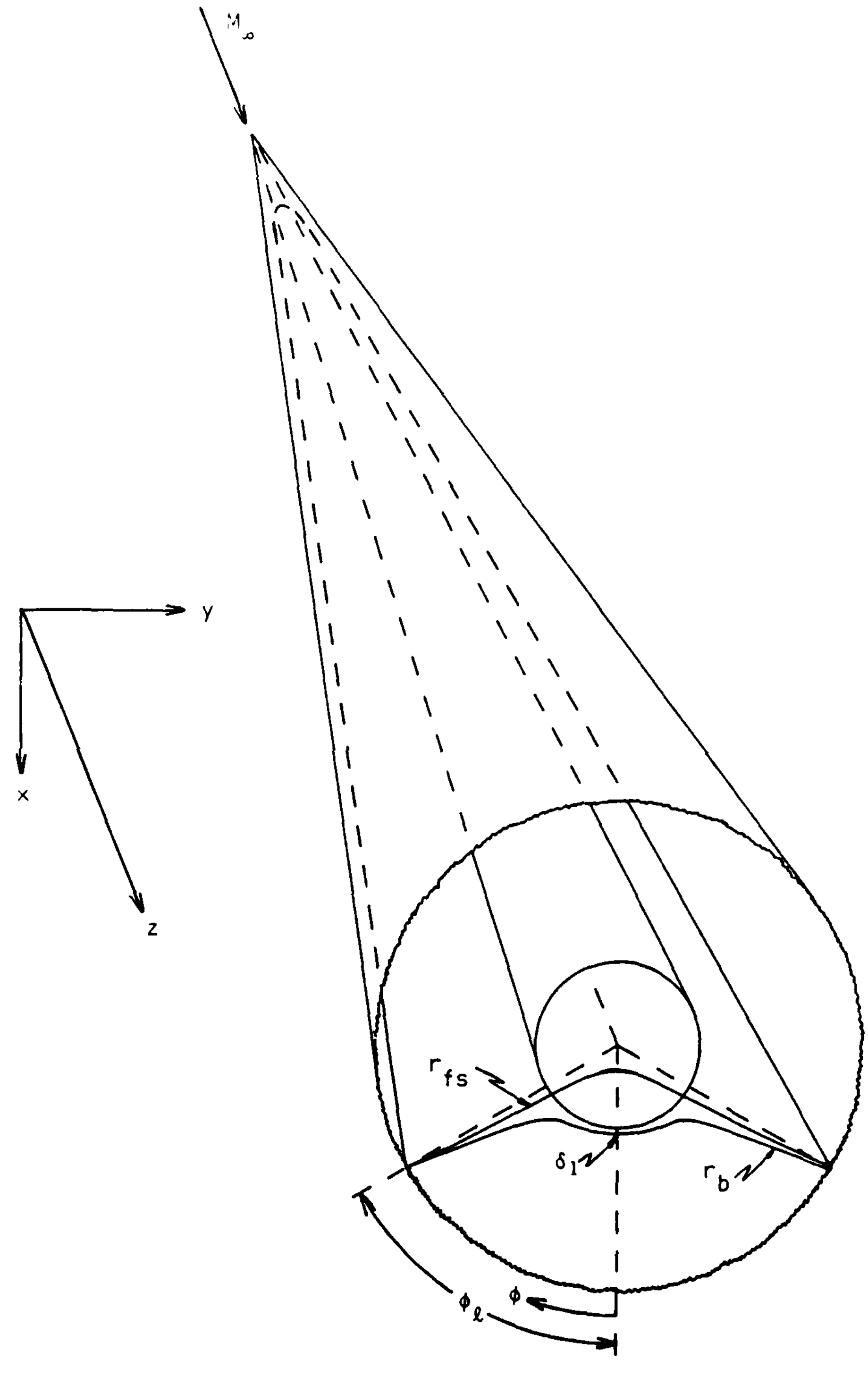


Figure 2-4. Rounded Nose Waverider Configuration

III. Parabolic Grid Equations

This Chapter generates the 2-D and 3-D parabolic grid generation equations. A finite difference equation is generated from the 1-D Poission equation to generate boundary points. The 2-D and 3-D Poisson equations are parabolized to generate the 2-D and 3-D parabolic grid generation equations.

3.1 One-Dimensional Grid Generation Equation

In the physical plane, the linear one-dimensional (1-D) elliptic grid generation equation is given by

$$\xi_{rr} = P(\xi)\, \xi_r^2 \qquad (3-1)$$

This equation must be transformed from physical space into computational space such that there is a one-to-one mapping between the spaces. If the Jacobian does not go to zero the transformation exists. In the computational plane, the linear 1-D elliptic grid generation equation is given by

$$r_{\xi\xi} + P(\xi)\, r_\xi = 0 \qquad (3-2)$$

where the r can be any appropriate coordinate in a Cartesian, cylindrical, or spherical coordinate system. The r can also be in terms of arclength. The source term is not modified for the cylindrical and spherical coordinate systems.

The P term in Eqs. (3-1) and (3-2) is a grid control or source term. If P is equal to zero, the 1-D grid will be evenly spaced. If P is a constant, either positive or negative, the 1-D grid will vary exponentially from one boundary to the other. If P is negative it will exponentially cluster points at the inner boundary and disperse the points at the outer boundary. A positive value of P would cluster points at the outer boundary and disperse points at the inner boundary. By varying the values of P, a grid can be highly clustered at one boundary and constant at the other boundary.This approach generates a grid that can be custom-tailored to a specific type of flow field. The computer codes in this thesis do not have this option. but could easily be modified to include variable source terms.

The numerical solution of Eq. (3-2) is based on an exponential form. The exponential form is based on the Unified Difference Representation (UDR)[11]. If the grid control term P is constant. UDR gives an exact solution for an exponential grid. If the term P is varied. there will be truncation errors in the UDR solution. The magnitude of the truncation errors depends on the size of the variation of P. The truncation error for an exponential grid generated with UDR and a constant grid control term is approximately zero or is on the order of the accuracy of the computer. The finite difference equation using the UDR is given by

$$-\left[\mu(-P) + \mu(P)e^{-|P|}\right] \vec{r}_{i-1} + \left[1 + e^{-|P|}\right] \vec{r}_i - \left[\mu(P) + \mu(-P)e^{-|P|}\right] \vec{r}_{i+1} = \vec{0} \tag{3-3}$$

$$\mu(S) = \begin{cases} 0 \text{ if } S \leq 0 \\ 1 \text{ if } S > 0 \end{cases} \tag{3-4}$$

Equation (3-3) contains two equations. one for P less than or equal to zero and one for P greater than zero. Equation (3-3) requires no iteration. By specifying two boundary values and P. the 1-D grid generation equation can be efficiently solved using a tridiagonal algorithm.

3.2 Two-Dimensional Grid Generation Equations

In the physical plane, the 2-D elliptic grid generation equation is given by

$$\xi_{xx} + \xi_{yy} = P(\xi, \eta)(\xi_x^2 + \xi_y^2) \tag{3-5a}$$

$$\eta_{xx} + \eta_{yy} = Q(\xi, \eta)(\eta_x^2 + \eta_y^2) \tag{3-5b}$$

These equations are transformed into computational space, such that there is a one-to-one mapping between the spaces. A one-to-one mapping is guaranteed for the

Poisson equation using Dirchlet boundary conditions with a positive, negative or zero source term. In the computational plane, the linear 2-D elliptic grid generation equation is given by

$$\alpha_{11}(\vec{r}_{\xi\xi} + P\vec{r}_{\xi}) + \alpha_{22}(\vec{r}_{\eta\eta} + Q\vec{r}_{\eta}) = -2\alpha_{12}\vec{r}_{\xi\eta} \qquad (3-6)$$

The vector r is given by

$$\vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ or } \begin{pmatrix} r \\ \theta \\ z \end{pmatrix} \text{ or } \begin{pmatrix} r \\ \theta \\ \phi \end{pmatrix} \Longleftrightarrow \begin{pmatrix} \xi \\ \eta \\ \varsigma \end{pmatrix} \qquad (3-7)$$

Where the vector r is given by any two coordinates in a right-handed Cartesian, cylindrical or spherical coordinate system as in Eq. (3-7). The vector r can also be written in terms of tangential and normal coordinates. The coefficients in Eq. (3-6) are given by the ijth signed cofactor of the matrix in Eq. (3-10). The grid control term P stretches the grid in the ξ direction, whereas the grid control term Q stretches the grid in the η direction.

$$\alpha_{ij} = \sum_{n=1}^{2} C_{ni}C_{nj} \qquad (3-8)$$

$$C_{k\ell} = (-1)^{k+\ell} M_{k\ell} \qquad (3-9)$$

$$M = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} \text{ or } \begin{pmatrix} r_\xi & r_\eta \\ \theta_\xi & \theta_\eta \end{pmatrix} \qquad (3-10)$$

Two-dimensional grids can be generated from Eq. (3-6), but it requires many iterations to generate a solution. A marching or non-iterative solution can be generated, if the elliptic equations are parabolized. This has already been done by Nakamura[6], Edwards[7] and Noack[8], except this solution will march in the ξ direction and use grid control terms for clustering grid points.

In order to parabolize Eq. (3-6) in the ξ direction, the second derivative in ξ is approximated by a central difference operator and is split into the difference between first derivatives. The first derivative with respect to ξ is approximated by first order directed differences and the derivatives with respect to η are approximated by the UDR. The parabolic approximation equation is given by

$$\alpha_{11}[1+\mu(-P)|P|]\,\vec{r}_{\xi_{i+\frac{1}{2}}} - \alpha_{22}[\vec{r}_{\eta\eta} + Q\vec{r}_{\eta}] = \alpha_{11}[1+\mu(P)|P|]\,\vec{r}_{\xi_{i-\frac{1}{2}}} + 2\alpha_{12}\vec{r}_{\xi\eta} \tag{3 - 11}$$

The two terms on the right hand side of the equation are the source term. These terms are approximated by prescribing points at some I^- location. The points prescribed at the I^+ location are the approximation to the elliptic equations. The I^- location can be fixed or moving at a prescribed interval ahead of the solution. The I^- location can be anywhere between the present solution and the end of the computational domain. The points to be prescribed at I^+ are represented by either a straight line or a nearly orthogonal curve with the proper Q distribution of points.

As in the 1-D solution, the 2-D solution is based on the exponential UDR form. The finite difference equation is given by

$$\begin{aligned}
&\frac{-\alpha_{22}|Q|}{1-e^{-|Q|}}\left[\left[\mu(-Q)+\mu(Q)e^{-|Q|}\right]\vec{r}_{i,j-1} + \left[\mu(Q)+\mu(-Q)e^{-|Q|}\right]\vec{r}_{i,j+1}\right] \\
&+ \left[\alpha_{11}\left[1+|P|+\frac{1}{I^{+}-i}\right] + \frac{\alpha_{22}|Q|}{1-e^{-|Q|}}\left[1+e^{-|Q|}\right]\right]\vec{r}_{i,j} = \\
&\alpha_{11}\left[\left[1+\mu(-P)|P|\right]\vec{r}_{i-1,j} + \frac{1+\mu(P)|P|}{I^{+}-i}\,\vec{r}_{I^{+},j}\right] \\
&+ \frac{\alpha_{12}}{I^{+}-i+1}\left[\vec{r}_{I^{+},j+1} - \vec{r}_{I^{+},j-1} - \vec{r}_{i-1,j+1} + \vec{r}_{i-1,j-1}\right]
\end{aligned} \tag{3 - 12}$$

$$K = \frac{-\alpha_{22}|Q|}{1-e^{-|Q|}} \tag{3 - 13}$$

If Q is equal to zero, the denominator of Eq. (3-13) is zero. When Q is in the vicinity of zero Eq. (3-13) is replaced by

$$K = -\alpha_{22}\left[1 - \frac{|Q|}{2} + \frac{|Q|^2}{6} - \frac{|Q|^3}{24} + \frac{|Q|^4}{120}\right] \text{ if } |Q| < 10^{-6} \tag{3 - 14}$$

Equation (3-14) is a five-term Maclaurin expansion. The $I^{+} - i$ factor scales the approximation to the elliptic equations according to it's distance from the solution point. Since Eq. (3-12) is quasi-linear, it may require iteration to improve the coefficients. Nakamura[6] and Edwards[7] did not iterate their solutions. Noack[8] and Hodge, et al[9] have an option to iterate.

The parabolic grid generation procedure requires that the inner (j=1) and the outer (j=JMAX) boundary surfaces be specified with the proper P distribution of points. To start the 2-D grid generation algorithm from a plane-of-symmetry, the boundary points at i=2 are used by the 1-D grid generation equation to generate either a straight line or orthogonal curve with the proper Q distribution of points. This is the projected solution at I^-. This solution is reflected to the other side of the plane-of-symmetry at i=0 to get the solution at the prior station. The boundary points at i=1 are used in Eq. (3-3) to generate a straight line with the proper Q distribution of points, so a guess at the metrics can be generated. The metrics and coefficients can now be approximated with either a backward or a central differencing technique. The 2-D grid generation equations are solved by a tridiagonal algorithm. In order to march, the projected solution must be recalculated for each marching step. The generation of the metrics and the 2-D grid generation equations are solved for each marching step. At the last solution station, at the plane-of-symmetry, the projected solution is generated by reflecting the previous solution, at IMAX-1, across the plane-of-symmetry. The metrics and coefficients are calculated and the tridiagonal matrix is solved. This step completes the 2-D grid generation procedure.

3.3 Three-Dimensional Grid Generation Equation

In the physical plane, the 3-D elliptic grid generation equations are given by

$$\xi_{xx} + \xi_{yy} + \xi_{zz} = P(\xi,\eta,\zeta)(\xi_x^2 + \xi_y^2 + \xi_z^2) \qquad (3-15a)$$

$$\eta_{xx} + \eta_{yy} + \eta_{zz} = Q(\xi,\eta,\zeta)(\eta_x^2 + \eta_y^2 + \eta_z^2) \qquad (3-15b)$$

$$\zeta_{xx} + \zeta_{yy} + \zeta_{zz} = R(\xi,\eta,\zeta)(\zeta_x^2 + \zeta_y^2 + \zeta_z^2) \qquad (3-15c)$$

These equations are transformed into computational space, assuming there is a one-to-one mapping between the spaces, and are given by

$$\alpha_{11}[\vec{r}_{\xi\xi} + P\vec{r}_{\xi}] + \alpha_{22}[\vec{r}_{\eta\eta} + Q\vec{r}_{\eta}] + \alpha_{33}[\vec{r}_{\varsigma\varsigma} + R\vec{r}_{\varsigma}] = -2[\alpha_{12}\vec{r}_{\xi\eta} + \alpha_{13}\vec{r}_{\xi\varsigma} + \alpha_{23}\vec{r}_{\eta\varsigma}] \qquad (3-16)$$

The vector r is given by Eq. (3-7) in either a Cartesian, cylindrical or spherical right handed coordinate system. The coefficients in Eq. (3-16) are given by the ijth signed cofactor of the matrix in Eq. (3-18). The grid control term P stretches the grid in the ξ direction, the grid contol term Q stretches the grid in the η direction, and the grid contol term R stretches the grid in the ς direction.

$$\alpha_{ij} = \sum_{n=1}^{3} C_{ni}C_{nj} \qquad (3-17)$$

$$M = \begin{pmatrix} x_{\xi} & x_{\eta} & x_{\varsigma} \\ y_{\xi} & y_{\eta} & y_{\varsigma} \\ z_{\xi} & z_{\eta} & z_{\varsigma} \end{pmatrix} \text{ or } \begin{pmatrix} r_{\xi} & r_{\eta} & r_{\varsigma} \\ \theta_{\xi} & \theta_{\eta} & \theta_{\varsigma} \\ z_{\xi} & z_{\eta} & z_{\varsigma} \end{pmatrix} \text{ or } \begin{pmatrix} r_{\xi} & r_{\eta} & r_{\varsigma} \\ \theta_{\xi} & \theta_{\eta} & \theta_{\varsigma} \\ \Phi_{\xi} & \Phi_{\eta} & \Phi_{\varsigma} \end{pmatrix} \qquad (3-18)$$

In order to parabolize Eq. (3-16) in the ξ and ς directions, the second derivatives again must be approximated by a central difference operator and split into the difference between first derivatives. The first derivative with respect to ξ and ς are approximated by first order directed differences and the derivative with respect to η are approximated by the UDR. The parabolic approximation equation is given by

$$\alpha_{11}[1 + \mu(-P)|P|]\,\vec{r}_{\xi}\big|_{i-\frac{1}{2}} + \alpha_{33}[1 + \mu(-R)|R|]\,\vec{r}_{\varsigma}\big|_{k-\frac{1}{2}} - \alpha_{22}[\vec{r}_{\eta\eta} + Q\vec{r}_{\eta}] =$$
$$\alpha_{11}[1 + \mu(P)|P|]\,\vec{r}_{\xi}\big|_{i+\frac{1}{2}} + \alpha_{33}[1 + \mu(R)|R|]\,\vec{r}_{\varsigma}\big|_{k+\frac{1}{2}}$$
$$+ 2[\alpha_{12}\vec{r}_{\xi\eta} + \alpha_{13}\vec{r}_{\xi\varsigma} + \alpha_{23}\vec{r}_{\eta\varsigma}] \qquad (3-19)$$

There are now three terms on the right hand side of the equation. These three terms are the source term that must be approximated. In the 3-D case, points both at the I^+ and K^+ locations must be prescribed.

Again the 3-D solution is based on the exponential form of the UDR. The finite difference equation is given by

$$\frac{-\alpha_{22}|Q|}{1-e^{Q}}\left[\left[\mu(-Q)+\mu(Q)e^{-Q}\right]\vec{r}_{i,j-1,k}+\left[\mu(Q)+\mu(-Q)e^{-Q}\right]\vec{r}_{i,j+1,k}\right]$$

$$+\left[\alpha_{11}\left[1+|P|+\frac{1}{I^{+}-i}\right]+\alpha_{33}\left[1+|R|+\frac{1}{K^{+}-k}\right]\right.$$

$$\left.+\frac{\alpha_{22}|Q|}{1-e^{-Q}}\left[1+e^{Q}\right]\right]\vec{r}_{i,j,k}=$$

$$\alpha_{11}\left[\left[1+\mu(-P)|P|\right]\vec{r}_{i-1,j,k}+\left[1+\mu(P)|P|\right]\frac{1}{I^{+}-i}\vec{r}_{I^{+},j,k}\right]$$

$$+\alpha_{33}\left[\left[1+\mu(-R)|R|\right]\vec{r}_{i,j,k-1}+\left[1+\mu(R)|R|\right]\frac{1}{K^{+}-k}\vec{r}_{i,j,K^{+}}\right]$$

$$+\frac{\alpha_{12}}{I^{+}-i+1}\left[\vec{r}_{I^{+},j+1,k}-\vec{r}_{i-1,j+1,k}-\vec{r}_{I^{+},j-1,k}+\vec{r}_{i-1,j-1,k}\right]$$

$$+\frac{2\alpha_{13}}{(I^{+}-i+1)(K^{+}-k+1)}\left[\vec{r}_{I^{+},j,K^{+}}-\vec{r}_{I^{+},j,k-1}-\vec{r}_{i-1,j,K^{+}}+\vec{r}_{i-1,j,k-1}\right]$$

$$+\frac{\alpha_{23}}{K^{+}-k+1}\left[\vec{r}_{i,j,K^{+}}-\vec{r}_{i,j-1,K^{+}}-\vec{r}_{i,j+1,k-1}+\vec{r}_{i,j-1,k-1}\right] \qquad (3-20)$$

Equation (3-20) is quasi-linear and may require iteration to improve the coefficients.

The 3-D parabolic grid generation procedure is almost identical to the 2-D procedure. The differences include: specifying inner and outer boundary values in three dimensions, having an extra marching direction, more complex metrics and coefficients, and calculating an extra projected solution at K^{+}. This procedure also requires the use of nested loops.

IV. Numerical Procedure

This Chapter explains in detail how the 2-D and 3-D grid generation procedures were constructed. It explains the rationale behind what was done in the computer code, so modification can be accomplished more readily. See Appendix B for 2-D wrapped grid listing. See Appendix C for 3-D wrapped grid listing.

4.1 Two-Dimensional Wrapped Grid Procedure

4.1.1 Program Setup

Key parameters must be specified, before the waverider geometry and the grid can be generated. Subroutine SETUP controls 21 of these parameters. SETUP initially specifies all 21 parameters. The user can interactively modify any of the parameters through a menu generated at the computer terminal.

The shock angle needs to be calculated before the waverider geometry can be generated. The shock angle is used as one parameter in Eqs. (2-1) and (2-2) to generate the base plane geometry of the waverider. Subroutine SHOCK uses the Mach number and cone's semi-vertex angle to iterate on the transcendental shock relation. An initial guess of the shock angle is generated by

$$\beta = \sin^{-1}\sqrt{\frac{\frac{\gamma+1}{2}\frac{M_\infty^2}{\sqrt{M_\infty^2-1}}\delta + 1}{M_\infty^2}} \qquad (4-1)$$

The transcendental shock relation is given by

$$\beta_{n+1} = \sqrt{\frac{\gamma+1}{2}\frac{\sin\beta_n \sin\delta}{\cos(\beta_n-\delta)} + \frac{1}{M_\infty^2}} \qquad (4-2)$$

Equation (4-2) is solved iteratively, starting with the initial guess from Eq. (4-1).

4.1.2 Boundary Points

In order to properly distribute points on the curved boundaries, the arclength must be specified in Eq. (3-3). Because the waverider is highly-curved, it is impractical to distribute points by specifying an increment on one of the axes. This process does not take into account the shape of the geometry, and would result in an erratic distribution of points over the waverider surface. If the points are distributed over the curved surface according to arclength, a smooth distribution of points will result.

A table of angles, radii, and arclengths must be constructed to convert the distributed arclengths into cylindrical coordinates. Subroutine ARCLEN generates this table. ARCLEN starts by calling subroutine GEOM, which calculates the radius of the lower surface of the waverider, the radius of the upper surface of the waverider, and the radius of the outer boundary ellipse at zero degrees. The angle is then incremented by one degree and the three radii are re-calculated at the incremented angle. When ARCLEN has two points from each set of radii, it calculates the arclength by determining the distance between the two points. At the third point and beyond, ARCLEN calculates the distance between the two points from each set of radii and adds it to the previous total arclength. This process is continued until the dihedral angle is reached, where the table of ϕ, radius of lower surface, radius of upper surface, arclength of lower surface, and arclength of upper surface is completed. The table for the outer boundary is continued until the angle of 180 degrees is reached.

The arclength of the outer boundary ellipse and the grid control term P are used, in Eq. (3-3), to generate the points on the outer boundary. The two boundary values needed are zero for the beginning of the outer boundary ellipse and the arclength for the end of the outer boundary ellipse. The coefficients of Eq. (3-3) are generated. These coefficients are used in subroutine TRIDIG, which implicitly solves the equation with a tridiagonal solver. TRIDIG outputs IMAX points with the proper P distribution of arclength. The distributed arclengths are input into subroutine INTERP, where corresponding values of r and ϕ are interpolated. INTERP searches for the distributed arclength in the table for the outer boundary ellipse. A linear interpolation is done between the points stored in the table on both sides of the distributed arclength, to calculate r and ϕ for a point on the outer boundary ellipse. The boundary points are then converted from cylindrical coordinates into Cartesian coordinates. This conversion is necessary, because the origin of the system does not reside within the waverider. This problem will be discussed in detail in Section 5.1.1.

The number of points to be placed on the two surfaces of the waverider needs to be calculated. Section 5.1.1 discusses three different methods for calculating the number of points to be placed on each surface. The best method found is discussed here. A point on the upper surface is retreived from the ARCLEN table at five degrees less than the dihedral angle, and the same for the lower surface. These two points are averaged to generate a point that lies between the two surfaces. The equation of a line is generated using the averaged point and the point on the wing tip. The intersection of this bisecting line and the outer boundary ellipse is calculated. The intersecting point is then compared with the points distributed on the outer boundary ellipse. The position number of the closest outer boundary point with the intersecting point is calculated. This position number is equal to the number of points to be placed on the lower surface, and IMAX minus the position number plus one is the number of points to be placed on the upper surface. The upper and lower waverider surfaces intersect at the wing tip. These two surfaces are kept separate until the proper distribution of points are placed upon them.

The most difficult task in generating a 2-D grid is generating a good solution around the sharp wing tip of the waverider. This difficulty will be discussed in detail in Seltion 5.1.2. Grid spacing refinement around the wing tip proved to be a solution to this problem[13]. A specified even number of points have their spacing refined near the wing tip. These points do not lie on top of each other. but at a closely spaced interval. If all the refined points were on top of each other the maximum principle would be violated on the boundary. A valid grid can be generated with the maximum principle violated on the boundary. but it is easy to avoid this situation. The number of points on the upper and lower surfaces are decreased by half the specified number of refined points. The adjusted number of points will be used to generate the proper P distribution of points on the upper and lower surfaces of the waverider with respect to arclength. The refined points are automatically placed on the wing tip disregarding the grid control distribution. Subroutine INTERP evenly spaces the refined points at 1/10.000 of a unit of arc away from the wing tip on the upper and lower surfaces of the waverider.

The adjusted number of points for each surface are used to generate Cartesian coordinates for the waverider's surface. With the two boundary values along with the grid control term P. the coefficients in Eq. (3-3) are generated. The two boundary values are zero for the beginning of both upper and lower surfaces. and the total arclength of either the upper or the lower waverider surface. A tridiagonal matrix is solved by subroutine TRIDIG for the upper surface and another for the lower surface. The proper distribution of points with respect to arclength are output. Subroutine INTERP does a linear interpolation on the distributed arclengths to get Cartesian point values for the boundary of the upper and lower surfaces of the waverider. The boundary values of the upper and lower surfaces of the waverider are now stored in one array as the inner boundary points. The outer boundary ellipse points are stored in another array as the outer boundary points. The boundary points are now complete.

4.1.3 One-Dimensional Elliptic Equation with Q Distribution

Equation (3-3) must be solved repeatedly for each projected solution station and each guess at the present solution station, so the metrics and the coefficients in Eq. (3-12) can be generated. The grid control term Q is used to distribute points between the boundaries. Q is constant in this program, therefore the tridiagonal solver will similarly distribute points over a different interval. The solution is stored and interpolated whenever the 1-D distribution of Q points is needed.

Again, it is easy to modify the program so the source terms are variable. This allows the points to be distributed with varying density throughout the grid. For example, if the waverider where at angle-of-attack, many points are desired in the boundary layer on the lower surface, while fewer are desired in the boundary layer on the upper surface. The variable grid control terms would give this result, but would require that the tridiagonal solver be used every time the projected solution station or a guess at the present solution station needs to be generated.

4.1.4 Plane-of-Symmetry Conditions

The station at i=1, on the lower plane-of-symmetry, is where the 2-D parabolic grid generation procedure starts. To this point only the inner and outer boundary points have been determined, along with the desired Q distribution of points to be placed between the boundaries. The 2-D procedure starts by scaling 1-D stored Q distribution of points with the the boundary points at i=2 to generate the projected solution or the approximation of the elliptic equations at $i + 1^{+}$. The projected solution is reflected over to the other side of the plane-of-symmetry at i=0 to generate the prior solution at i-1. The projected solution can either be the scaled straight line stored Q distribution of points, or can be a nearly orthogonal curve. The nearly orthogonal curve will have an approximate distribution of points with respect to Q. The details on how to generate the nearly orthogonal curve are discussed in Section 4.1.6. The boundary points at i=1 are scaled with the straight line stored Q distribution of points solution to generate a guess at the solution station. Enough information has now been determined to start the 2-D parabolic grid generation procedure.

One of the parameters set in subroutine SETUP is whether second-order central differencing or first-order backward differencing be used on the metrics. The solution at i=1 is at the plane-of-symmetry, therefore the solution should be a straight line between the boundaries. Second-order central differencing is used at this plane-of-symmetry station to get this straight line result. The projected solution at the plane-of-symmetry must be one station downstream of the solution station to acheive this result. The metrics and coefficients are calculated and the tridiagonal matrix is solved to generate the solution at the plane-of-symmetry at the beginning of the grid.

The station at i=IMAX is at the other plane-of-symmetry station and is also the last solution station. A guess at this solution station needs to be made by scaling the stored 1-D Q distribution of points. The solution at i=IMAX-1, the prior solution, is reflected over to the other side of the plane-of-symmetry to generate the projected solution at $i + I^{-}$. The 2-D procedure is completed by calculating the metrics with central differences and solving the the tridiagonal matrix.

4.1.5 Check on Convexity

The 2-D parabolic grid generation procedure allows the projected solution to be anywhere between the present solution and the end of the computational domain. This can lead to problems if the surface is exceedingly convex and the projected solution is too far from the solution station. This problem is discussed in more detail in Section 5.1.3. A test is done to see if the projected solution is too far downstream when there is a lot of curvature on the inner boundary. A line is generated from the j=2 point of the prior solution at i-1. to the j=2 point of the projected solution at $i + 1^*$. If this line intersects the waverider geometry, the surface is too convex and the projected solution position is not at an acceptable position for this solution station. If this line does not intersect the waverider geometry, the projected solution is at an acceptable position. These two situations are shown in Fig. (4-1).

This criterion is tested by checking each station between the projected solution and the solution station to see whether the line passed through the inner boundary. The boundary points between the projected solution and the solution station are used to generate the equations of the lines between the boundaries. The intersection of the line between the j=2 nodes of the prior and projected solutions and the lines between the boundaries are calculated. A check is done to see whether the intersecting points lie between the inner and outer boundaries. If all the intersections lie between the boundaries, the projected solution position is acceptable. If one or any number of the intersecting points do not lie between the two boundaries, the projected solution is too far downstream and is forced back one position closer to the solution station. A new line is generated between j=2 of the prior solution and j=2 of the new projected solution. The intersections are re-calculated and checked to see if they lie between the boundaries. This procedure is continued until all intersecting points lie between the boundaries.

The proper projected solution location is now determined. The stored 1-D Q distribution of points is scaled between these boundary points to generate the straight line projected solution. This type of projected solution does not yield grid lines that are nearly orthogonal to the waverider's surface, but yield grid lines that almost run straight from one boundary to the other. How straight the grid lines run depends on how much smoothing is used to generate the grid. The straight line projected solution is used in generating the orthogonal projected solution, so it

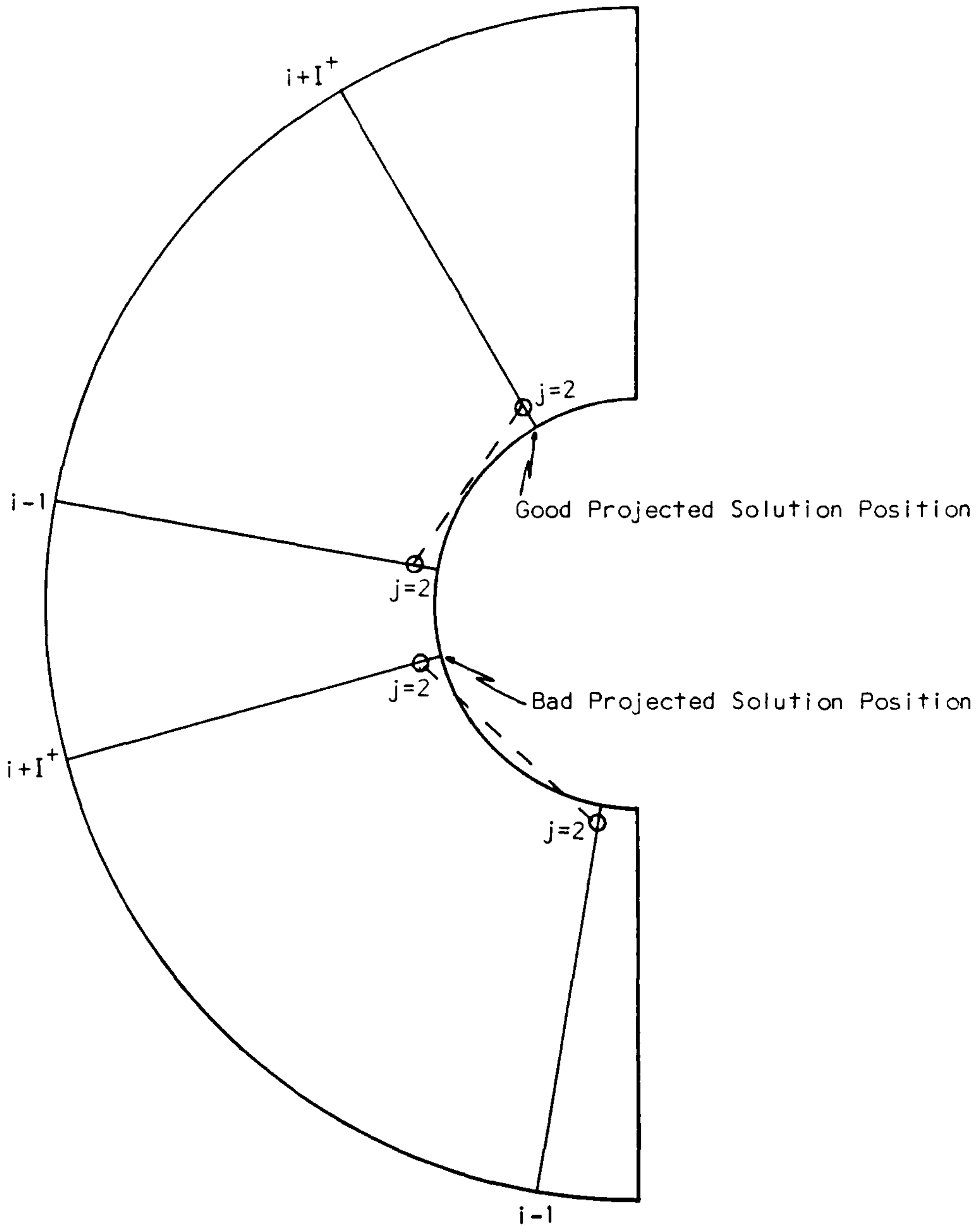

i+I$^+$
j=2
Good Projected Solution Position
i-1
j=2
j=2
Bad Projected Solution Position
i+I$^+$
j=2
i-1

Figure 4-1. Criterion for Good and Bad Projected Solution Positions

must be calculated regardless of whether orthogonality is specified or not.

4.1.6 Orthogonal Grid Generation

Many of the flow solvers that use flow field grids require that the grid be orthogonal to the inner boundary. This zeros out the partial derivative of the pressure with respect to the normal direction. A nearly orthogonal grid is generated by making the projected solution orthogonal to the inner boundary. The projected solution slowly bends and makes its way back to the outer boundary point. This is accomplished by scaling the angle of a line normal to the waverider's surface and a line drawn between the inner and outer boundary points. This scaled angle in combination with the proper distance from the inner boundary surface gives a point on the nearly orthogonal projected solution. The proper distance from the inner boundary is dictated by the straight line projected solution.

The angle of the line between the inner and outer boundary points is generated. The angle equals the arctangent of the slope of the line between the boundary points at the I^+ location. A problem may arise if the orthogonal projected line will lie in one quadrant and the line between the boundary points will lie in another quadrant. If the angles are not put into the proper quadrants, the orthogonal projected curve may not be generated correctly. The proper quadrant is identified by calculating x and y quadrant factors, which are given by

$$\text{x factor} = \frac{x_{2_{I^+}} - x_{1_{I^+}}}{|x_{2_{I^+}} - x_{1_{I^+}}|} \qquad (4-3a)$$

$$\text{y factor} = \frac{y_{2_{I^+}} - y_{1_{I^+}}}{|y_{2_{I^+}} - y_{1_{I^+}}|} \qquad (4-3b)$$

If the x and y factors are both positive, the angle is in the first quadrant. If the x factor is negative and the y factor is positive, the angle is in the second quadrant. If the x and y factors are both negative, the angle is in the third quadrant. If the x factor is positve and the y factor is negative, the angle is in the fourth quadrant.

The angle of the line normal to the waverider's surface at the I^- location is generated. The true normal to a paneled surface, in the limit, is approximately the average angle generated from the normal lines of the panels on each side of the I^- location. The arctangent of the negative inverse of the slope of one panel and the arctangent of the negative inverse of the slope of the other panel are averaged together to get the normal angle to the surface. This angle must also be put into the correct quadrant. This is more difficult, because only a point and an angle are known. This is enough information to generate a line. The intersection of this line with the outer boundary is calculated as in Section 4.1.2. The only difference is both boundary intersections need to be generated. The boundary intersection point closest to the I^- outer boundary point is the proper point that will not intersect the geometry. The I^- inner boundary point and the intersection point are used in Eq. (4-3) to generate the x and y quadrant factors. From these two quadrant factors the quadrant of the normal line can be identified. If the angle of the normal line is in the first quadrant and the angle of the line between the boundaries is in the fourth quadrant or vice versa, there will be a problem. This problem is alleviated by subtracting 360 degrees from the angle in the fourth quadrant, so the nearly orthogonal curve can be generated in the first and fourth quadrant. If 360 degrees were not subtracted, the orthogonal curve would be generated through all the quadrants, which yields an incorrect projected solution.

To generate the orthogonal projected solution, an angle has to be varied from the normal line to the line between the boundaries. The hyperbolic tangent function is used to vary this angle. If the normal line and the line between the boundaries are close together, the projected solution can remain orthogonal longer without the danger of grid crossover. If these two lines differ greatly in angle, the projected solution can leave the surface orthogonal, but must bend towards the I^+ outer boundary point quickly so not to cause grid crossover. The hyperbolic tangent has these qualities. If the hyperbolic tangent is in the interval from one-half to minus one-half, it will approximately generate a straight line. In this case, the hyperbolic tangent will vary the angle swiftly as shown in Fig (4-2). If the hyperbolic tangent is in the interval from five to minus five, it stays near one for awhile then swiftly moves to minus one. This type of distribution would cause the projected solution to be orthogonal to the surface longer before it proceeds to the outer boundary point at the I^- location and is shown in Fig (4-2). The factor that governs rate of change of the angular distribution is given by

$$\text{speed factor 1} = \frac{1}{|\phi_2 - \phi_1|} \text{ in radians} \qquad (4-4)$$

If the two lines are close together, the factor will be large. If the two lines are far apart, the factor will be small.

The hyperbolic tangent function is centered somewhere between the two boundary points. The specified centering position is where the varied angle will bisect the normal line and the line between the boundaries. If the centering is positioned too close to the outer boundary, a concave surface may cause the grid lines to coalesce and zero out the Jacobian. The centered value is based on lengths and can range from zero to one. The centering value is the length from the inner boundary to the center point on the straight line projected solution, normalized by the length of the line between *the inner and* outer boundary points.

The nearly orthogonal projected solution is generated from the angle of the normal line, the angle of the line between the boundaries, the speed factor and the centering position. The angle that varies the angle of the normal line to the angle of the line between the boundaries is given by

$$\phi_3 = \frac{\frac{\tanh\phi}{\text{factor}}(\phi_2 - \phi_1) + \phi_2 + \phi_1}{2} \qquad (4-5)$$

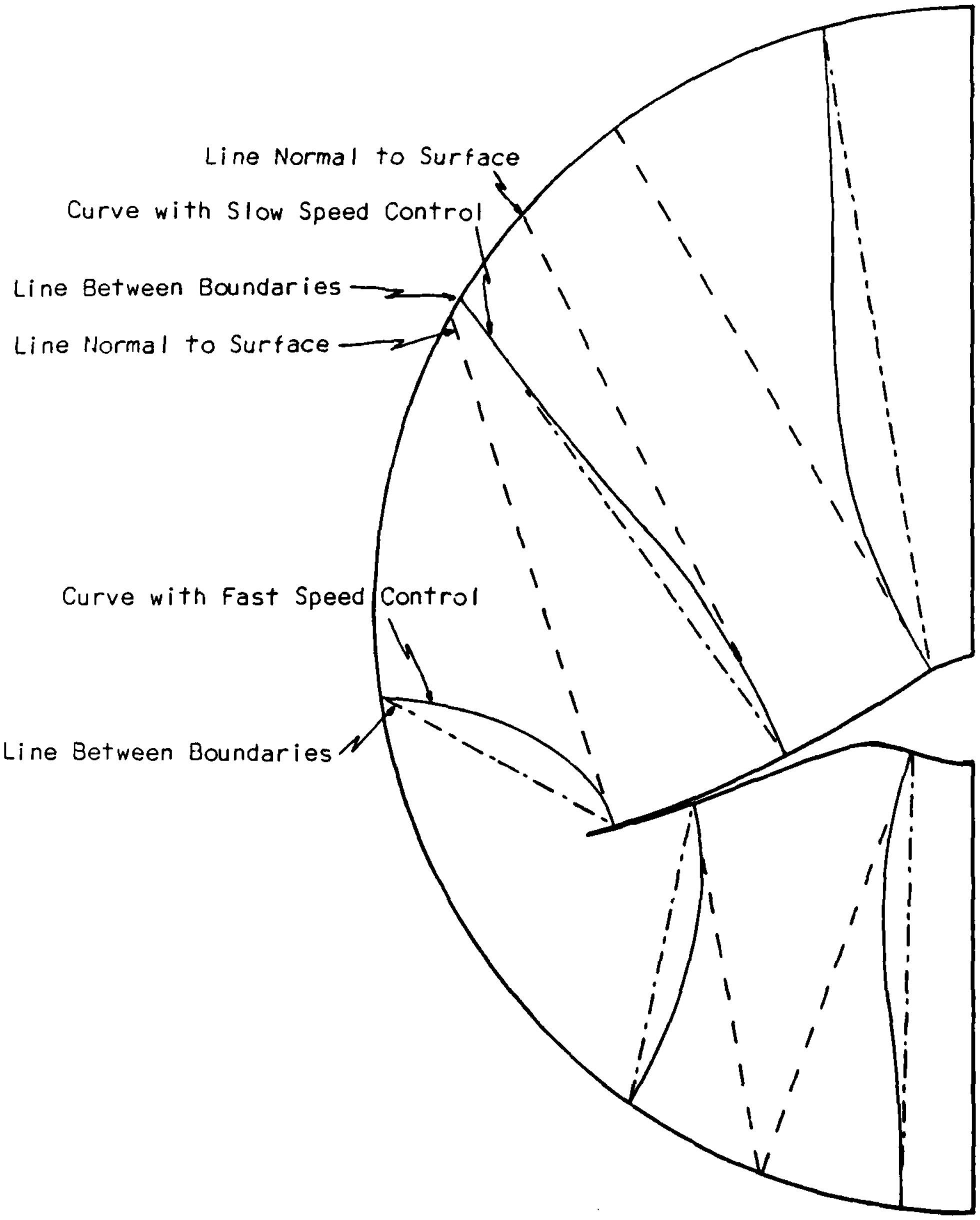


Figure 4-2. Effect of Hyperbolic Tangent on the Generation of a Nearly Orthogonal Curve

The angle ϕ_1 is that of the angle of a straight line connecting the inner and outer boundary points at the I^- location, and ϕ_2 is the angle of the normal line at the I^- location. The term $\frac{\tanh\phi}{\text{factor}}$ generates values from one to minus one, making ϕ_3 range from ϕ_2 to ϕ_1. The angle ϕ is varied slowly when points are clustered and quickly when points are dispersed. The angle ϕ is given by

$$\phi = \frac{\sqrt{(x_{I^-,j} - x_{1_{I^-}})^2 + (y_{I^-,j} - y_{1_{I^-}})^2}}{|\phi_2 - \phi_1|\sqrt{(x_{2_{I^-,j}} - x_{1_{I^-}})^2 + (y_{2_{I^-,j}} - y_{1_{I^-}})^2}} - \frac{\sqrt{(x_{I^+,jcent} - x_{1_{I^+}})^2 + (y_{I^+,jcent} - y_{1_{I^+}})^2}}{|\phi_2 - \phi_1|\sqrt{(x_{2_{I^+,j}} - x_{1_{I^+}})^2 + (y_{2_{I^+,j}} - y_{1_{I^+}})^2}} \qquad (4-6)$$

The speed factor term in Eq. (4-4) governs how the resulting angle will vary. The denominator in Eq. (4-6) is the distance between boundaries at the I^+ location. The first term in the numerator is the distance from the inner boundary to the projected solution point being calculated on the straight line projected solution. Since the hyperbolic tangent is centered in the interval, the term factor in Eq. (4-5) gives a value equal to the $\tanh\phi$ at the first station and slowly varies to the $-\tanh\phi$ at the last station. The term j is incremented from two to JMAX-1. The term factor is given by

$$\text{factor} = \frac{\text{JMAX} - 1 - j}{\text{JMAX} - 3}\tanh\phi^1 + \left[\frac{j - \text{JMAX} + 1}{\text{JMAX} - 3} + 1\right]\tanh\phi^{\text{JMAX}-1} \qquad (4-7)$$

The angle ϕ_3, along with the first term in the numerator of Eq. (4-6), defines the point location on the nearly orthogonal projected solution.

A problem arises in generating orthogonal curves near the wing tip. It would be difficult to trust a value of ϕ_3 as generated above, because the refined points near the wing tip are so close together. If the projected solutions at the refined points are generated nearly orthogonal to the inner boundary, the grid that will be generated near the wing tip will have a big gap in it. Therefore, the first refined point encountered on the lower surface uses the same value of ϕ_2 as at the station directly upstream. The last refined point on the upper surface uses the same value of ϕ_2 as at the station directly downstream. The number of remaining refined points plus two are divided into the difference in ϕ_2 of the first and last refined point. This is the interval that will be added to the ϕ_2 value of the first refined point to each consecutive refined point between the first and last refined points. The calculation of ϕ_1 and the rest of the nearly orthogonal curve generation is the same as before.

4.1.7 Grid Crossover Check

The maximum principle may be violated when generating a nearly orthogonal grid in a concave region. There are several ways of controlling grid crossover and they will be discussed in more detail in Section 5.1.4. The automatic procedure will be discussed here. The equations of a line at the I^+ projected solution position and at $I^- - 1$ are generated between the boundaries. The intersection of these two lines is calculated. The distance between the intersection point and the inner boundary point is determined at I^- and divided by the total distance between the I^+ boundary points. A valid result gives a value between zero and one. The calculated centering value is where the hyperbolic tangent should be centered to eliminate grid crossover. If the centering value is closer to the inner boundary than the user specified centering value, it will be locally inserted to control grid crossover. Even though the maximum principle is not violated, this procedure will still produce grid lines that are fairly close together if the boundary is highly-concave.

4.1.8 Two-Dimensional Parabolic Grid Equation Procedure

The 2-D parabolic grid generation procedure solves Eq. (3-12) with a tridiagonal solver. Before the tridiagonal solver can be initiated, the metrics and the coefficients have to be determined. First a guess at the solution station has to be generated for use in determining the metrics. This is accomplished by scaling the stored 1-D Q distribution or points between the boundary points at the solution station. The metrics can be generated by using either a second-order central difference or a first-order backward difference. At the plane-of-symmetry stations, the metrics are calculated by the central difference. The central difference will yield a straight line solution at the plane-of-symmetry. At all other solution stations, the metrics are generated by the backward difference. If the projected solution is one station downstream of the solution station and central differencing has been specified, central differences will be used to generate the metrics. The coefficients in Eq. (3-12) are determined by combining the metrics properly. The coefficients are sent into subroutine TRIDIG, were the 2-D parabolic equation is solved. This process has to be done twice, once for the x coordinate and once for the y coordinate.

The program allows the solution to be iterated. This is done by recalculating the metrics using the newly solved solution instead of the initial guess. If central differencing is used on the metrics, iteration is unnecessary, because the solution station values are not used in determining the metrics. After the solution station is solved, the values are written into an output file. The solution station values are then rotated back into the prior solution position, and the next solution station is solved.

The following is a summary of the procedure for generating 2-D parabolic grids:

1) Generate the outer boundary points with 1-D grid generation equation, Eq. (3-3), with P distribution and with IMAX points.
2) Calculate the number of points needed to go on lower and upper surface of the waverider.
3) Generate inner boundary points with 1-D grid generation equation, Eq. (3-3), with P distribution and with the number of points calculated for each surface.
4) Generate and store solution to 1-D grid generation equation, Eq. (3-3), with Q

distribution and JMAX points.

5) Initialize 2-D grid generation process by scaling the solution from step 4 between the plane-of-symmetry boundaries (i=1) and between the i=2 boundaries. The solution at i=2 can be a straight line or can be put into nearly orthogonal form for a nearly orthogonal grid. The straight line or orthogonal solution at i=2 is reflected over the plane-of-symmetry at i=0 to create the prior solution.
6) The projected solution is calculated with a straight line or a nearly orthogonal curve at the I^- location. At the plane-of-symmetries the projected solution is calculated at i=2 or i=IMAX+1, otherwise at a good projected solution position.
7) The metrics and coefficients are generated and can be approximated by either a first-order backwards difference or a second-order central difference. A second-order difference is used at the plane-of-symmetries.
8) The 2-D parabolic grid generation equation given by Eq. (3-12) is solved by a tridiagonal algorithm.
9) The present solution is rotated back into the prior solution position for the next marching step.
10) Steps 6 through 9 are repeated until the other plane-of-symmetry is reached.
11) At the last plane-of-symmetry the prior solution is reflected across the plane-of-symmetry to create the projected solution and the process is completed by doing steps 7 and 8.

4.2 Three-Dimensional Wrapped Grid Procedure

The 3-D wrapped grid procedure is a relatively simple extension of the 2-D wrapped grid procedure. The highly-curved surface of the waverider only occur in the x-y planes. The waverider is wedged-shaped in the x-z plane, and parabolic shaped in the y-z plane. This is shown in Fig. (4-3). The geometry is invariant along a generated ray in the x-z and y-z planes. Therefore, there will not be problems with the positioning of the projected solution, generating the grid around sharp corners, or the nearly orthogonal grid lines coalescing in concave regions in the added dimension. Most of the procedures in 3-D are the same as in 2-D; therefore, only the new techniques will be discussed in detail here.

4.2.1 Boundary Points

The 3-D program initialization routines are the same as the 2-D procedure, except for the generation of the waverider geometry. Before the waverider geometry can be generated in three dimensions, the waverider nose has to be located, the length has to be calculated, and the ς locations of the cross sections have to be determined. The length of the waverider is determined by dividing the tangent of the shock angle into the non-dimensional distance from the z axis to the shock wave in the base plane. Since the upper surface of the waverider is aligned with the freestream flow and the nose is in the plane-of-symmetry, the x and y coordinates of the nose are known. The z coordinate of the nose is determined by putting the x and y coordinates of the nose into the cone equation. Since the upper surface is parallel with the z axis, the arclength is not needed to distribute points in the ς direction. The z coordinates of the nose and the base plane, along with the grid control term R are used to generate the coefficients of Eq. (3-3). These coefficients are input to subroutine TRIDIG, which implicitly solves the equation with a tridiagonal solver. TRIDIG outputs the z locations of the cross sections in the ς direction with respect to the grid control term R.

The waverider geometry can now be generated in three dimensions. The waverider geometry is generated in the base plane as in the 2-D wrapped grid procedure. This procedure is referred to in Section 4.1.2. The geometry generated in the base plane is stored for scaling the other cross sections upstream of the base

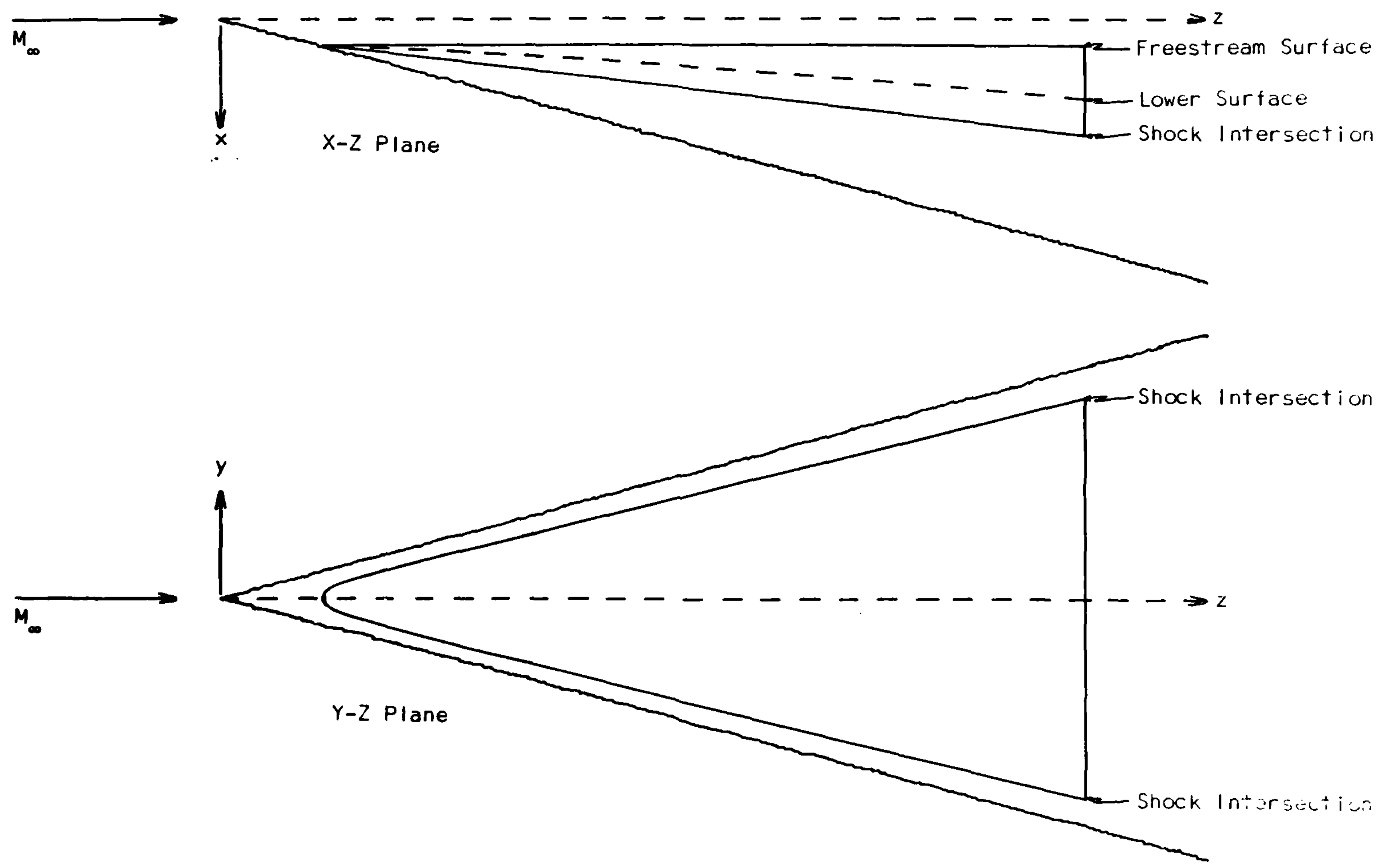


Figure 4-3. Waverider Shock Intersections in X-Z Plane and Y-Z Plane

plane. The scaling factors depend on where the wing tip will intersect the shock on each cross section upstream of the base plane. The equation of a line is generated in the x-z plane between the nose and wing tip in the base plane. The x location of the cross section's wing tip can be generated by putting the z coordinate of each cross section into the equation of the line. Figure (4-3) shows the location of the shock intersection in the x-z plane. The x and z coordinates of the wing tip are put into the cone equation to generate the y coordinate of the wing tip. Figure (4-3) also shows the location of the shock intersection in the y-z plane. These three coordinates are where each cross section intersects the shock. The factors that scale the base plane waverider geometry are given by

$$\text{xz factor} = \frac{\text{XSHOCK}_k - \text{XNOSE}}{\text{XSHOCK}_{\text{KMAX}} - \text{XNOSE}} \tag{4 – 8a}$$

$$\text{yz factor} = \frac{\text{YSHOCK}_k}{\text{YSHOCK}_{\text{KMAX}}} \tag{4 – 8b}$$

$$\text{outer boundary factor} = \text{yz factor} \tag{4 – 8c}$$

The x location of the nose has to be subtracted from the xz scaling factor if it does not reside on the axis. The outer boundary scaling factor scales the semi-major and the semi-minor parameters that generate the outer boundary ellipse.

A non-conical waverider has a rounded nose and is parabolic in the y-z plane. A small problem arises when generating the 3-D outer boundary. A straight line can not be drawn from the nose of the waverider to the outer boundary near the wing tip in the base plane without crossing the waverider's surface. The scaling factor has to force the outer boundary to stay the same relative distance away from the wing tip at all the cross sections. Both the x and y coordinates of the outer boundary are scaled the same way to simplify the geometry. The geometry is stored as inner and outer boundary points. One cross section downstream of the base plane is generated and stored for use in generating the metrics and coefficients for the base plane solution. The z location of the inner and outer boundary points at each cross section are the same.

Equation (3-3) is solved and stored using the grid control term Q and JMAX points. As stated in Section 4.1.3. the stored solution is used for generating the projected and present solutions. It is more efficient to store this result in 3-D. because it will be used approximately IMAX by KMAX times.

4.2.2 Three-Dimensional Starting and Stopping Procedures

The 2-D grid generation procedure used three grid lines to generate the solution. The 3-D grid generation procedure uses three grid lines in the solution cross section. three grid lines in the cross section just upstream of the solution cross section. and three grid lines in the downstream projected cross section. The finite differencing scheme is shown in Fig. (4-4).

The plane-of-symmetry conditions are the same in the x-y plane as in the 2-D wrapped grid procedure. These plane-of-symmetry conditions were discussed in Section 4.1.4. Starting and stopping the solution in the ς direction is different than the plane-of-symmetry conditions for the 2-D grid generation procedure. To simplify this procedure three 2-D arrays are used. one at the prior solution plane at k-1. one at the solution plane at k. and one at the projected solution plane at K^{+}. The prior solution plane contains known or previously calculated grid lines. The solution and projected solution planes are intitially filled with nearly orthogonal curves that are an estimated grid. The 3-D grid generation procedure starts at the second cross section. At the start up. the prior solution plane contains the nose coordinate. The 3-D grid generation procedure concludes with a solution in the base plane. The projected solution plane uses extrapolated boundary values.

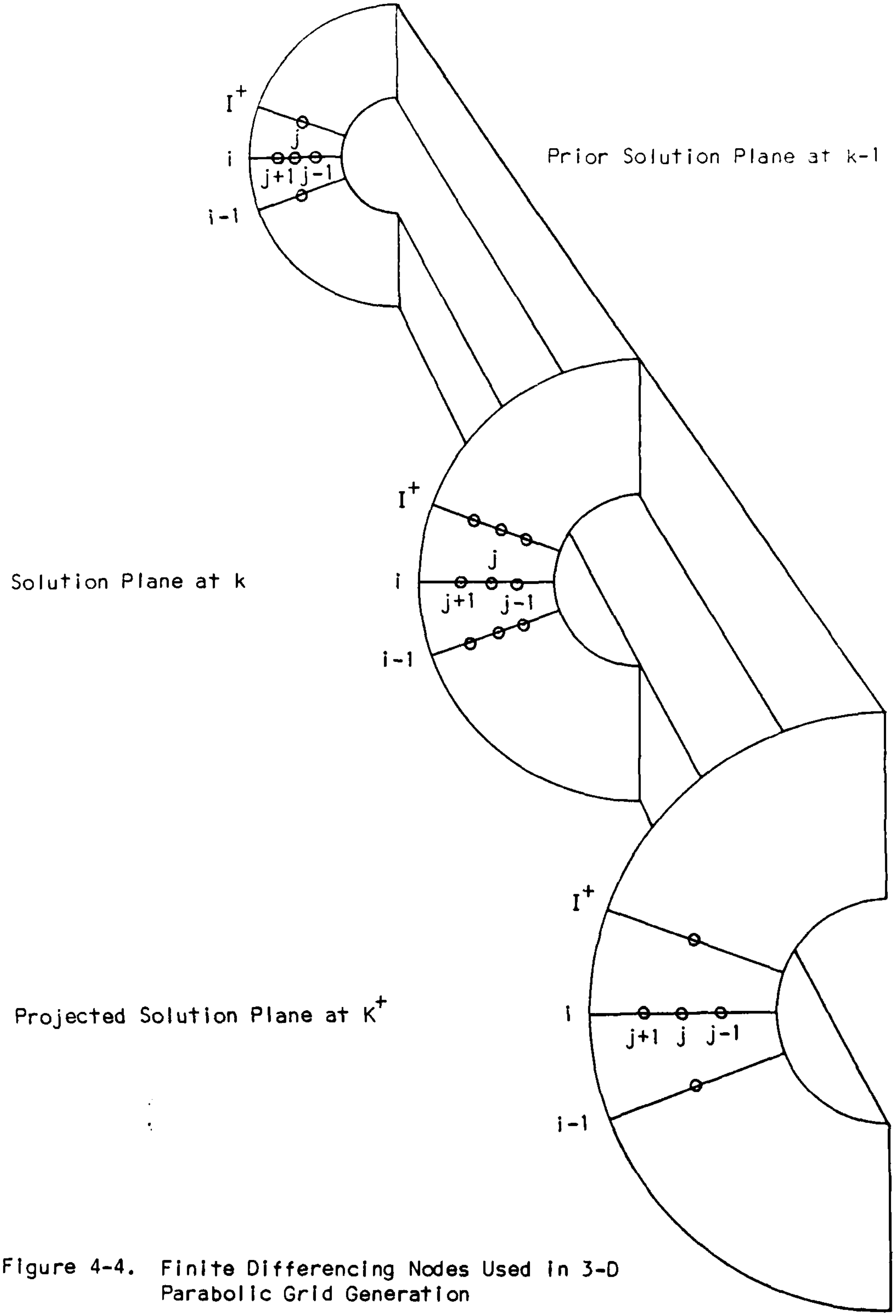


Figure 4-4. Finite Differencing Nodes Used in 3-D Parabolic Grid Generation

4.2.3 Orthogonal Projected Solutions

The most complicated part of the 3-D grid generation procedure is filling the solution and projected solution 2-D arrays with the nearly orthogonal curves. The procedure is similar to the 2-D wrapped grid procedure stated in Section 4.1.6. The solution and projected solution 2-D arrays are first filled with straight lines with the proper Q distribution of points. This is done by scaling the stored 1-D Q distribution of points between the inner and outer boundary points in the solution and projected solution planes.

Calculating orthogonality for the 3-D nearly orthogonal curve is analogous to the 2-D procedure. An additional set of angles in the x-z plane is required for three dimensions. These angles are shown in Fig. (4-5). The angle of the line between the boundaries and the angle of the normal line to the surface in the x-y plane are calculated the same way as in Section 4.1.6. Since the inner and outer boundary points of a cross section have the same z coordinate, the angle of the line between the boundaries in the x-z plane is zero. The angle of the normal line to the surface in the x-z plane is calculated by taking the arctangent of the negative inverse of the slope of the upstream panel. The geometry is invariant along a generated ray in the x-z plane, so it is unnecessary to average the arctangents between the panels on each side of the node point in the x-z plane. Since the waverider is wedge shaped in the x-z plane, the quadrant of the lines normal to the inner boundary are already specified. If this code is used on another geometry that is highly-curved in the x-z plane, angles in the x-z plane have to be calculated like those in the x-y plane.

The rest of the 3-D nearly orthogonal curve generation procedure is similar to the 2-D wrapped grid procedure. The factor that will retard or accelerate the hyperbolic tangent is calculated by Eq. (4-4) for the x-y plane and the factor for

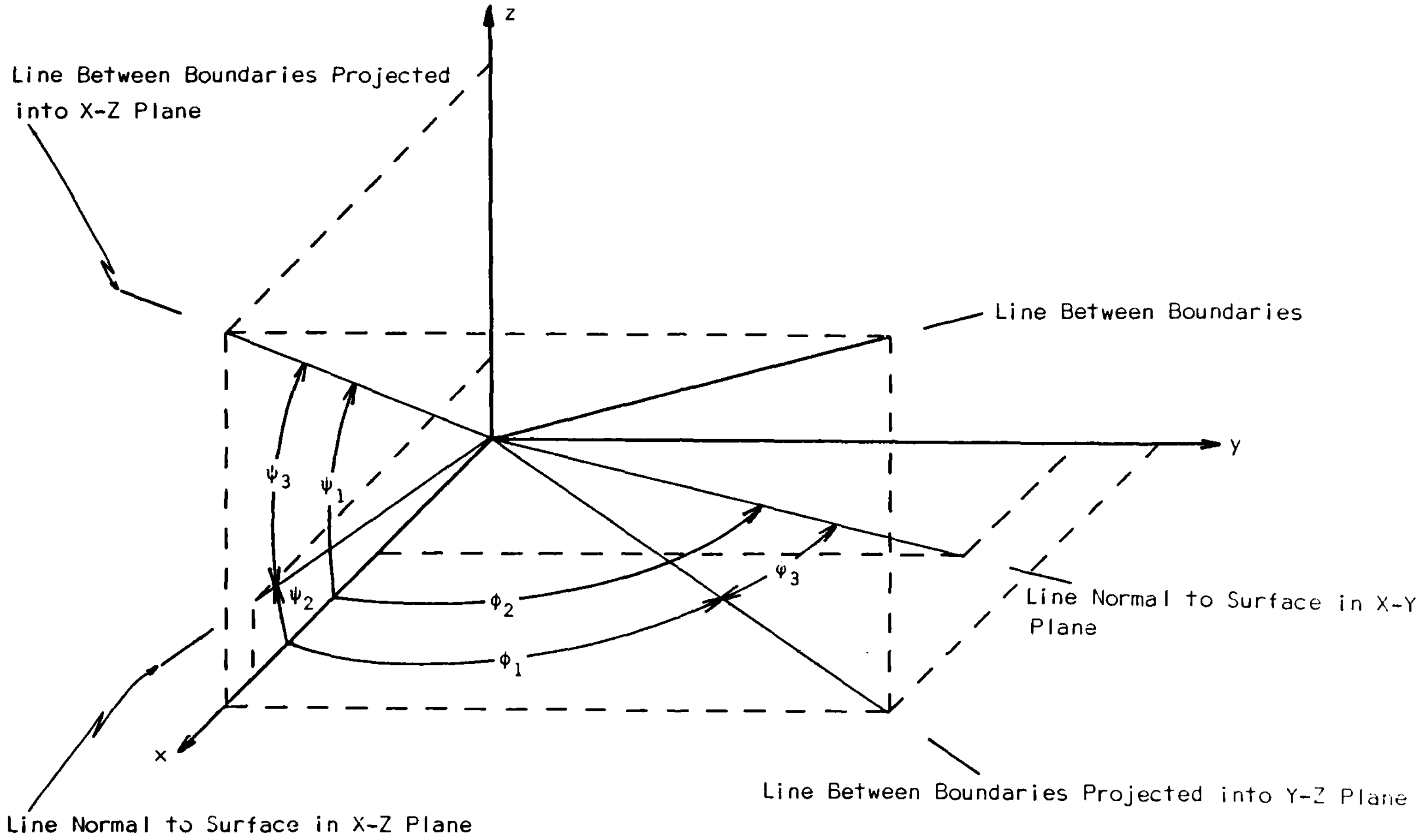

z
Line Between Boundaries Projected into X-Z Plane
Line Between Boundaries
y
ψ_3
ψ_1
ψ_2
ϕ_2
ϕ_1
ψ_3
Line Normal to Surface in X-Y Plane
x
Line Between Boundaries Projected into Y-Z Plane
Line Normal to Surface in X-Z Plane

Figure 4-5. Angles Used in Calculating the 3-D Nearly Orthogonal Curves

the x-z plane is given by

$$\text{speed factor 2} = \frac{1}{|\psi_2 - \psi_1|} \text{ in radians} \qquad (4-9)$$

The distance between the inner and outer boundary and the centering position are calculated in three dimensions. The automatic grid crossover check is done the same way as the 2-D wrapped grid procedure. This procedure was discussed in Section 4.1.7. Since there are no concave regions in the x-z plane, the automatic grid crossover check is only utilized in the x-y plane. Again, if there are concave regions in the x-z plane, the x-y plane procedure can be used in the x-z plane. The angle that controls the orthogonal curve is generated in the x-z and x-y planes the same way as the 2-D wrapped grid procedure. The angle in the x-y plane is given by Eq. (4-5), the angle in the x-z plane uses the same equation, but with the ϕ's replaced with ψ's. The x, y and z coordinates of the nearly orthogonal curve are generated by using ϕ_3, ψ_3 and the distance from the inner boundary to the point of concern on the straight line solution. The x and y coordinates are generated in the same manner as in the 2-D wrapped grid procedure as stated in Section 4.1.6. The z coordinate uses the $\sin\phi_3$ and the distance from the inner boundary. If after the solution marches downstream once in the ς direction and K^+ equals one, the previous projected solution plane is now equal to the present solution plane. The previous solution plane values are rotated into the present solution plane. Only the nearly orthogonal curves that fill the projected solution plane needs to be calculated.

Now that the three 2-D arrays have been filled, the 3-D parabolic grid generation procedure can be started. For each marching step taken in the ς direction, IMAX marching steps are taken in the ξ direction. The 3-D parabolic grid generation procedure is basically doing the 2-D procedure KMAX times, but with different metrics and coefficients. Since the geometry is highly-curved in the x-y plane the same convexity check that was discussed in Section 4.1.5 has to be used to calculate the proper I^- location. The convexity check is not done in the x-z plane, because there are no convex surfaces. Although, there could be problems with placing the projected solution position too far downstream in the ς direction, but these problems would be associated with too much smoothing. In general, there will be a lot of smoothing if the projected solution is far away from the solution station. If another geometry is used, the x-y plane procedure can easily be used in the x-z plane to check for convex surfaces. The metrics and coefficients of Eq. (3-20) are generated with either a first-order backward differencing or a second-order central differencing. Again the metrics and coefficients at the first and last ξ solutions at the plane-of-symmetries are calculated with the second-order central difference. Central differencing can be used if I^- and K^- are equal to one. The coefficients are calculated and the finite difference matrix, Eq. (3-20), is sent to subroutine TRIDIG where the 3-D parabolic equation are solved. This process is done three times, once for each of the three coordinates. Again the program is set up so it can iterate, but unless backward differencing is used on the metrics, iteration is futile. The values in the 2-D solution plane are rotated into the 2-D prior solution plane. The procedure continues until the solution has marched KMAX times in the ς direction.

The following is a summary of the procedure for generating 3-D parabolic grids:

1) Locate nose of waverider, calculate length of waverider, and use Eq. (3-3) to calculate location of cross sections in the ς direction with respect to the grid control term R.

2) Generate and store waverider geometry in base plane, calculate cross section wing tip intersections of the shock, calculate geometry scale factors, and scale base plane geometry to generate inner and outer boundaries.

3) Generate and store solution to 1-D grid generation equation, Eq. (3-3), with Q

distribution and JMAX points.

4) Load waverider nose point into 2-D prior solution plane array.
5) Generate nearly orthogonal curves in the 2-D solution plane array and the 2-D projected solution plane array.
6) March one step in the ξ direction.
7) Check for grid crossover and a good I^{-} location in the x-y plane.
8) Calculate metrics and coefficients with a first-order backward difference or a second-order central difference. Second-order central differences are used at the plane-of-symmetries.
9) The 3-D parabolic grid generation equation given by Eq. (3-20) is solved by a tridiagonal algorithm.
10) Go to step six until the solution marches IMAX times in the ξ direction.
11) Rotate present solution plane in prior solution plane and take another marching step in the ζ direction by going to step six. If finished marching KMAX times in the ζ direction. the 3-D grid generation procedure is finished.

V. Results

5.1 Two-Dimensional Wrapped Grid Problems and Results

5.1.1 Boundary Generation Problems

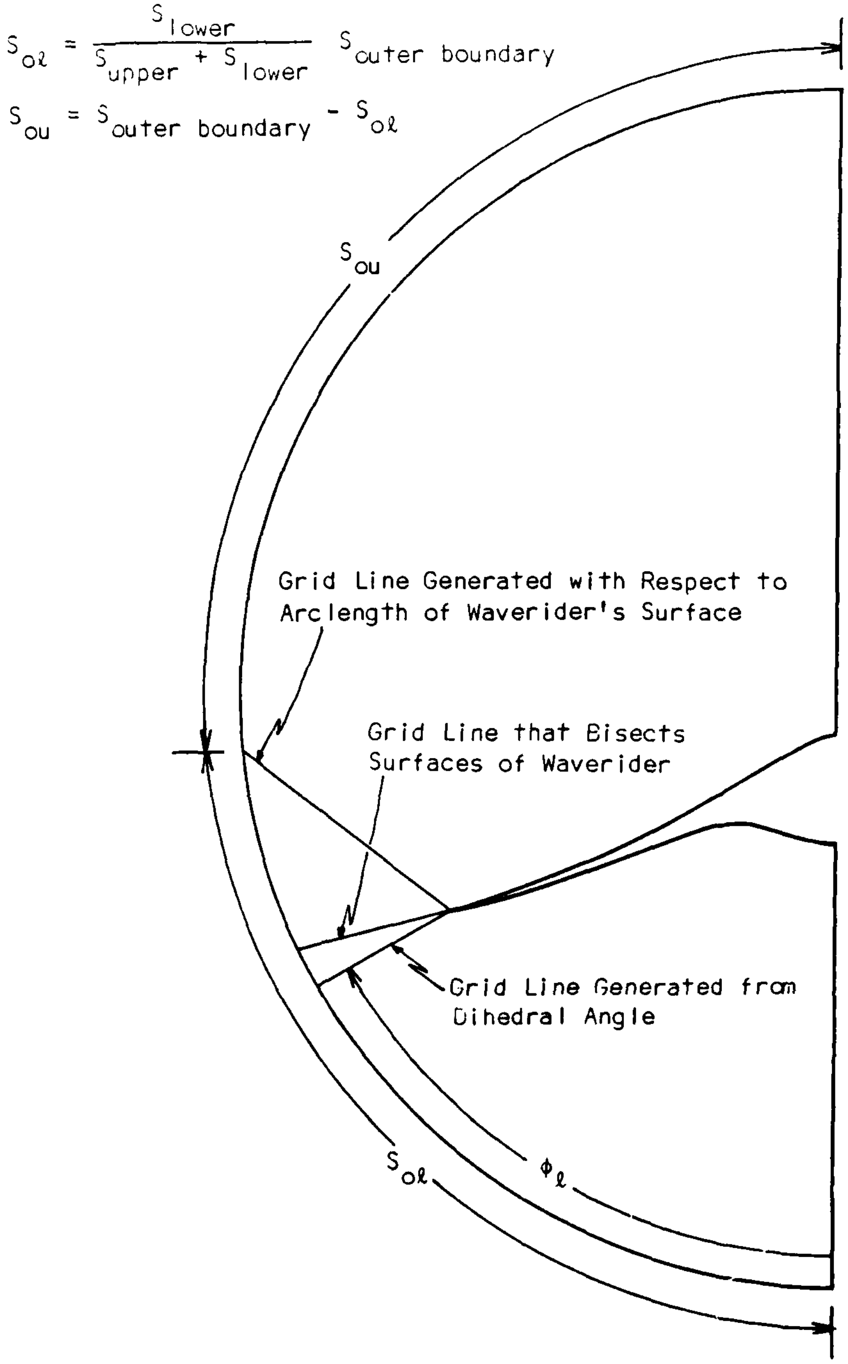
The first problem encountered was the location of the origin of the coordinate system with respect to the waverider. The waverider geometry was generated in Eqs. (2-1) and (2-2) using polar coordinates. These equations construct the waverider in such a way, that the origin does not reside within it's boundaries. The origin will sit on the upper surface of the waverider at the plane-of-symmetry if the variable δ_1 is equal to one, a conical waverider. There is still a problem with the origin, even if it lies on the waverider surface. The outer boundary varies from 0 to 180 degrees. The waverider varies from 0 degrees to the dihedral angle and back to 0 degrees. This will cause a big problem with generating grid lines on the upper surface. For example, at the plane-of-symmetry on the upper surface, the outer boundary has a value of 180 degrees, whereas the waverider symmetry point has a value of 0 degrees. It is impossible for the 2-D parabolic procedure to vary ϕ and still get the correct distribution of points on the straight plane-of-symmetry line. The parabolic procedure will vary ϕ and produce a wildly curved grid line.

There are two immediate solutions to this problem. The first one would be to move the origin between the upper and lower surfaces of the waverider on the plane-of-symmetry. This would require that the inner and outer boundary points be modified to reflect the change of the origin. The other solution is to convert the polar boundary points into Cartesian coordinates. The polar type grid distribution is still achieved, because the boundary points have already been calculated. There are no problems with the way x and y vary as did r and ϕ.

The second problem encountered was how to distribute the points on the waverider surface. In order to properly distribute points on the curved boundaries, the arclength must be specified in Eq. (3-3). Since the waverider is highly-curved, points distributed from one of the axes would lead to an erratic distribution of points along the waverider's surface. If the points are distributed over the curved surface according to arclength, a smooth distribution of points will result.

The number of points to be placed on the upper and lower surfaces where first determined by their arclengths. The lower surface, for example, would take its arclength and divide it by the total waverider arclength to generate a fraction. This fraction times the number of points (IMAX), would be the number of points to be placed on the lower surface. This strategy did not work because the arclengths of the lower and upper surfaces are generally about equal, no matter what dihedral angle was used to generate the geometry. The maximum principle was violated, because a straight line can not be drawn between the boundaries at all positions without crossing over the geometry's surface. The wing tip grid line for this situation is shown in Fig. (5-1), and it is plain to see that grid lines generated near the wing tip on the lower surface will have to cross the geometry to get to the outer boundary.

The second method of placing points on the lower and upper surfaces of the waverider was to use the dihedral angle. The number of points to be placed on the lower surface, for example, was determined by dividing 180 degrees into the dihedral angle to generate a fraction. Again, this fraction times the number of points (IMAX), would be the number of points placed on the lower surface. This strategy gave better results than the arclength scheme, but violated the maximum principle. The waverider is constructed in such a way that the wing tip is cusped at an angle greater than the dihedral angle. Therefore, this procedure also caused grid lines to cross the configuration's surface, because not all of the inner boundary points can see their corresponding outer boundary point without crossing the waverider's



$$S_{o\ell} = \frac{S_{lower}}{S_{upper} + S_{lower}} \; S_{outer\ boundary}$$

$$S_{ou} = S_{outer\ boundary} - S_{o\ell}$$

S_{ou}
Grid Line Generated with Respect to Arclength of Waverider's Surface
Grid Line that Bisects Surfaces of Waverider
Grid Line Generated from Dihedral Angle
$S_{o\ell}$
ϕ_ℓ

Figure 5-1. Three Wing Tip Grid Lines Generated by Distributing Points Differently on the Waverider's Surface

surface. This scheme is also shown in Fig. (5-1).

As a rule of thumb for generating boundary points. a straight line has to be drawn from the inner boundary to the outer boundary without crossing the configuration's surface. Because the wing of the waverider is cusped, a line has to be constructed that locally bisects the upper and lower surfaces near the wing tip. The intersection of this line with the outer boundary is determined. The number of points that reside between the start of the outer boundary ellipse and the intersection point is the number of points placed on the lower surface. The rest of the points plus one are the number of points placed on the upper surface. This scheme works extremely well, because all inner and outer boundary points can be connected without intersecting the geometry. This scheme is shown in Fig. (5-1).

Because of the dihedral angle, the lower surface usually contains less points than the upper surface for a constant spaced outer boundary. Since the shock wave is attached to the lower surface, there should be enough points to discriminate the higher pressures. If a negative P distribution of points is used on the outer boundary ellipse, more points can be pulled down onto the lower surface. The outer boundary no longer has an equal distribution of points with respect to arc, but the desired number of points on the lower surface can be achieved without much additional clustering on the outer boundary.

5.1.2 Wing Tip Grid Wrap Around Problem

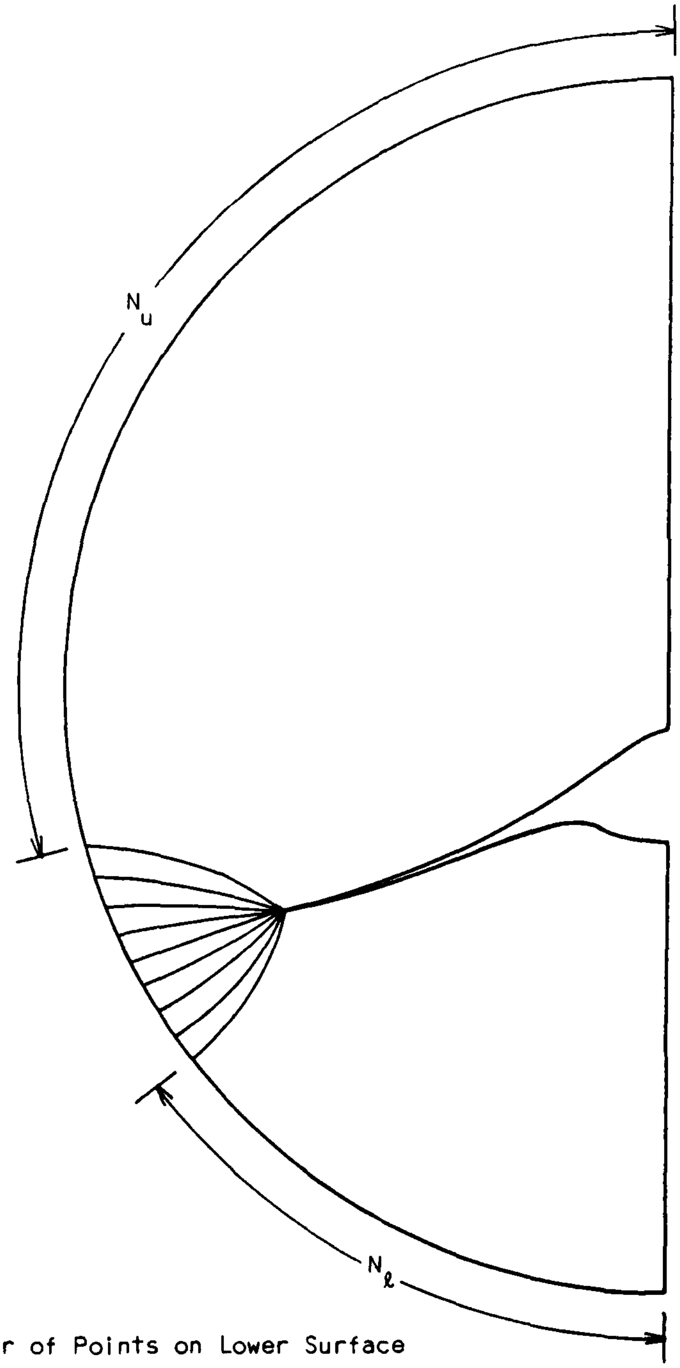
The third problem encountered was how to make the 2-D parabolic grid generation procedure march around the wing tip. Since the wing tip is thin, the parabolic grid procedure has to march around an almost 180 degree turn. The 2-D grid generated around this corner has one or more grid lines that intersect the waverider's surface or each other. Even though a straight line can be drawn from all inner boundary points to their respective outer boundary points without crossing the surface, the maximum principle is still violated.

One solution to this problem is to refine the points near the wing tip. This is shown in Fig. (5-2). The added points at the wing tip fan out the grid lines, so the grid lines in the area of the wing tip do not lie close to the inner boundary. This process eases the 2-D grid generation procedure around the wing tip. The other solution to this wrap around problem is to use a slit transformation, which is discussed in Appendix A.

5.1.3 Projected Solution Convexity Problem

The fourth problem encountered was the projected solution being too far downstream around a convex surface. A test case was run, where a grid was generated between two half circles. The projected solution was set at the far boundary. Q was set at zero so the grid would be constant in the η direction. The 2-D solution resulted in grid points crossing the inner boundary and the first plane-of-symmetry, as shown in Fig. (5-3).

The 2-D parabolic grid generation procedure needs to check and see if the projected solution is too far downstream over a convex surface. This is accomplished by drawing a line through the j=2 nodes of the prior solution and the projected solution. Lines are constructed from the boundary points between the present and projected solutions. The intersections of the line between the two nodes and the lines between the boundaries are calculated. The intersections are checked to see if they lie between the boundaries. If all the intersections are between the boundaries, the projected solution position is acceptable. If any of the intersections are below the inner boundary, the projected solution has to be retarded by one position. The check has to be repeated until all intersections are between the boundaries. Figure (5-4) shows the projected solution set at the far plane-of-symmetry where the convexity criterion has moved it to an acceptable position. Maximum smoothing occurs when



N_ℓ = Number of Points on Lower Surface

N_u = Number of Points on Upper Surface

Figure 5-2. Grid Lines Emanating from Wing Tip

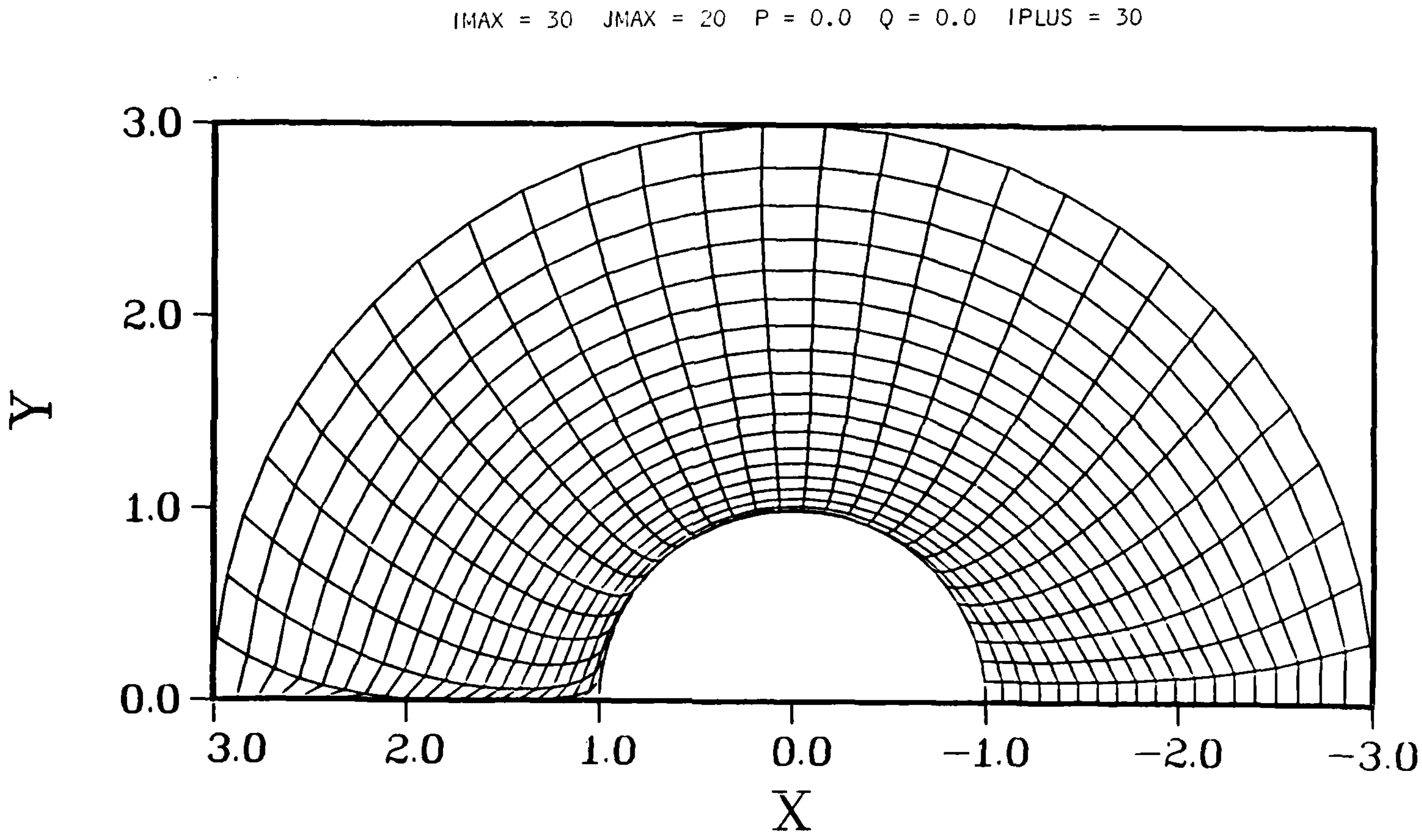


Figure 5-3. Projected Solution at Maximum Position Causing Grid to Cross Boundaries

the projected solution is as far away from the solution station as possible. Figure (5-4) shows the result of the projected solution being as far away from the solution station as the convexity criterion will allow. Figure (5-5) shows minimum smoothing where the projected solution is one station downstream of the solution station. If the grid is highly-clustered near the inner boundary, this procedure will not allow the projected solution to be too far downstream of the solution position, because the second grid points are barely off the geometry's surface. The projected solution is limited to be at least one station downstream of the solution station.

5.1.4 Grid Crossover Problem

The last problem encountered with the 2-D wrapped grid procedure was grid crossover in concave regions when orthogonality has been specified. This problem results because the hyperbolic tangent in Eq. (4-5) is centered too far away from the inner boundary. This caused the orthogonal projected solution to stay orthogonal too far from the inner boundary. The result is grid crossover, which violates the maximum principle. The solution was to move the centering position closer to the inner boundary.

5.1.5 Results

The first 2-D wrapped grid generated, after all the problems stated above were solved, was non-orthogonal to the waverider surface. This grid is shown in Fig. (5-6). There are no problems with grid crossover in this case, because the grid lines basically proceed straight from the inner to the outer boundary. Eight refined points were placed near the wing tip to fan out the grid lines in the wing tip area. This helps the 2-D parabolic grid generator march around the wing tip corner. Fewer refined points can be used near the wing tip, but would make the grid lines lie closer to the waverider's surface in the area of the wing tip. More refined points

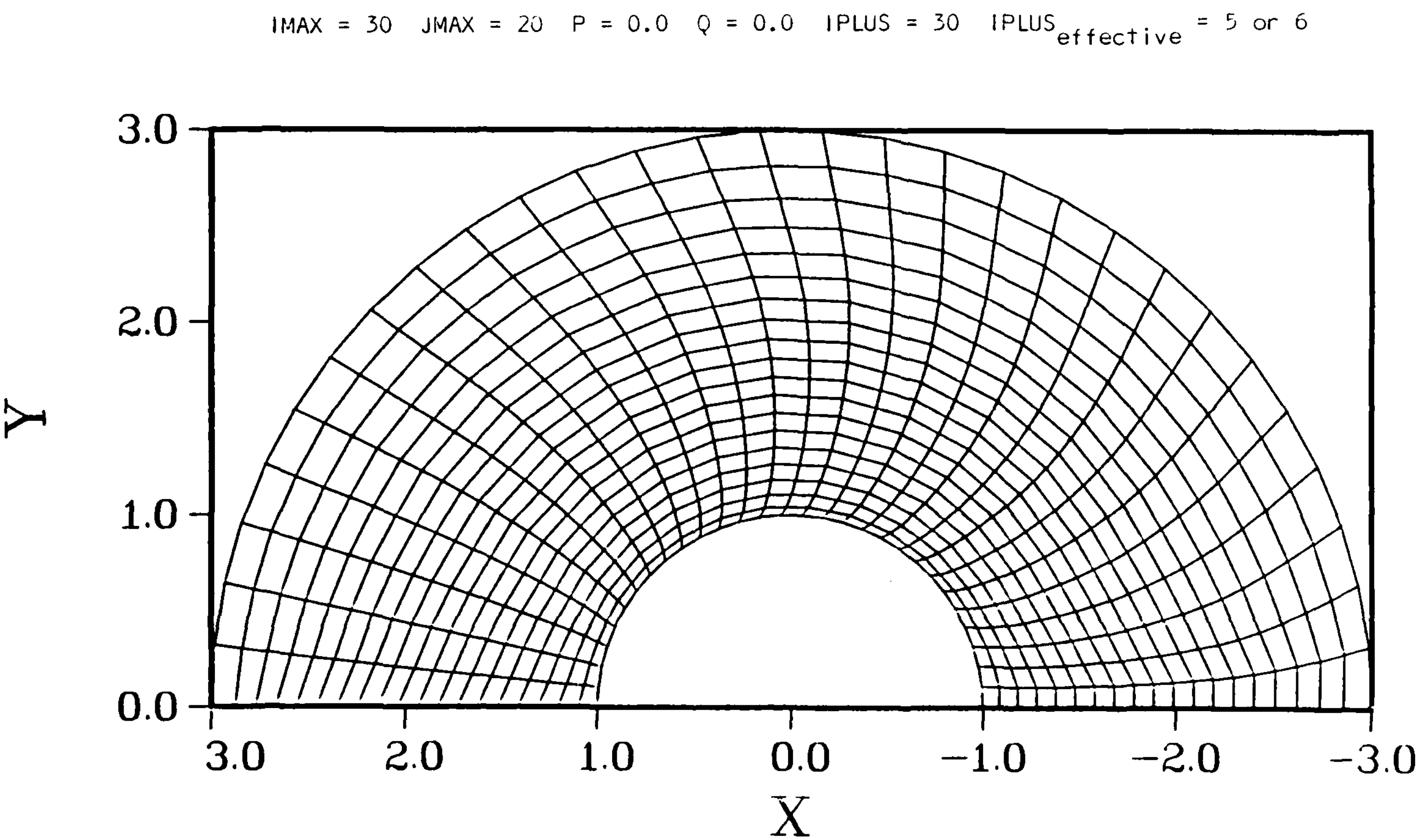


Figure 5-4. Convexity Criterion Backs off Maximum Projected Solution Position to an Acceptable Position with Maximum Smoothing

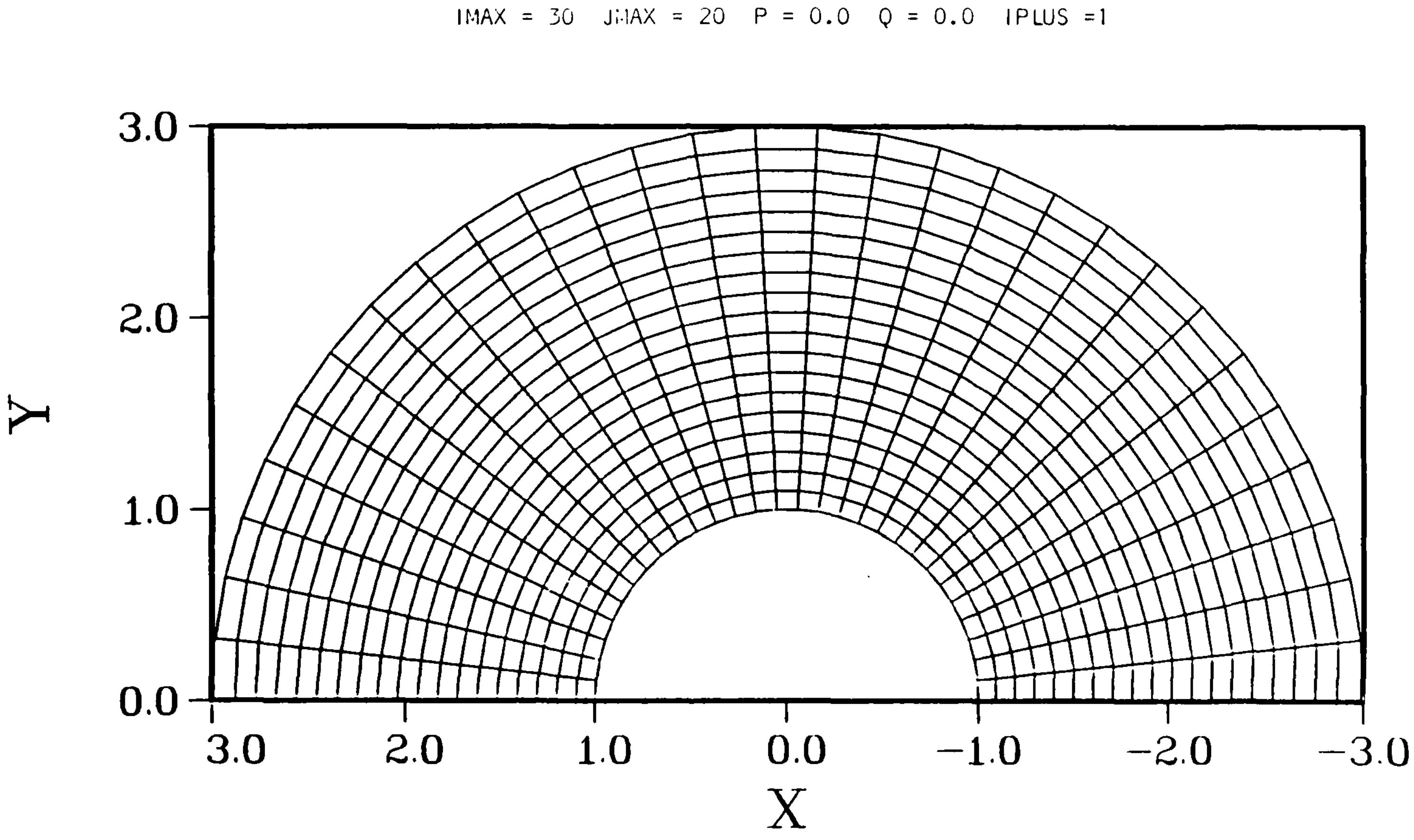


Figure 5-5. Minimum Projected Solution Position Gives Minimum Smoothing

can be used at the wing tip, but since there is a one-to-one correspondence between inner and outer boundary points, less points would actually be put on the rest of the waverider's surface. In order to get the best definition of the waverider's surface, as few as possible refined points should be placed near the wing tip. This grid has 60 points in the ξ direction and 20 points in the η direction. More points are placed on the lower surface, because the outer boundary points were generated with a P distribution of -.02. The negative P distribution of points on the outer boundary clusters points near the lower plane-of-symmetry. This P distribution results in a close to equal spacing of points on the waverider surface. If P is made more negative, more points will accumulate on the lower surface. If P is made more positive, more points will accumulate on the upper surface. The points are also clustered near the inner surface with a Q distribution of -.15. By decreasing Q further, points can be highly-clustered near the surface. The projected solution in this case was one station downstream of the solution station. This grid used central differencing on the metrics, but this approximately generates the same grid as the backward differencing would generate. The 2-D parabolic grid generated an acceptable non-orthogonal grid.

The next grids that were generated were orthogonal to the inner boundary and slowly curved towards the outer boundary. Many grids were generated without grid crossover problems when a coarse grid in the ξ direction was used. When the grid was densified, grid crossover occured in concave regions. This is shown in Fig. (5-7). The same parameters are used to generate this grid as were used to generate the non-orthogonal grid in Fig. (5-6). The Q distribution was set to zero, which generated a constant grid in the η direction. The hyperbolic tangent was centered on the tenth point, which is centered between the inner and outer boundaries. At this centering point the orthogonal curve bisects the normal line and the line that

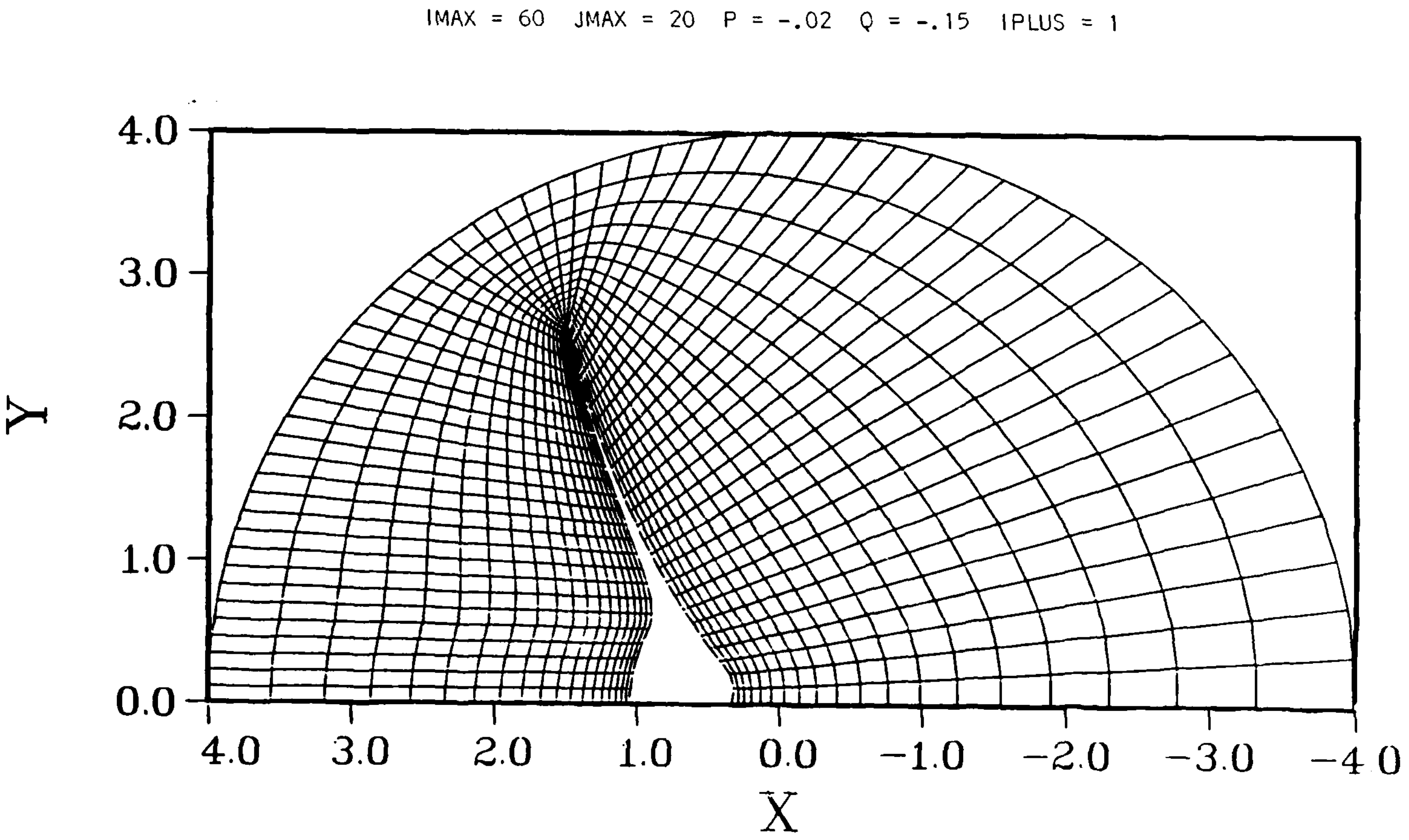


Figure 5-6. Non-Orthogonal Waverider Grid (2-D Parabolic Grid)

connects the inner and outer boundaries. This resulted in grid crossover in the concave region, because the nearly orthogonal centering is locally positioned too close to the outer boundary. The grid lines everywhere else are okay.

The first way of solving this problem was to simply move the point where the hyperbolic tangent was centered. The result of this variation is shown in Fig. (5-8). The same parameters are used in this grid as used in the grid in Fig. (5-7), except the centering point was moved from the tenth point to the sixth point. This solves the problem with the grid crossover in the concave region, but also decreased the orthogonality everywhere else.

The second way of solving the grid crossover problem is similiar to the first. Instead of moving the centering point, the points are clustered closer to the inner boundary. This grid is shown in Fig. (5-9). Even though the hyperbolic tangent is still centered on the tenth point, the tenth point is pulled closer to the waverider's surface with the user specified Q distribution.

The third way of solving the grid crossover problem, is to contol the centering position locally in a concave region. This grid is shown in Fig. (5-10). Again, the grid is generated with the same parameters that caused the grid crossover in Fig. (5-7), except the program locally controls the centering position.

The grids used in Figs. (5-7) through (5-10) have 1200 grid points. The total CPU time to generate these grids on a Cyber 170-845 is .24 seconds. This translates into .2 milliseconds per grid point.

5.2 Three-Dimensional Wrapped Grid Problems and Results

The only significant problem encountered in generating the 3-D grid was in generating the boundary points. This was not a trivial task. The inner and outer boundaries had to be fully visualized before a grid could be generated. If the waverider is conical, the nose is at the apex of the shock cone. Therefore, it is

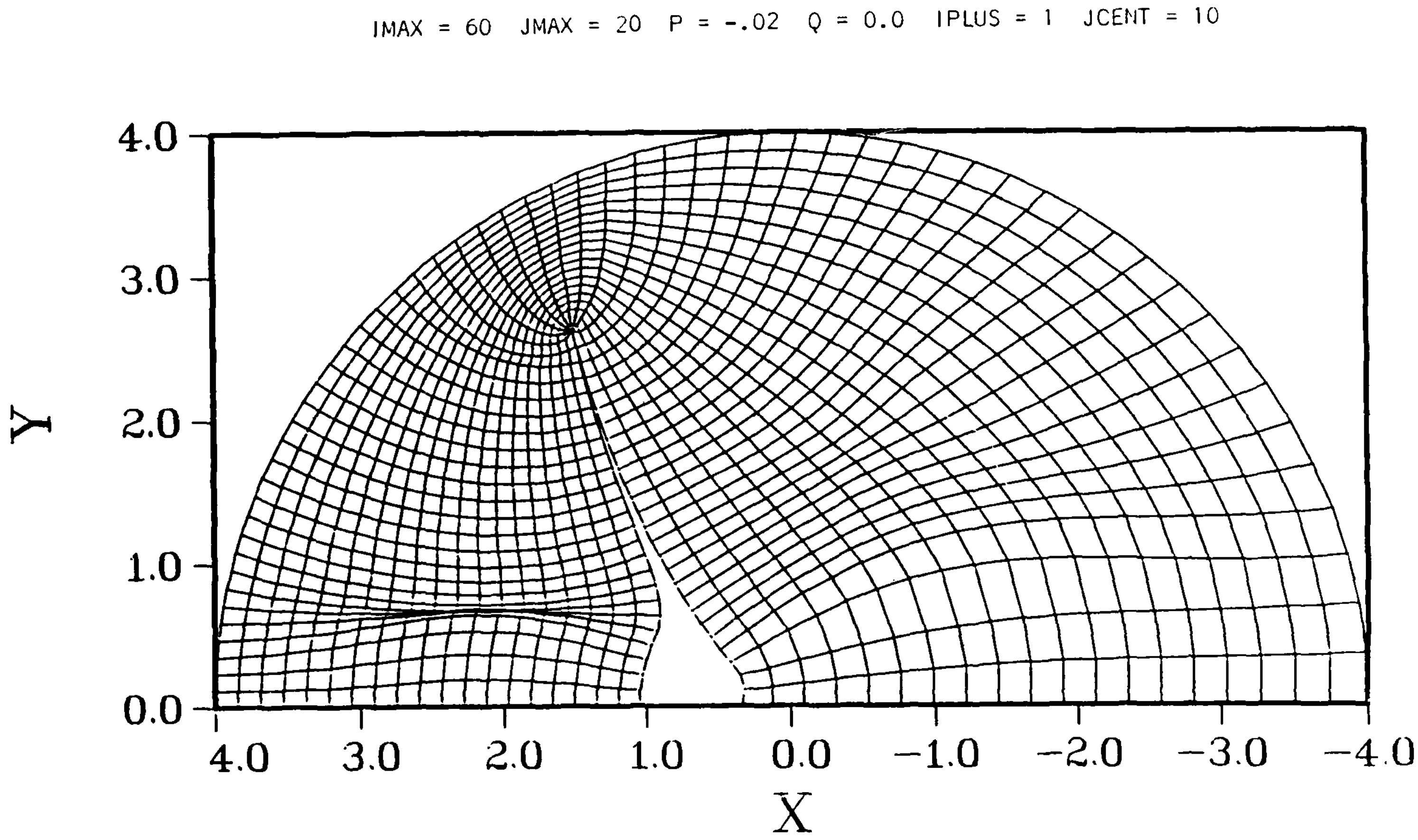


Figure 5-7. Nearly Orthogonal Waverider Grid with Grid Crossover (2-D Parabolic Grid)

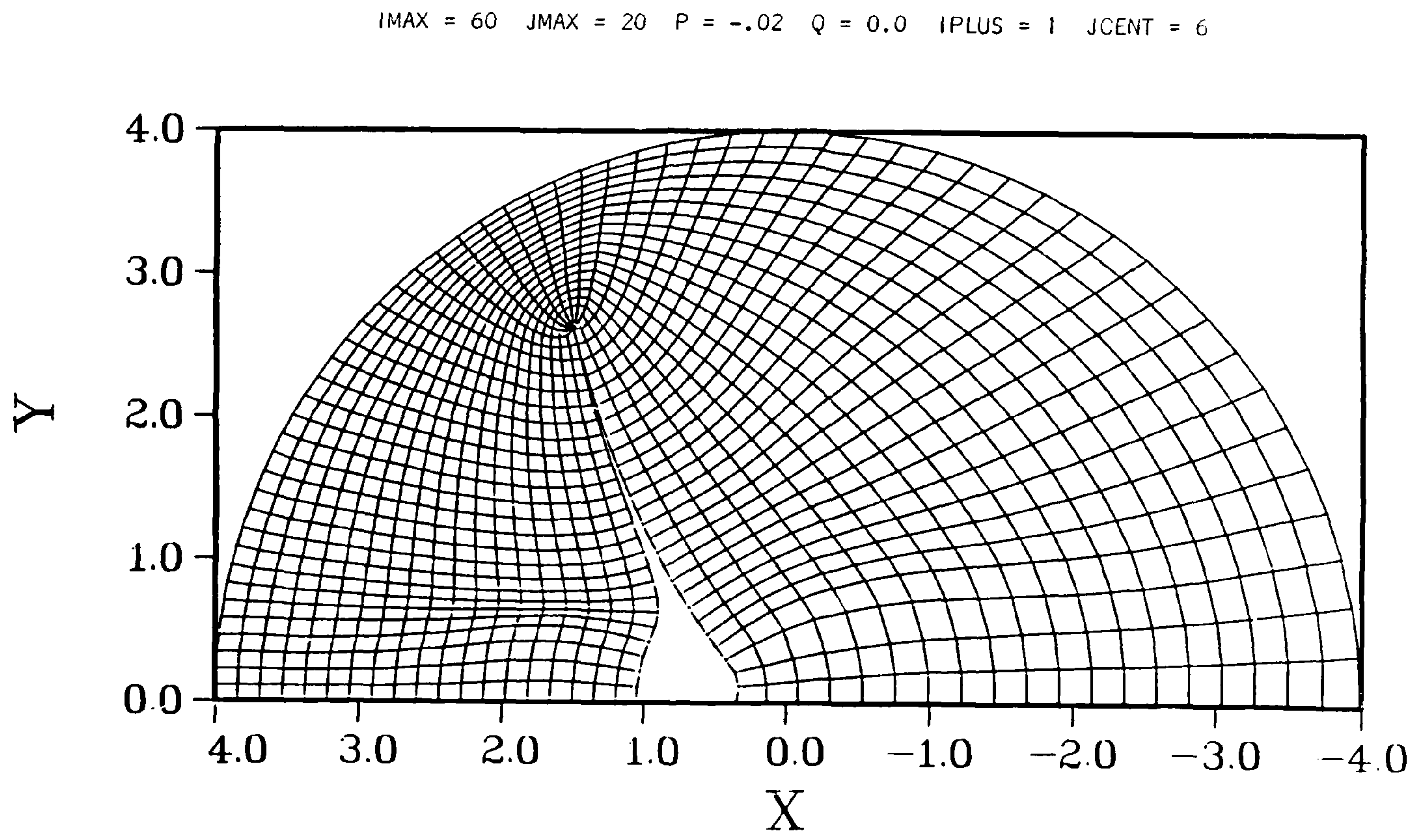


Figure 5-8. Nearly Orthogonal Waverider Grid with Grid Crossover Controlled by Re-Calculating Hyperbolic Tangent (2-D Parabolic Grid)

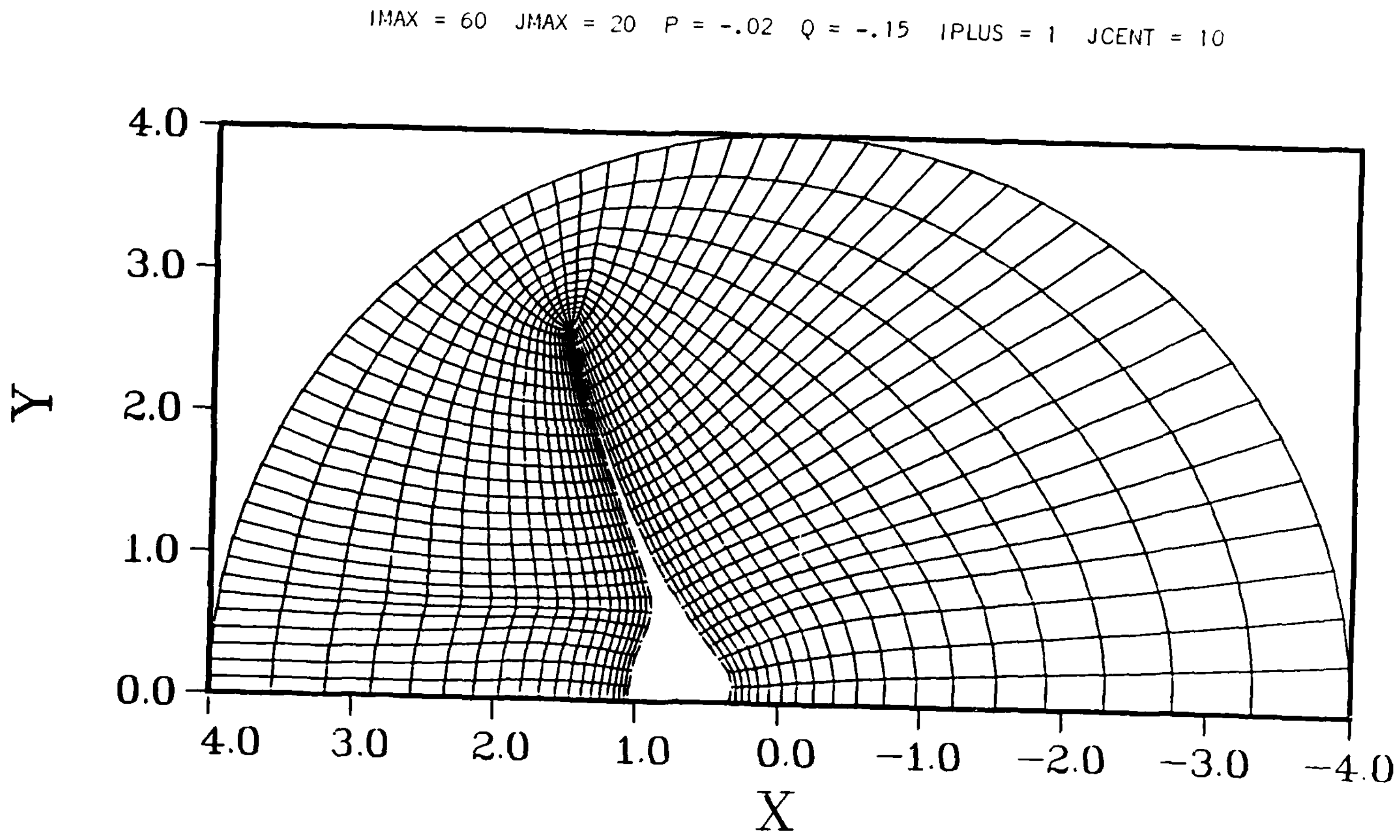


Figure 5-9. Nearly Orthogonal Waverider Grid with Grid Crossover Controlled by Clustering Points (2-D Parabolic Grid)

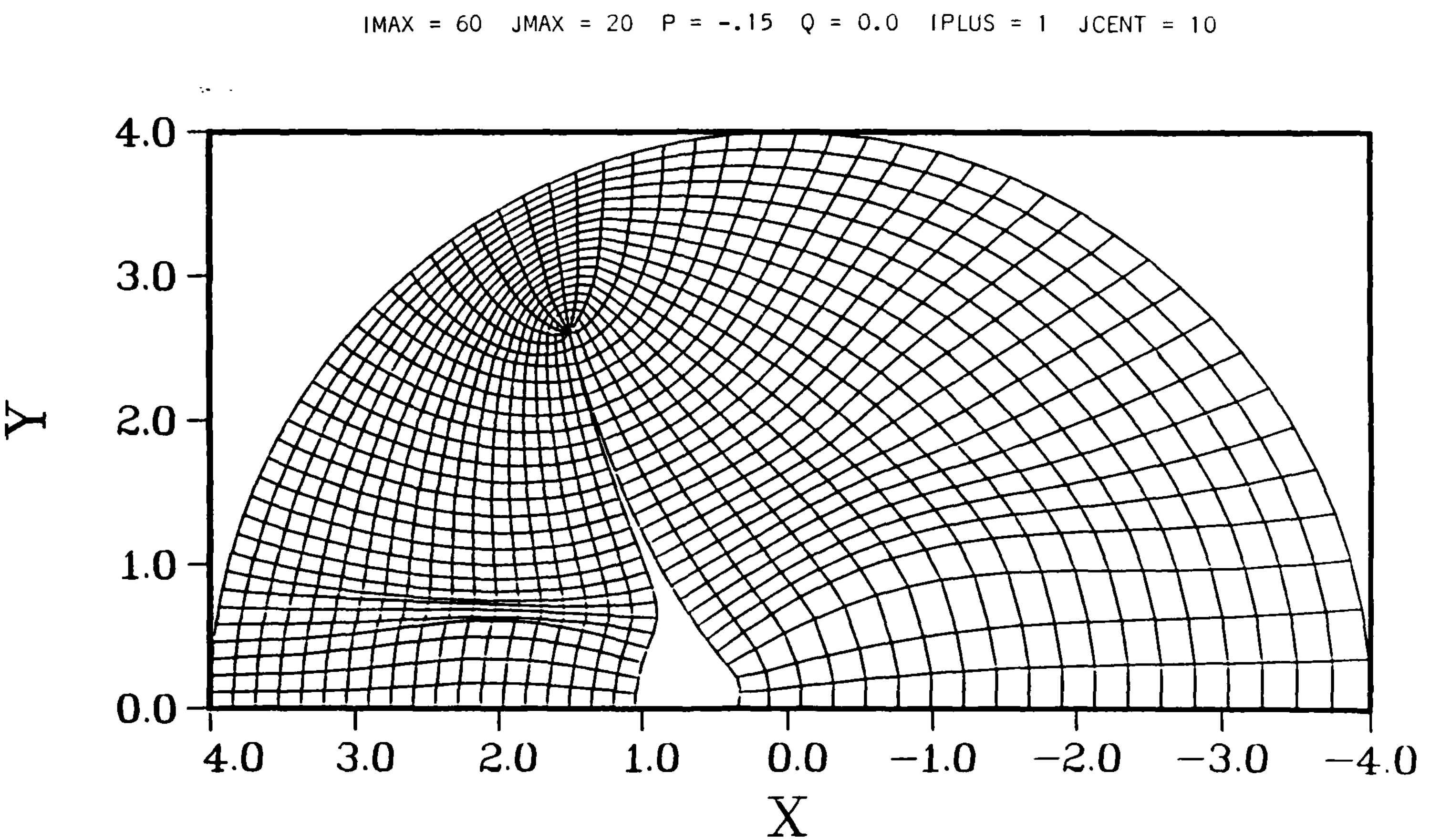


Figure 5-10. Nearly Orthogonal Waverider Grid with Automatic Grid Crossover Check (2-D Parabolic Grid)

resonable to start the outer boundary there also. Since the waverider is wedged shaped in the x-z and y-z planes, the outer boundary will also be conical.

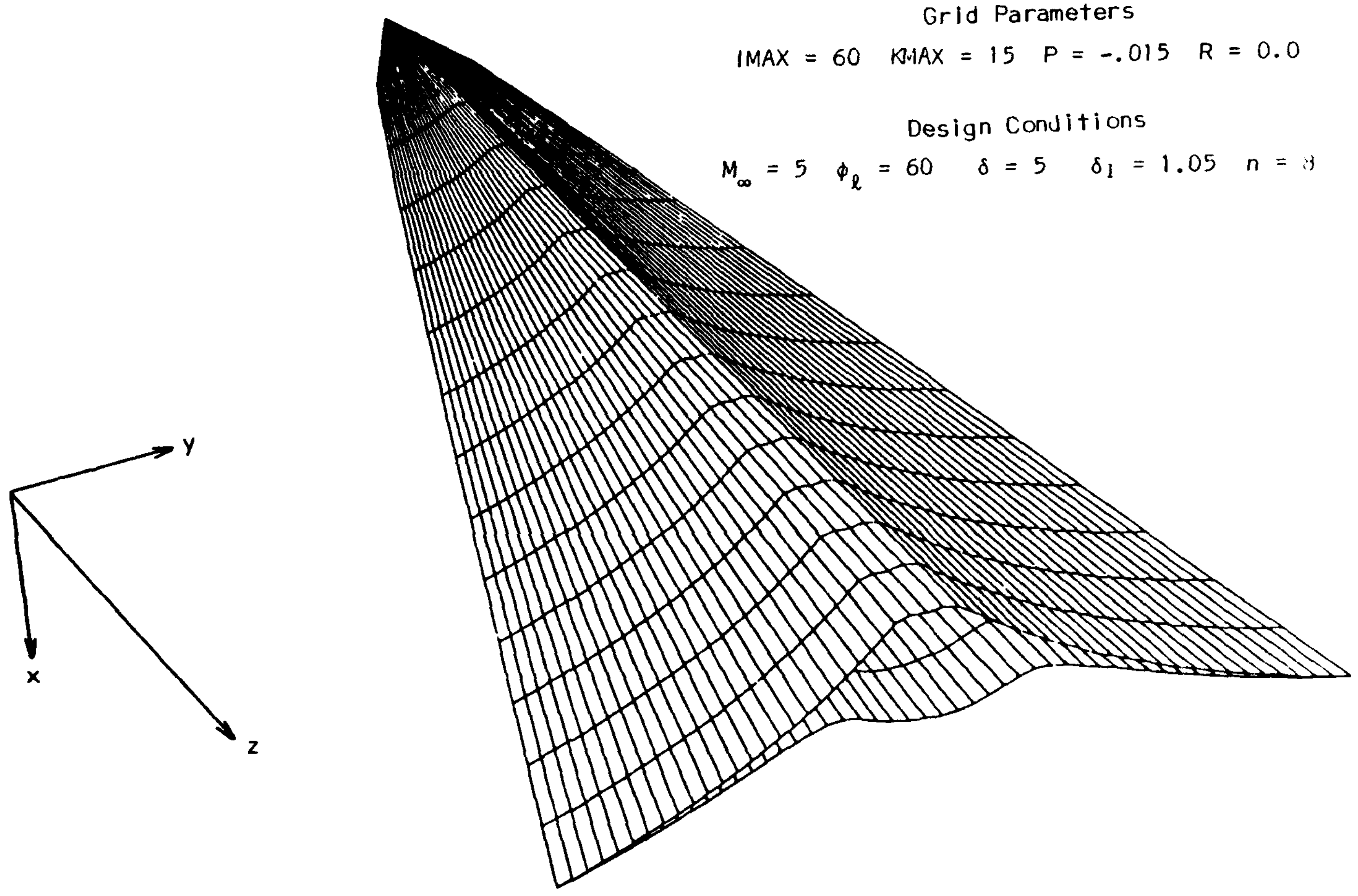
The problem arises for a non-conical waverider where the nose is not on the apex of the shock cone, but at some other location on the shock in the plane-of-symmetry. The outer boundary at the nose is elliptical and extends outside the actual shock wave. The nose of the waverider will sit near the lower side of the outer boundary ellipse in the plane-of-symmetry. This point is at a distance away from the lower side of the outer boundary ellipse, according to how far the outer boundary is away from the shock. All the grid points that describe the waverider's surface and all the grid lines that go from the lower surface of the waverider to the shock wave are described by that one point. The grid lines originating from the lower surface will have most of it's points at the single nose point and then stretch a few points out to the outer boundary. This will generate a discountinuity at the beginning of the grid. Therefore, the outer boundary must start at the nose of the waverider with the outer boundary translated with respect to the x coordinate of the nose.

Since the outer boundary of a non-conical waverider is itself non-conical, it is important to discuss shock fitting versus shock capturing. Shock fitting calculates the outer boundary in a space marching scheme as it is needed. The outer boundary in this case would be at the ideal inviscid shock location. This causes a problem for the ideal case, because the shock sits directly on the wing tip. Since a grid can not be generated in the η direction if the inner and outer boundary points are at the same point, shock fitting may not be appropriate. Shock capturing can be used in place of shock fitting. Shock capturing is enhanced if the shock is coincident with a grid line. The waverider grid is generated in such a way that a grid line will not be coincident with the shock. The shock can still be captured if a grid line is not coincident with the shock. Shock capturing or shock fitting are appropriate for the parabolic grids, since the outer boundary can be specified. However, grids appropriate for shock capturing are generated.

Figure (5-11) shows the upper surface paneling representation of the waverider. Figure (5-12) shows the lower surface paneling representation of the waverider. The base plane waverider equations in Chapter Two will generate many different shaped waveriders. The design conditions chosen for this waverider were Mach number equal to 5, dihedral angle equal to 60 degrees, original cone semi-vertex angle equal to 5 degrees, lower surface of the waverider in the plane-of-symmetry to be 1.05 from the original cone origin in the base plane where 1.0 is the original cone's surface, and shape factor equal to 8.

The 3-D grid generation procedure generated a nearly surface orthogonal grid. Since the original cone angle was five degrees, there is little variation in the grid lines in the z direction. This is most evident when the inner and outer boundary points of a cross section are at the same z location. Figure (5-13) shows the 3-D grid in the x-z plane-of-symmetry. Since the upper surface is aligned with the freestream, the grid lines on the upper plane-of-symmetry are fully orthogonal to the surface. The lower surface is inclined at an angle to the freestream, therefore the grid lines start orthogonal to the surface and bend back towards the outer boundary point. The grid in the x-y plane is very similar to the 2-D wrapped grid results.

The 3-D wrapped grid is similar to the 2-D wrapped grid, but there is more smoothing. The 3-D parabolic equations use 19 nodes, shown in Fig. (4-5), to generate the coefficients for one node point. The 2-D parabolic equations use 9 nodes to generate the coefficients for one node point. Therefore, one would expect differences in the solutions. Figure (5-14) shows a nearly surface orthogonal 3-D parabolic grid around a non-conical waverider. This picture of the 3-D grid



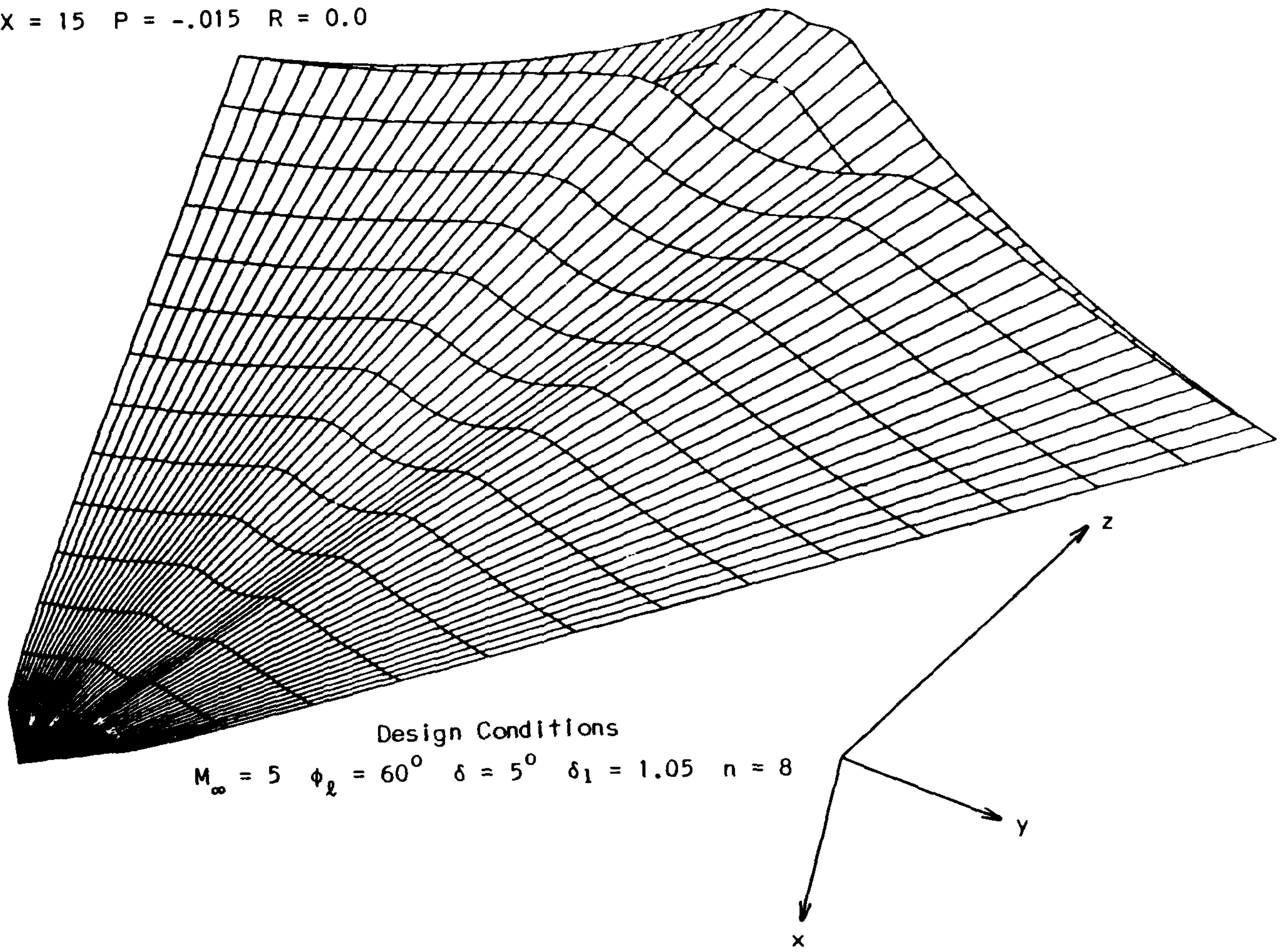
Grid Parameters

IMAX = 60 KMAX = 15 P = -.015 R = 0.0

Design Conditions

$M_\infty = 5$ $\phi_\ell = 60$ $\delta = 5$ $\delta_1 = 1.05$ $n = 8$

Figure 5-11. 3-D Upper Surface Paneling of Waverider Configuration



Grid Parameters

IMAX = 60 KMAX = 15 P = -.015 R = 0.0

Design Conditions

$M_\infty = 5 \quad \phi_\ell = 60^o \quad \delta = 5^o \quad \delta_1 = 1.05 \quad n = 8$

Figure 5-12. 3-D Lower Surface Paneling of Waverider Configuration

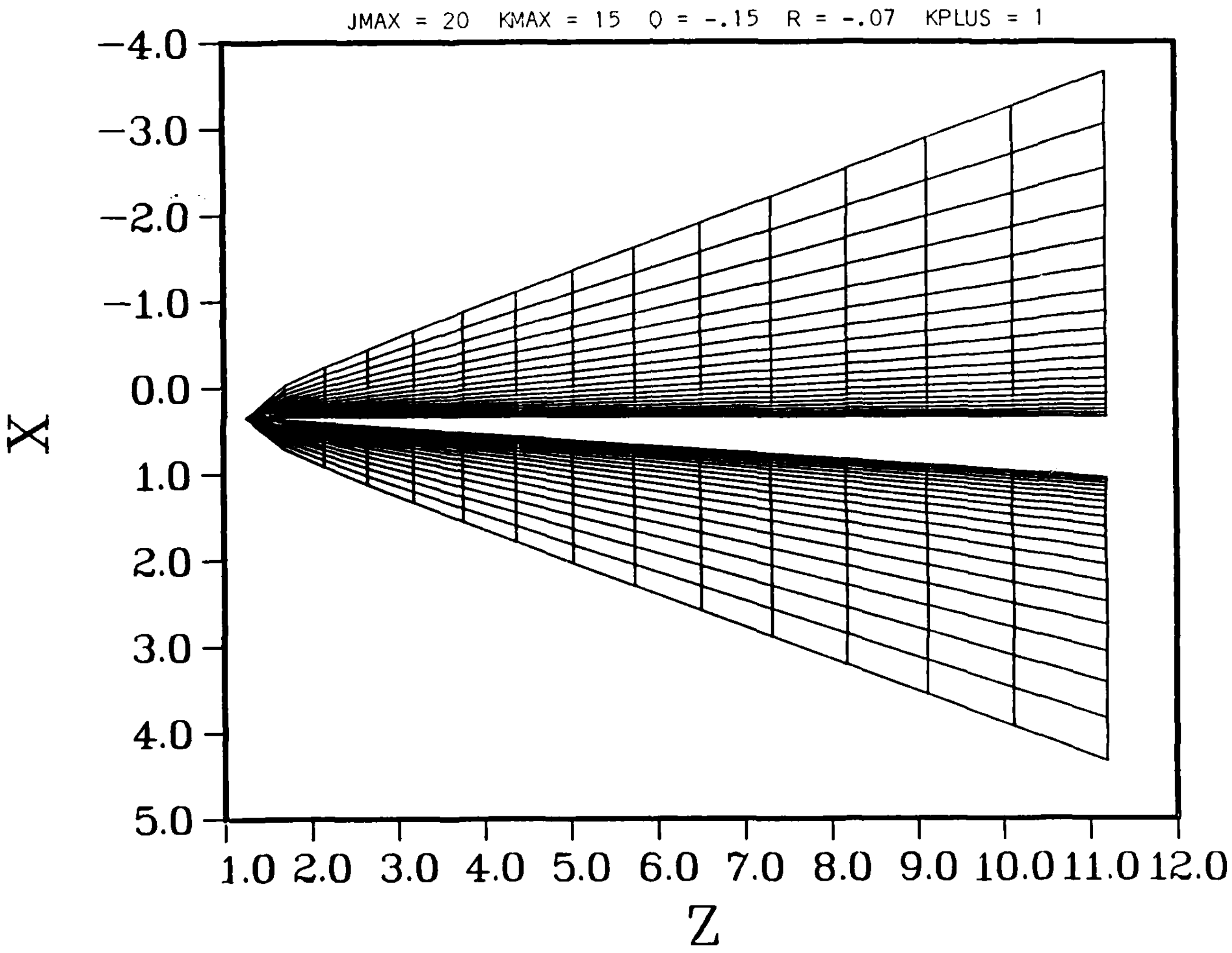
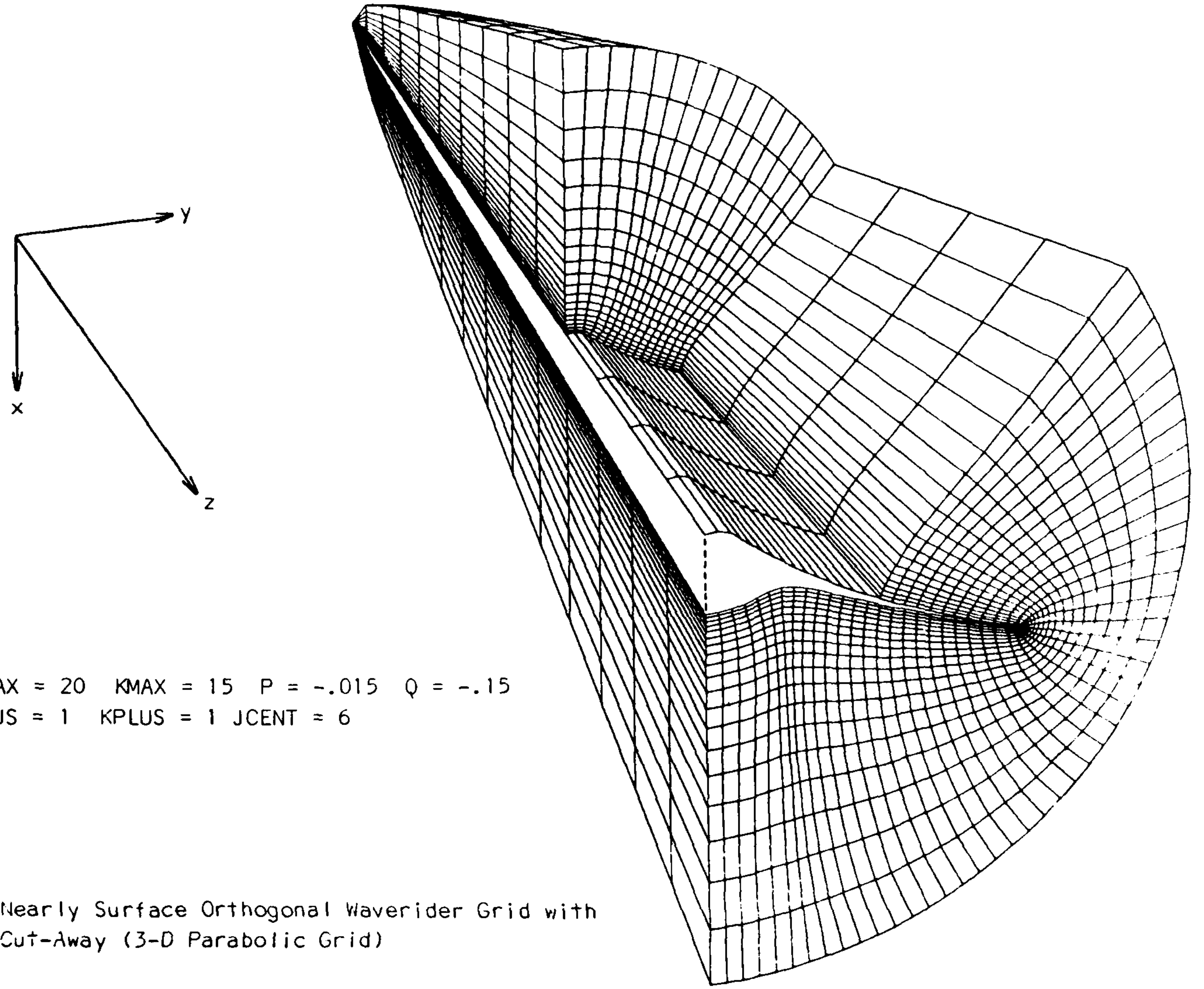


Figure 5-13. Nearly Surface Orthogonal Waverider Grid in Plane-of-Symmetry (3-D Parabolic Grid)

system shows how well this procedure works. The grid is smoothed around the discountinuity of the wing tip.

The grid used in Fig. (5-14) has 18.000 grid points. The total CPU time on a Cyber 170-845 was 7.2 seconds. This translates into .4 milliseconds per grid point. This result is two orders of magnitude faster than elliptic grid generation. To save a great deal of iteration with an elliptic grid generator. this parabolic grid can be put into the elliptic generator as the first guess. One of the best developed hyperbolic grid generation techniques[14] takes .24 milliseconds per grid point on a Cray-XMP-12. Considering that the scalar speed of the Cray is approximately six times faster than the Cyber 170-845 or faster. this parabolic procedure is comparative to the hyperbolic procedure. except for the fact that the parabolic scheme does not have all the limitations of the hyperbolic scheme.



IMAX = 60 JMAX = 20 KMAX = 15 P = -.015 Q = -.15
R = -.07 IPLUS = 1 KPLUS = 1 JCENT = 6

Figure 5-14. Nearly Surface Orthogonal Waverider Grid with Cut-Away (3-D Parabolic Grid)

VI. Conclusion

Parabolic grid generation equations were used to generate 2-D and 3-D surface orthogonal grids around a non-conical waverider configuration in terms of Cartesian coordinates. The grids were generated without iteration. Specified grid control terms allowed the grid to be exponentially stretched in any direction, which allowed the grid to be highly-clustered near any boundary without using grid embedding. The parabolic grid generation procedure generated a smooth grid, even around an almost 180 degree corner of the wing tip on the waverider. The parabolic grid generation procedure is efficient with respect to memory requirements and computer time. The 3-D grid generation procedure requires three tridiagonal solutions per grid line and takes approximately .4 milliseconds per grid point on a Cyber 170-845. The 2-D grid generation procedure requires two tridiagonal solutions per grid line and takes approximately .2 milliseconds per grid point on a Cyber 170-845. The parabolic grid generation procedure coupled with a marching flow solver allows the grid to be generated along with the flow solution. The degree of grid smoothing is controlled by the positioning of the projected solution with respect to the solution station position. Grid crossover is controlled in concave regions when orthogonality is specified.

VII. Recommendations

Further extensions of this research is recommended in several areas. First, the grids generated with the parabolic procedure should be used by a flow solver and compared with the results from grids generated by elliptic or other methods. Second, make this code easily adaptable to other geometries. This can be done by putting the convexity test, and grid crossover test into the third dimension. Instead of using straight lines to approximate solutions, curves or a combination of lines could be used. Third, the program can be extended to use variable grid control terms. Presently, the Q distribution of points is calculated once and stored. Using a variable Q distribution would require IMAX more tridiagonal solutions in 2-D and IMAX by KMAX more tridiagonal solutions in 3-D. Last, this grid should be adapted with an adaptive grid generation procedure that utilizes a space marching flow solution.

Bibliography

1. Anderson, Dale A., Tannehill, John C., and Pletcher, Richard H., **Computational Fluid Mechanics and Heat Transfer**. New York: Hemisphere Corporation. (1984)

2. Hodge, James K., Lecture materials distributed in AE 7.51, Computational Aerodynamics I. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH. (January-April 1985)

3. Thompson, Joe F., "Elliptic Grid Generation," **Numerical Grid Generation**. Joe Thompson (ed.), New York: Elsevier Science Publishing Co., Inc., (1982)

4. Sorenson, R.L. and Steger, J.L., "Grid Generation in Three Dimensions by Poisson Equations with Control of Cell Size and Skewness at Boundary Surfaces," **Advances in Grid Generation**. Ghia, U. (ed.), ASME, FED-Vol. 5, (1983), (Also in AFWAL-TR-83-3129)

5. Steger, J.L. and Chaussee, D.S., "Generation of Body Fitted Coordinates Using Hyperbolic Partial Differential Equations," **SIAM J. Sci. Stat. Comput., Vol. 1, No. 4**, (Dec. 1980)

6. Nakamura, S., "Marching Grid Generation Using Parabolic Partial Differential Equations," **Numerical Grid Generation**. Thompson, J.F.(ed.), Elsevier Science Publishing Co., Inc., (1982)

7. Edwards, T.A., "Noniterative Three-Dimensional Grid Generation Using Parabolic Partial Differential Equations," AIAA 23rd Aerospace Sciences Meeting, AIAA-85-0485, (January 14-17,1985)

8. Noack, R.W., "Inviscid Flow Field Analysis of Maneuvering Hypersonic Vehicles Using the SCM Formulation and Parabolic Grid Generation," AIAA 18th Fluid Dynamics and Plasmadynamics and Lasers Conference, AIAA-85-1682, (July 16-18, 1985)

9. Hodge, Major James K., Leone, Capt Sal A., and McCarty, Capt Robert L., "Non-iterative Parabolic Grid Generation for Parabolized Equations," AIAA 7th Computational Fluid Dynamics Conference, AIAA-85-1528, (July 15-17, 1985)

10. Rasmmussen, M.L., Lecture materials distributed in AE 6.30, Re-Entry Aerodynamics. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, (April-June 1985)

11. Kim, B.S., Rasmussen, M.L., and Jischke, M.C., "Optimization of Waverider Configurations Generated form Axisymmetric Conical Flows," AIAA 9th Atmospheric Flight Mechanics Conference, AIAA-82-1299, (August 9-11, 1982)

12. Roscoe,D.F., "New Methods for the Derivation of Stable Difference Represen-taions for Differential Equations." **Journal to Institute of Mathemathical Applications**, **Vol. 16**, pgs. 291-321, (1975)

13. Chakravarthy, S.W. "Numerical Treatment of Inviscid Flow Past Sharp Edges and Corners." AIAA 17th Fluid Dynamics and Plasmadynamics and Lasers Conference, AIAA-84-1647, (June 25-27, 1984)

14. Rizk, Yehia, NASA-Ames, "Numerical Solution of Supersonic Viscous Regions Around 3-D Winged Configurations." Presentation at 1985 Parabolized Navier-Stokes Code Workshop, Flight Dynamics Laboratory, Wright-Patterson AFB OH, (23-26 September 1985)

Appendix A: Two-Dimensional Slit Transformed Grid Procedure

A.1 Computational Domain

Another means of generating a grid around the waverider's wing tip is to use a slit transformed grid procedure. The previous grids were transformed from physical space into a rectangular computational plane in computational space, as shown in Fig. (A-1). One side of this rectangular computational plane represents the inner boundary, the opposite side represents the outer boundary, and the other two sides represent the plane-of-symmetry above the upper surface and the plane-of-symmetry below the lower surface. The slit transformation requires that an extra line be constructed inside of the rectangular computational plane, as shown in Fig. (A-1). This extra line leaves the inner boundary side of the rectangular computational plane and proceeds part way towards the outer boundary side of the rectangular computational plane. This slit represents the slender wing. The inner boundary side of the computational plane is now split by the slit. The side of the computational plane adjacent to the slit on the left represents the lower surface of the waverider up to the segment represented by the slit. The other side of the computational plane adjacent to the slit represents the upper surface of the waverider up to the segment represented by the slit. The slit has double values on it, one for the lower wing surface and one for the upper wing surface. Where the slit leaves the wing and becomes a grid line, the double values are equal to each other.

A.2 Boundary Points

The boundary points can be determined now that the computational plane is defined. The outer boundary is to be divided into two sections. The division is at the intersection of the bisecting line of the wing tip with the outer boundary. This

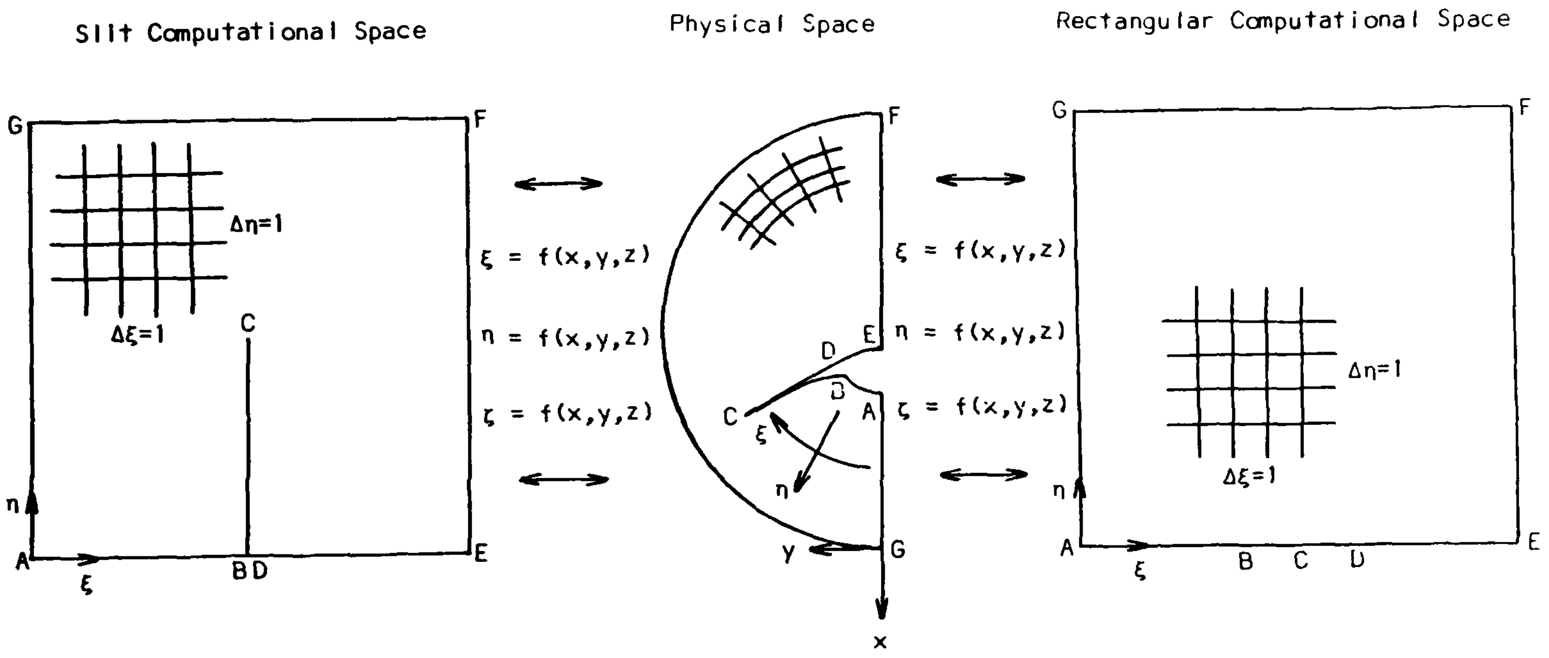


Figure A-1. Transformation of Physical Space into Either a Rectangular or a Slit Computational Space was previously discussed in the 2-D wrapped grid procedure, Section 4.1.2. The waverider geometry that the slit will physically represent has to be determined. The surface of the wing is analogous to a two-sided grid line, therefore the proper Q distribution of points is placed on each side of the slit. In order for the doubled-valued slit to have the same values beyond the wing tip, the same Q distribution will have to be placed on the double-valued slit over the same interval. In order for the lengths to be the same, the arc length of slit surfaces must be equal. Therefore, the arc length of the upper and lower surfaces of the waverider must be equal in order to generate the proper grid points beyond the wing tip. This severly reduces flexibility. And there is no guarantee that the wing tip will be represented by one of these points.

After the surfaces that will be represented by the slit have been determined, points can be distributed along the surface of the geometry that are not represented by the slit. Clustering grid points near the waverider's surface not represented by the slit is accomplished by using a positive value of P. Points have to be highly-clustered away from the plane-of-symmetry on the upper and lower surfaces that are not represented by the slit, to get the grid lines to lie near the surface represented by the slit. Points also have to be highly clustered away from the plane-of-symmetry on both segments of the outer boundary near the bisection point. The grid does not need to be dense near the outer boundary, because of the nature of the far field. The slit transformed grid becomes expensive to use with a flow solver, because of the large number of points needed to represent the grid.

A.3 Orthogonal Projected Solutions

The goal of this thesis is to generate nearly orthogonal grids around a waverider. The slit transformation procedure is not too difficult if a non-orthogonal grid is constructed. Since the waverider's surface is highly curved, the projected solution has to take on the same shape as the wing's surface in the vicinity of the wing's surface in order to cluster points there. The projected solution will not only have to take on the shape of the wing's surface in the vicinity of wing, but also an orthogonal curve off of the main body's surface. There are too many constraints to contend with. *Orthogonality can be enforced near the plane-of-symmetries.* But as the wing's surface is approached the projected solution has to take on the shape of the wing's surface and not the shape of an orthogonal curve. Some weighting criteria must be introduced. To get grid lines to be nearly orthogonal on the wing's surface, either the points on the wing must be moved or the points in the projected solution must be moved. Moving the specified points on the wing is not the solution. Moving the points in the projected solution would change the Q distribution of points and affect whatever representation of orthogonality that is left. Near orthogonality to the slit requires that points be altered more near the wing's surface and less the farther the solution is away from the wing's surface.

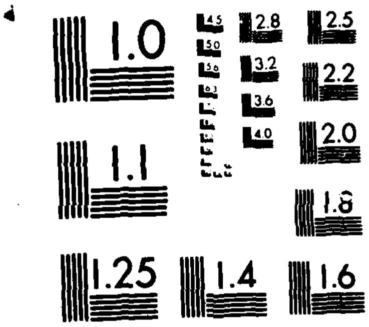
The slit transformation procedure is complicated if a nearly orthogonal grid is to be generated around the waverider. Many points are needed to represent the grid properly. Points are inappropriately clustered on the outer boundary near the wing tip bisection. The grid can not be orthogonal and cluster points near the geometry's surface. It is difficult to specify orthogonality in two directions. Since there is not enough time to program this procedure along with the problems stated above, the 2-D slit transformed grid will not be programmed.

Appendix B Two-Dimensional Wrapped Grid Computer Code

```
      PROGRAM GRID2D
      COMMON/SET/DIMEN,RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,P,Q,IPLUS,MIN
     +OR,MAJOR,CENTRA,ITERAT,TEST,CONVEX,IPM,ORTHO,JCENT
      COMMON/TRI/A(100),C(100),D(100)
      COMMON/BETA/BETA,PI
      COMMON/BOUND/X1(100),Y1(100),X2(100),Y2(100)
      COMMON ANS(100)
      COMMON/ARC/APHI(0:180),ARB(0:180),ARFS(0:180),ARCB(0:180),ARCFS(0:
     +180),ARE(0:180),ARCE(0:180)
      COMMON/LENGTH/BARC,FSARC,EARC
      DIMENSION X(100),Y(100),XM1(100),YM1(100),XPROJ(100),YPROJ(100)
      DIMENSION NODE(3),ARC(100),A2D(100),C2D(100),D2D(100,2)
      REAL MAJOR,MINOR,LENGTH(3),INTER1,INTER2,K
      LOGICAL CENTRA,TEST,CONVEX,ORTHO,TEST2,DIMEN
      REWIND 7
      PI=ACOS(-1.0)
C
C GET INITIAL VALUES FOR RUN
C
      CALL SETUP
      IMXM1=IMAX-1
      JMXM1=JMAX-1
      IF(TEST) THEN
C
C CALCULATES HALF CIRCLES FOR TEST CASE
C
      WRITE(*,'(A)')'ENTER RADIUS OF INNER TEST CIRCLE'
      READ(*,*) RINNER
      WRITE(*,'(A)')'ENTER RADIUS OF OUTER TEST CIRCLE'
      READ(*,*) ROUTER
C
C CALCULATES INNER AND OUTER BOUNDARY POINTS FOR TEST CASE
C
      DO 1 I=1,IMAX
      PHI=FLOAT(I-1)*PI/IMXM1
      X1(I)=RINNER*COS(PHI)
      Y1(I)=RINNER*SIN(PHI)
      X2(I)=ROUTER*COS(PHI)
    1 Y2(I)=ROUTER*SIN(PHI)
      ELSE
C
C CALCULATE WAVERIDER, ELLIPSE BOUNDARY POINTS
C CALCULATE SHOCK ANGLE
C
      CALL SHOCK(BETA)
C
C CALCULATE ARCLENGTH OF OUTER BOUNDARY ELLIPSE, AND LOWER AND UPPER SURFACES
C OF WAVERIDER.  EARC, ARCLENGTH OF ELLIPSE.  BARC, ARCLENGTH OF LOWER SURFACE.
C FSARC, ARCLENGTH OF FREESTREAM OR UPPER SURFACE OF WAVERIDER.
C
      CALL ARCLEN
      I=0
      LENGTH(1)=EARC
      LENGTH(2)=BARC
      LENGTH(3)=FSARC
      NODE(1)=IMAX
    2 I=I+1
C
C GENERATE COEFFICIENTS FOR 1-D ELLIPTIC EQUATION
C
      BB=1.+EXP(-ABS(P))
      AA=-1./BB
      CC=-EXP(-ABS(P))/BB
      A(1)=0.0
      A(NODE(I))=0.0
      C(1)=0.0
      C(NODE(I))=0.0
      D(1)=0.0
      ANS(1)=0.0
      D(NODE(I))=LENGTH(I)
      ANS(NODE(I))=LENGTH(I)
       IF(P .LE. 0.0) THEN
       DO 5 J=2,NODE(I)-1
       A(J)=AA
       C(J)=CC
    5  D(J)=0.0
       ELSE
       DO 10 J=2,NODE(I)-1
       A(J)=CC
       C(J)=AA
   10  D(J)=0.0
       ENDIF
C
C SOLVE 1-D ELLIPTIC EQUATION FOR DISTRIBUTION P AND FOR NODE(I) POINTS.
C
      CALL TRIDIG(NODE(I)-1)
C
C NOW THAT THE ARCLENGTH DISTRIBUTION IS KNOWN, CONVERT ARCLENGTH INTO ELLIPSE
C AND WAVERIDER BOUNDARY POINTS.
C
      CALL INTERP(NODE(I),I)
C
C CALCULATE THE INTERSECTION OF LINE BISECTING UPPER AND LOWER SURFACES OF
C WAVERIDER AND THE OUTER BOUNDARY.
C
      P=0.0
       IF(I .EQ. 1) THEN
       SLIMIT=10000.
C
C GO 5 DEGREES BACK FROM THE WING TIP AT PHIL AND GET X3,Y3 ON LOWER BODY
C SURFACE AND X4,Y4 ON UPPER BODY SURFACE.  AVERAGE X3,Y3 AND X4,Y4 TO GET
C XMID,YMID POINT.
C
       IPHIL=PHIL
       IPOS=IPHIL-5
       ANGLE=FLOAT(IPOS)*PI/180.
       X3=ARB(IPOS)*COS(ANGLE)
       Y3=ARB(IPOS)*SIN(ANGLE)
       X4=ARFS(IPOS)*COS(ANGLE)
       Y4=ARFS(IPOS)*SIN(ANGLE)
C
C CALCULATE POINT ON WING TIP
C
       XMID=X3+(X4-X3)/2.
       YMID=Y3+(Y4-Y3)/2.
        IF(IPHIL .EQ. PHIL) THEN
        ANGLE=FLOAT(IPHIL)*PI/180.
        XEND=ARB(IPHIL)*COS(ANGLE)
        YEND=ARB(IPHIL)*SIN(ANGLE)
        ELSE
        ANGLE=PHIL*PI/180.
        XEND=ARB(IPHIL+1)*COS(ANGLE)
        YEND=ARB(IPHIL+1)*SIN(ANGLE)
        ENDIF
C
C CALCULATE SLOPE AND Y-INTERSECT OF LINE BISECTING WAVERIDER GEOMETRY.
C THERE ARE TWO SPECIAL CASES WHEN THE SLOPE IS EQUAL TO ZERO OR INFINITY.
C
        IF(XEND-XMID .EQ. 0.0) THEN
        XCROSS=0.0
        YCROSS=MAJOR
        ELSE
        SLOPE1=(YEND-YMID)/(XEND-XMID)
         IF(SLOPE1 .EQ. 0.0) THEN
         XCROSS=MINOR
         YCROSS=0.0
         ELSE
         INTER1=YMID-SLOPE1*XMID
C
C CALCULATE INTERSECTION POINT OF BISECTING LINE AND OUTER BOUNDARY
C ONLY THE POSITIVE RADICAL TERM OF THE QUADRATIC EQUATION IS NEEDED
C
         XCROSS=(-2.*MINOR**2*SLOPE1*INTER1+(4.*MINOR**4*SLOPE1**2*INTER
     +   1**2-4.*(MAJOR**2+MINOR**2*SLOPE1**2)*(MINOR**2*INTER1**2-MAJOR
     +   **2*MINOR**2))**.5)/(2.*(MAJOR**2+MINOR**2*SLOPE1**2))
         YCROSS=SLOPE1*XCROSS+INTER1
         ENDIF
        ENDIF
C
C DETERMINE WHICH POINT ON THE OUTER BOUNDARY IS CLOSEST TO THE INTERSECTION
C POINT
C
       DO 3 J=1,IMAX
       XDELTA=ABS(X2(J)-XCROSS)
       YDELTA=ABS(Y2(J)-YCROSS)
       DISTAN=(XDELTA**2+YDELTA**2)**.5
        IF(DISTAN .LE. SLIMIT) THEN
        SLIMIT=DISTAN
        NODE(2)=J
        ENDIF
    3  CONTINUE
C
C NODE(2) IS THE NUMBER OF POINTS TO BE PLACED ON LOWER SURFACE.  IF POINTS ARE
C REFINED, SUBTRACT HALF THE REFINED POINTS FROM THE LOWER SURFACE TO GET THE
C PROPER DISTRIBUTION OF POINTS ON THAT SURFACE.
C
       NODE(2)=NODE(2)-IPM/2
       NODE(3)=IMAX-NODE(2)+1-IPM
       ENDIF
C
C INITIALIZE INNER BOUNDARY POINT ON OTHER SIDE OF PLANE OF SYMMETRY, TO BE USED
C IN ORTHOGONALITY GENERATION.
C
      IF(I .LT. 3) GO TO 2
      ENDIF
      X1(IMAX+1)=X1(IMAX-1)
      Y1(IMAX+1)=-Y1(IMAX-1)
C
C INITIALIZE GRID LOCATION COUNTER, AQ, EQ, AND K
C
      CALL SECOND(CPI)
      ICOUNT=-2
      AQ=ABS(Q)
      IF(AQ .LT. 1.E-6) THEN
      EQ=1.-AQ+AQ**2/2.-AQ**3/6.+AQ**4/24.-AQ**5/120.
      K=1./(1.-AQ/2.+AQ**2/6.-AQ**3/24.+AQ**4/120.)
      ELSE
      EQ=EXP(-AQ)
      K=AQ/(1-EQ)
      ENDIF
C
C GENERATE COEFFICIENTS FOR 1-D ELLIPTIC EQUATION
C
      A(1)=0.0
      A(JMAX)=0.0
      C(1)=0.0
      C(JMAX)=0.0
      D(1)=0.0
      ANS(1)=D(1)
      D(JMAX)=10.0
      ANS(JMAX)=D(JMAX)
      BB=1.+EXP(-ABS(Q))
      AA=-1./BB
      CC=-EXP(-ABS(Q))/BB
      IF(Q .LE. 0.0) THEN
      DO 16 J=2,JMXM1
      A(J)=AA
      C(J)=CC
   16 D(J)=0.0
      ELSE
      DO 17 J=2,JMXM1
      A(J)=CC
      C(J)=AA
   17 D(J)=0.0
      ENDIF
C
C SOLVE 1-D ELLIPTIC EQUATION
C
      CALL TRIDIG(JMXM1)
C
C SINCE Q IS CONSTANT THROUGHOUT GRID, SOLVE 1-D ELLIPTIC EQUATION ONCE AND
C STORE IN ARC AND THEN CAN SCALE FOR BOTH INITIAL GUESS AT I AND AT IPLUS, THE
C PROJECTED SOLUTION
C
      DO 18 J=1,JMAX
   18 ARC(J)=ANS(J)
C
C BEGINNING OF 2-D GRID GENERATION LOOP, INCREMENT COUNTER
C JPLUS IS THE LOCATION OF PROJECTED SOLUTION, JPLSM1 IS IPLUS-I IN 2-D EQUATION
C
      IF(TEST) GO TO 20
      TEST2=.TRUE.
      LCOUNT=0
   19 LCOUNT=LCOUNT+1
      IF(LCOUNT .EQ. 3) THEN
      TEST2=.FALSE.
      IF(ANG1 .GT. PI) ANG1=ANG1-2.*PI
      ANG3=(ANG2-ANG1)/FLOAT(IPM)
      GO TO 20
      ENDIF
      IF(LCOUNT .EQ. 1) THEN
      JPLUS=NODE(2)-1
      ELSE
      JPLUS=NODE(2)+IPM+1
      ENDIF
      GO TO 58
   20 ICOUNT=ICOUNT+1
      JPLUS=ICOUNT+IPLUS
      JPLSM1=IPLUS
C
C TO START 2-D GRID GENERATION, PROJECT SOLUTION AT SECOND POSITION AND REFLECT
C ON OTHER SIDE OF PLANE OF SYMMETRY.  NEED A GUESS AT POSITION 1 AND NEED
C PROJECTED SOLUTION AT SECOND POSITION.  THIS FORCES IPLUS TO EQUAL ONE AT
C PLANE OF SYMMETRY.
C
      IF(ICOUNT .EQ. -1) JPLUS=2
      IF(ICOUNT .EQ. 0) JPLUS=1
      IF(ICOUNT .EQ. 1) THEN
      JPLUS=2
      JPLSM1=1
      ENDIF
C
C DON'T WANT PROJECTED SOLUTION TO PASS FAR PLANE OF SYMMETRY
C
      IF(JPLUS .GT. IMAX) THEN
```

AD-A164 020 NUMERICAL GRID GENERATION FOR PARABOLIC PARTIAL 2/2
DIFFERENTIAL EQUATIONS US..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. S G MILLER
UNCLASSIFIED DEC 85 AFIT/GA/AA/85D-7 F/G 20/4 NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      JPLUS=IMAX
      JPLSM1=JPLUS-ICOUNT
      ENDIF
C
C HAVE TO DO THE SAME KIND OF SOLUTION AT THE FAR PLANE OF SYMMETRY AS DONE IN
C STARTING THE ORIGINAL SOLUTION
C
      IF(ICOUNT .EQ. IMAX) THEN
      JPLUS=IMXM1
      JPLSM1=1
      DO 21 I=1,JMAX
      XPROJ(I)=XM1(I)
   21 YPROJ(I)=-YM1(I)
      GO TO 56
      ENDIF
C
C DO A CHECK ON CONVEXITY IF NOT SOLVING EITHER PLANE OF SYMMETRY AND THE
C CONVEXITY SWITCH IS ON
C
      IF(ICOUNT .GT. 1 .AND. ICOUNT .LT. IMAX .AND. CONVEX) THEN
C
C DETERMINE THE SCALING FACTOR FOR CURRENT JPLUS LOCATION FROM THE STORED 1-D
C ELLIPTIC DISTRIBUTION
C
   22 FACT1=(X2(JPLUS)-X1(JPLUS))/10.
      FACT2=(Y2(JPLUS)-Y1(JPLUS))/10.
      DIFF=ARC(2)-ARC(1)
C
C CALCULATE THE SECOND POINT FROM THE PROJECTED SOLUTION AWAY FROM BODY SURFACE
C
      XPROJ(2)=X1(JPLUS)+DIFF*FACT1
      YPROJ(2)=Y1(JPLUS)+DIFF*FACT2
C
C GET EQUATION OF LINE FROM THE SECOND POINT OF PROJECTED SOLUTION TO THE SECOND
C POINT OF THE PRIOR SOLVED SOLUTION
C
      SLOPE1=(YPROJ(2)-YM1(2))/(XPROJ(2)-XM1(2))
      INTER1=YM1(2)-SLOPE1*XM1(2)
C
C CHECK IF BODY INTERSECTS THE LINE JUST CALCULATED.  CALCULATE EQUATION OF LINE
C FOR EACH GRID LINE BETWEEN PRIOR SOLUTION AND PROJECTED SOLUTION AND DETERMINE
C WHETHER THE SURFACE IS TOO CONVEX
C
      DO 23 I=JPLUS-1,ICOUNT,-1
      SLOPE2=(Y2(I)-Y1(I))/(X2(I)-X1(I))
      INTER2=Y1(I)-SLOPE2*X1(I)
      XCROSS=(INTER2-INTER1)/(SLOPE1-SLOPE2)
       IF(X1(I) .LT. X2(I)) THEN
        IF(XCROSS .GT. X1(I) .AND. XCROSS .LT. X2(I)) THEN
        ELSE
C
C SURFACE TOO CONVEX, MOVE PROJECTED SOLUTION BACK ONE
C
```

```
        JPLUS=JPLUS-1
        JPLSN1=JPLSN1-1
         IF(JPLUS-ICOUNT .EQ. 1) THEN
C
C IPLUS IS NOW EQUAL TO ONE AND THIS IS AS FAR AS WE CAN GO AND STILL GET A
C PROJECTED SOLUTION
C
         FACT1=(X2(JPLUS)-X1(JPLUS))/10.
         FACT2=(Y2(JPLUS)-Y1(JPLUS))/10.
         GO TO 26
         ENDIF
        GO TO 22
        ENDIF
       ELSE
        IF(XCROSS .LT. X1(I) .AND. XCROSS .GT. X2(I)) THEN
        ELSE
        JPLUS=JPLUS-1
        JPLSN1=JPLSN1-1
         IF(JPLUS-ICOUNT .EQ. 1) THEN
         FACT1=(X2(JPLUS)-X1(JPLUS))/10.
         FACT2=(Y2(JPLUS)-Y1(JPLUS))/10.
         GO TO 26
         ENDIF
        GO TO 22
        ENDIF
       ENDIF
   23 CONTINUE
C
C NOW THAT WE DETERMINED A GOOD IPLUS LOCATION, LOAD THE PROJECTED SOLUTION
C WITH SCALED VALUES OF 1-D ELLIPTIC DISTRIBUTION
C
   26 XPROJ(1)=X1(JPLUS)
      XPROJ(JMAX)=X2(JPLUS)
      YPROJ(1)=Y1(JPLUS)
      YPROJ(JMAX)=Y2(JPLUS)
      DO 24 I=2,JMXM1
      DIFF=ARC(I)-ARC(I-1)
      XPROJ(I)=XPROJ(I-1)+DIFF*FACT1
   24 YPROJ(I)=YPROJ(I-1)+DIFF*FACT2
      ELSE
C
C LOAD PROJECTED SOLUTION FROM SCALED VALUES WITHOUT CHECKING FOR CONVEXITY
C
      XPROJ(1)=X1(JPLUS)
      XPROJ(JMAX)=X2(JPLUS)
      YPROJ(1)=Y1(JPLUS)
      YPROJ(JMAX)=Y2(JPLUS)
      FACT1=(XPROJ(JMAX)-XPROJ(1))/10.
      FACT2=(YPROJ(JMAX)-YPROJ(1))/10.
      DO 25 I=2,JMXM1
      DIFF=ARC(I)-ARC(I-1)
      XPROJ(I)=XPROJ(I-1)+DIFF*FACT1
   25 YPROJ(I)=YPROJ(I-1)+DIFF*FACT2
```

```
      ENDIF
C
C LOAD INITIAL GUESS TO START 2-D SOLUTION
C
      IF(ICOUNT .EQ. 0) THEN
      DO 55 I=1,JMAX
      X(I)=XPROJ(I)
   55 Y(I)=YPROJ(I)
      GO TO 20
      ENDIF
C
C IF ORTHOGONALITY SWITCH IS ON, TURN THE PROJECTED SOLUTION FROM A STRAIGHT
C TO A CURVE THAT STARTS ORTHOGONAL TO SURFACE AND ENDS UP AT OUTER BOUNDARY
C SURFACE
C
       IF(.NOT.ORTHO) GO TO 61
C
C CALCULATE ANGLE OF LINE BETWEEN INNER AND OUTER BOUNDARIES
C
       THETA1=ATAN((Y2(JPLUS)-Y1(JPLUS))/(X2(JPLUS)-X1(JPLUS)))
C
C PUT ANGLE IN PROPER QUADRAT, HAVE SPECIAL CONSIDERATIONS WHEN SLOPES ARE
C EITHER ZERO OR INFINITY
C
        IF(X2(JPLUS)-X1(JPLUS) .EQ. 0.0) THEN
        FACT1=1.
        ELSE
        FACT1=(X2(JPLUS)-X1(JPLUS))/ABS(X2(JPLUS)-X1(JPLUS))
        ENDIF
        IF(Y2(JPLUS)-Y1(JPLUS) .EQ. 0.0) THEN
        FACT2=1.
        ELSE
        FACT2=(Y2(JPLUS)-Y1(JPLUS))/ABS(Y2(JPLUS)-Y1(JPLUS))
        ENDIF
C
C SINCE QUADRAT NOW KNOWN, CHANGE ANGLE TO REFLECT QUADRAT
C
       IF(FACT1 .EQ. -1 .AND. FACT2 .EQ. 1) THETA1=PI+THETA1
       IF(FACT1 .EQ. -1 .AND. FACT2 .EQ. -1) THETA1=PI+THETA1
       IF(FACT1 .EQ. 1 .AND. FACT2 .EQ. -1) THETA1=2*PI+THETA1
C
C CALCULATE ANGLE OF NORMAL LINE TO BODY SURFACE USING AN AVERAGE OF PANELS ON
C EACH SIDE OF THE PROJECTED SOLUTION
C
       IF(JPLUS .GT. NODE(2)-1 .AND. JPLUS .LT. NODE(2)+IPM+1) THEN
       THETA2=ANG1+ANG3*FLOAT(ICOUNT-NODE(2))
       GO TO 57
       ENDIF
   58  THETA2=(ATAN((X1(JPLUS-1)-X1(JPLUS))/(Y1(JPLUS)-Y1(JPLUS-1)))+ATA
     + N((X1(JPLUS)-X1(JPLUS+1))/(Y1(JPLUS+1)-Y1(JPLUS))))/2.
       SLOPE1=TAN(THETA2)
C
C HAVE A QUADRAT PROBLEM AGAIN.  HAVE SLOPE AND POINT AND THEREFORE EQUATION
```

```
C OF LINE.  INTERSECT THIS LINE WITH OUTER BOUNDARY WITH BOTH POSITIVE AND
C NEGATIVE SIGNS ON THE RADICAL OF THE QUADRATIC EQUATION.  SPECIAL PROBLEMS
C AGAIN WITH SLOPES EQUAL TO ZERO AND INFINITY
C
        IF(SLOPE1 .EQ. 0.0) THEN
        YQUAD1=Y1(JPLUS)
        YQUAD2=Y1(JPLUS)
        XQUAD1=((1.-Y1(JPLUS)**2/MAJOR**2)*MINOR**2)**.5
        XQUAD2=-XQUAD1
        ELSE
         IF(SLOPE1 .GT. 10000.) THEN
         XQUAD1=X1(JPLUS)
         XQUAD2=X1(JPLUS)
         YQUAD1=((1.-X1(JPLUS)**2/MINOR**2)*MAJOR**2)**.5
         YQUAD2=-YQUAD1
         ELSE
         INTER1=Y1(JPLUS)-SLOPE1*X1(JPLUS)
         FACT1=-2*MINOR**2*SLOPE1*INTER1
         FACT2=2*(MAJOR**2+MINOR**2*SLOPE1**2)
         RADICA=(FACT1**2-2.*FACT2*(MINOR**2*INTER1**2-MAJOR**2*MINOR**2
     +   ))**.5
         XQUAD1=(FACT1+RADICA)/FACT2
         XQUAD2=(FACT1-RADICA)/FACT2
         YQUAD1=SLOPE1*XQUAD1+INTER1
         YQUAD2=SLOPE1*XQUAD2+INTER1
         ENDIF
        ENDIF
C
C CALCULATE DISTANCE BETWEEN INTERSECTIONS AND OUTER BOUNDARY POINT.  THE
C SHORTER DISTANCE DETERMINES THE PROPER QUADRAT
C
       QUAD1=((XQUAD1-X2(JPLUS))**2+(YQUAD1-Y2(JPLUS))**2)**.5
       QUAD2=((XQUAD2-X2(JPLUS))**2+(YQUAD2-Y2(JPLUS))**2)**.5
        IF(QUAD1 .LT. QUAD2) THEN
        XCROSS=XQUAD1
        YCROSS=YQUAD1
        ELSE
        XCROSS=XQUAD2
        YCROSS=YQUAD2
        ENDIF
        IF(XCROSS -X1(JPLUS) .EQ. 0.0) THEN
        FACT1=1.
        ELSE
        FACT1=(XCROSS-X1(JPLUS))/ABS(XCROSS-X1(JPLUS))
        ENDIF
        IF(YCROSS-Y1(JPLUS) .EQ. 0.0) THEN
        FACT2=1.
        ELSE
        FACT2=(YCROSS-Y1(JPLUS))/ABS(YCROSS-Y1(JPLUS))
        ENDIF
C
C CHANGE ANGLE TO REFLECT QUADRAT
C
```

```
      IF(FACT1 .EQ. -1 .AND. FACT2 .EQ. 1) THETA2=PI+THETA2
      IF(FACT1 .EQ. -1 .AND. FACT2 .EQ. -1) THETA2=PI-THETA2
      IF(FACT1 .EQ. 1 .AND. FACT2 .EQ. -1) THETA2=2*PI+THETA2
      IF(TEST2) THEN
      IF(LCOUNT .EQ. 1) THEN
      ANG1=THETA2
      ELSE
      ANG2=THETA2
      ENDIF
      GO TO 19
      ENDIF
C
C FACTOR USED TO ACCELERATE OR RETARD HYPERBOLIC TANGENT
C
   57 IF(THETA2-THETA1 .EQ. 0.0) THEN
      ATH=25.
      ELSE
      ATH=1./ABS(THETA2-THETA1)
      IF(ATH .GT. 25.) ATH=25.
      ENDIF
C
C LENGTH OF PROJECTED SOLUTION
C
      SPAN=((X2(JPLUS)-X1(JPLUS))**2+(Y2(JPLUS)-Y1(JPLUS))**2)**.5
C
C CENTER OF PROJECTED SOLUTION WITH RESPECT TO THE MIDDLE POINT
C
      CENTER=(((XPROJ(JCENT)-XPROJ(1))**2+(YPROJ(JCENT)-YPROJ(1))**2)**
     + .5)/SPAN
C
C CHECK TO SEE WHERE THE HYPERBOLIC TANGENT SHOULD BE CENTERED, SO THERE WON'T
C BE GRID CROSS OVER DUE TO THE ORTHOGONALITY.  CALCULATE EQUATION OF
C ORTHOGONAL IPLUS LINE AND AT PRIOR STATION.  CALCULATE INTERSECTION AND USE
C MORE RESTRICTIVE CENTERING CRITERION.
C
      IF(DIMEN) THEN
      CHANGE=10000.
       IF(JPLUS .GT. 2 .AND. JPLUS .LT. NODE(2) .OR. JPLUS .GT.
     + NODE(2)+IPM+1 .AND. JPLUS .LT. IMAX) THEN
       SLOPE1=TAN(THETA4)
       SLOPE2=TAN(THETA2)
       INTER1=Y1(JPLUS-1)-SLOPE1*X1(JPLUS-1)
       INTER2=Y1(JPLUS)-SLOPE2*X1(JPLUS)
       XCROSS=(INTER2-INTER1)/(SLOPE1-SLOPE2)
       YCROSS=SLOPE2*XCROSS+INTER2
        IF(XPROJ(JMAX) .GT. XPROJ(1)) THEN
         IF(XCROSS .GT. XPROJ(1) .AND. XCROSS .LT. XPROJ(JMAX)) THEN
         CHANGE=((XCROSS-XPROJ(1))**2+(YCROSS-YPROJ(1))**2)**.5/SPAN
     +   -.05/SPAN
         ENDIF
        ELSE
         IF(XCROSS .LT. XPROJ(1) .AND. XCROSS .GT. XPROJ(JMAX)) THEN
         CHANGE=((XCROSS-XPROJ(1))**2+(YCROSS-YPROJ(1))**2)**.5/SPAN
     +    -.05/SPAN
          ENDIF
         ENDIF
        ENDIF
       IF(CHANGE .LE. CENTER) CENTER=CHANGE
       THETA4=THETA2
       ENDIF
       DO 54 I=2,JMXM1
       R=((XPROJ(I)-XPROJ(1))**2+(YPROJ(I)-YPROJ(1))**2)**.5
       ANGLE=ATH*(R/SPAN-CENTER)
        IF(I .EQ. 2) THEN
        ANGLE1=TANH(ANGLE)
        ANGLE2=TANH(-ATH*(1.-CENTER))
        ENDIF
        FACTOR=FLOAT(JMXM1-I)*ANGLE1/FLOAT(JMXM1-2)+ANGLE2*((FLOAT(I-JMX
     + M1)/FLOAT(JMXM1-2))+1.)
C
C ANGLE THAT STARTS OUT AT THE ORTHOGONAL ANGLE AND GOES TO THE STRAIGHT LINE
C ANGLE
C
        IF(ABS(THETA2-THETA1) .GT. PI*.5) THEN
C
C FOR ANGLES THAT ARE IN BOTH FIRST AND FOURTH QUADRAT
C
        THETA3=(TANH(ANGLE)/FACTOR*(THETA2-2*PI-THETA1)+THETA2-2*PI+THET
     +  A1)/2
        ELSE
        THETA3=(TANH(ANGLE)/FACTOR*(THETA2-THETA1)+THETA2+THETA1)/2
       ENDIF
C
C NEW ORTHOGONAL PROJECTED SOLUTION
C
       XPROJ(I)=R*COS(THETA3)+XPROJ(1)
   54  YPROJ(I)=R*SIN(THETA3)+YPROJ(1)
C
C LOAD INITIAL SOLVED SOLUTION ON OTHER SIDE OF THE PLANE OF SYMMETRY
C
   61 IF(ICOUNT .EQ. -1) THEN
      DO 50 I=1,JMAX
      XM1(I)=XPROJ(I)
   50 YM1(I)=-YPROJ(I)
      GO TO 20
      ENDIF
C
C ITERATION LOOP
C
   56 DO 85 I=1,ITERAT
C
C USE SCALED 1-D DISTRIBUTION TO CALCULATE INITIAL GUESS IF ON FIRST ITERATION
C ONLY.  SUCCESSIVE ITERATIONS WILL USE VALUES CALCULATED FROM PREVIOUS
C ITERATION.
C
      IF(I .EQ. 1) THEN
      X(1)=X1(ICOUNT)
      X(JMAX)=X2(ICOUNT)
      Y(1)=Y1(ICOUNT)
      Y(JMAX)=Y2(ICOUNT)
      FACT1=(X(JMAX)-X(1))/10.
      FACT2=(Y(JMAX)-Y(1))/10.
      DO 60 J=2,JMXM1
      DIFF=ARC(J)-ARC(J-1)
      X(J)=X(J-1)+DIFF*FACT1
   60 Y(J)=Y(J-1)+DIFF*FACT2
      ENDIF
C
C FIRST TIME THROUGH THE LOOP, IT CALCULATES THE X-COMPONENT, THE SECOND TIME
C THROUGH THE LOOP, IT CALCULATES THE Y-COMPONENT.
C
      A2D(1)=0.0
      A2D(JMAX)=0.0
      C2D(1)=0.0
      C2D(JMAX)=0.0
      D2D(1,1)=X(1)
      D2D(JMAX,1)=X(JMAX)
      D2D(1,2)=Y(1)
      D2D(JMAX,2)=Y(JMAX)
      DO 65 L=2,JMXM1
      LP1=L+1
      LM1=L-1
C
C AT BOTH PLANES OF SYMMETRY, THE METRICS HAVE TO BE CALCULATED WITH
C CENTRAL DIFFERENCE.
C
      IF(ICOUNT .EQ. 1 .OR. ICOUNT .EQ. IMAX) THEN
      XETA=(X(LP1)-X(LM1))/2.
      YETA=(Y(LP1)-Y(LM1))/2.
      ALPH11=XETA**2+YETA**2
      ALPH22=0.0
      ALPH12=0.0
      ELSE
C
C DO CENTRAL DIFFERENCE ON METRICS IF IPLUS IS EQUAL TO 1 AND
C CENTRAL DIFFERENCING SWITCH IS ON.
C
       IF(JPLUS-ICOUNT .EQ. 1 .AND. CENTRA) THEN
       XXSI=(XPROJ(L)-XM1(L))/2.
       YXSI=(YPROJ(L)-YM1(L))/2.
       ELSE
C
C DO FIRST ORDER BACKWARD DIFFERENCING IF CENTRAL DIFFERENCING
C SWITCH IS OFF.
C
       XXSI=X(L)-XM1(L)
       YXSI=Y(L)-YM1(L)
       ENDIF
      XETA=(X(LP1)-X(LM1))/2.
      YETA=(Y(LP1)-Y(LM1))/2.
      ALPH11=XETA**2+YETA**2
      ALPH22=XXSI**2+YXSI**2
      ALPH12=-XXSI*XETA-YXSI*YETA
      ENDIF
      BBB=ALPH11*(1.+ABS(P)+1./FLOAT(JPLSM1))+ALPH22*K*(1.+EQ)
      AAA=-ALPH22*K/BBB
      CCC=-ALPH22*K*EQ/BBB
      IF(Q .LE. 0.0) THEN
      A2D(L)=AAA
      C2D(L)=CCC
      ELSE
      A2D(L)=CCC
      C2D(L)=AAA
      ENDIF
      DDD=ALPH12*(XPROJ(LP1)-XPROJ(LM1)-XM1(LP1)+XM1(LM1))/(FLOAT(JPLSM1
     +)+1.)
       IF(P .LE. 0.0) THEN
       D2D(L,1)=(ALPH11*(XM1(L)*(1.+ABS(P))+XPROJ(L)/FLOAT(JPLSM1))-DDD)
     + /BBB
       ELSE
       D2D(L,1)=(ALPH11*(XM1(L)+XPROJ(L)*(1.+ABS(P))/FLOAT(JPLSM1))-DDD)
     + /BBB
       ENDIF
      DDD=ALPH12*(YPROJ(LP1)-YPROJ(LM1)-YM1(LP1)+YM1(LM1))/(FLOAT(JPLSM1
     +)+1.)
       IF(P .LE. 0.0) THEN
       D2D(L,2)=(ALPH11*(YM1(L)*(1.+ABS(P))+YPROJ(L)/FLOAT(JPLSM1))-DDD)
     + /BBB
       ELSE
       D2D(L,2)=(ALPH11*(YM1(L)+YPROJ(L)*(1.+ABS(P))/FLOAT(JPLSM1))-DDD)
     + /BBB
       ENDIF
   65 CONTINUE
      DO 80 J=1,2
      DO 66 M=1,JMAX
      A(M)=A2D(M)
      C(M)=C2D(M)
   66 D(M)=D2D(M,J)
      ANS(1)=D(1)
      ANS(JMAX)=D(JMAX)
C
C SOLVE 2-D PARABOLIC EQUATION
C
      CALL TRIDIG(JMXM1)
C
C LOAD SOLUTION INTO X AND Y VECTORS
C
      IF(J .EQ. 1) THEN
      DO 70 M=2,JMXM1
   70 X(M)=ANS(M)
      ELSE
      DO 75 M=2,JMXM1
   75 Y(M)=ANS(M)
      ENDIF
   80 CONTINUE
   85 CONTINUE
C
C WRITE OUT SOLUTION
C
      DO 90 I=1,JMAX
   90 WRITE(7,'(2(F13.8,1X))')X(I),Y(I)
C
C MOVE PRESENT SOLUTION TO THE NOW PRIOR SOLUTION FOR NEXT MARCHING STEP
C
      DO 95 I=1,JMAX
      XM1(I)=X(I)
   95 YM1(I)=Y(I)
      IF(ICOUNT .LT. IMAX) GO TO 20
      CALL SECOND(CPF)
      CPU=CPF-CPI
      WRITE(*,'(A,F8.4)')'2-D GRID GENERATION TIME = ',CPU
      STOP
      END
      SUBROUTINE SETUP
      COMMON/SET/DIMEN,RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,P,Q,IPLUS,MIN
     +OR,MAJOR,CENTRA,ITERAT,TEST,CONVEX,IPM,ORTHO,JCENT
      LOGICAL DIMEN,CENTRA,TEST,CONVEX,ORTHO
      REAL MAJOR,MINOR
C
C INITIALIZE VARIABLES
C
      DIMEN=.TRUE.
      RMACH=5.0
      PHIL=60.0
      DELTA=5.0
      DELTA1=1.05
      N=8
      IMAX=60
      JMAX=20
      P=0.0
      Q=0.0
      IPLUS=1
      MAJOR=4.0
      CENTRA=.TRUE.
      ITERAT=1
      MINOR=4.0
      TEST=.FALSE.
      CONVEX=.TRUE.
      IPM=8
      ORTHO=.TRUE.
      JCENT=10
C
C WRITE OUT MENU
C
    5 WRITE(*,'(A)')'MENU TO GENERATE GRID AROUND A WAVERIDER'
      WRITE(*,'(A)')'0) GENERATE GRID'
      WRITE(*,'(A)')'1) RE-PRINT MENU'
      IF(DIMEN) THEN
      WRITE(*,'(A)')'2) AUTOMATIC GRID CROSSOVER CHECK ON'
      ELSE
      WRITE(*,'(A)')'2) AUTOMATIC GRID CROSSOVER CHECK OFF'
      ENDIF
      WRITE(*,'(A,F6.3)')'3) MACH NUMBER = ',RMACH
      WRITE(*,'(A,F6.3)')'4) DIHEDRAL ANGLE = ',PHIL
      WRITE(*,'(A,F6.3)')'5) BASIC-CONE SEMI-VERTEX ANGLE = ',DELTA
      WRITE(*,'(A,F6.3)')'6) BASIC-CONE SEMI-VERTEX ANGLE < DELTA1 < SHO
     +CK ANGLE, DELTA1 = ',DELTA1
      WRITE(*,'(A,I4)')'7) SHAPE FACTOR, AN EVEN INTEGER = ',N
      WRITE(*,'(A,I4)')'8) NUMBER OF GRID POINTS ALONG BODY SURFACE = ',
     +IMAX
      WRITE(*,'(A,I4)')'9) NUMBER OF GRID POINTS NORMAL FROM BODY SURFAC
     +E = ',JMAX
      WRITE(*,'(A,F6.3)')'10) STRETCHING IN BODY DIRECTION ON OUTER BOUN
     +DARY ONLY] = ',P
      WRITE(*,'(A,F6.3)')'11) STRETCHING IN DIRECTION NORMAL TO BODY SUR
     +FACE = ',Q
```

```
      WRITE(*,'(A,I4)')'12) PARABOIC UPSTREAM BOUNDARY = ',IPLUS
      WRITE(*,'(A,F6.3)')'13) SEMI-MAJOR AXIS FOR OUTER BOUNDARY ELLIPSE
     + = ',MAJOR
      WRITE(*,'(A,F6.3)')'14) SEMI-MINOR AXIS FOR OUTER BOUNDARY ELLIPSE
     + = ',MINOR
      IF(CENTRA) THEN
      WRITE(*,'(A)')'15) SECOND ORDER CENTRAL DIFFERENCING ON METRICS'
      ELSE
      WRITE(*,'(A)')'15) FIRST ORDER BACKWARDS DIFFERENCING ON METRICS'
      ENDIF
      WRITE(*,'(A,I4)')'16) NUMBER OF ITERATIONS = ',ITERAT
      IF(TEST) THEN
      WRITE(*,'(A)')'17) TEST CASE ON'
      ELSE
      WRITE(*,'(A)')'17) TEST CASE OFF'
      ENDIF
      IF(CONVEX) THEN
      WRITE(*,'(A)')'18) TEST FOR CONVEXITY ON'
      ELSE
      WRITE(*,'(A)')'18) TEST FOR CONVEXITY OFF'
      ENDIF
      WRITE(*,'(A)')'19) NUMBER OF EXTRA POINTS ON WING TIP'
      WRITE(*,'(A,I4)')'    (EVEN INTEGER) = ',IPM
      IF(ORTHO) THEN
      WRITE(*,'(A)')'20) GRID WILL BE SURFACE ORTHOGONAL'
      ELSE
      WRITE(*,'(A)')'20) GRID WILL NOT BE SURFACE ORTHOGONAL'
      ENDIF
      WRITE(*,'(A,I4)')'21) POINT AT WHICH HYPERBOLIC TANGENT IS CENTER
     +FOR ORTHOGONALITY = ',JCENT
   10 WRITE(*,'(A)')'ENTER DIRECTIVE'
      READ(*,*) IDIR
      IDIR=IDIR+1
C
C MODIFY VARIABLES
C
      GO TO (15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,
     +110,115,120),IDIR
   15 RETURN
   20 GO TO 5
   25 DIMEN=.NOT.DIMEN
      GO TO 10
   30 WRITE(*,'(A)')'ENTER MACH NUMBER'
      READ(*,*) RMACH
      GO TO 10
   35 WRITE(*,'(A)')'ENTER DIHEDRAL ANGLE'
      READ(*,*) PHIL
      GO TO 10
   40 WRITE(*,'(A)')'ENTER BASIC-CONE SEMI-VERTEX ANGLE'
      READ(*,*) DELTA
      GO TO 10
   45 WRITE(*,'(A)')'ENTER DELTA1'
      READ(*,*) DELTA1
      GO TO 10
   50 WRITE(*,'(A)')'ENTER SHAPE FACTOR'
      READ(*,*) N
      GO TO 10
   55 WRITE(*,'(A)')'ENTER IMAX'
      READ(*,*) IMAX
      GO TO 10
   60 WRITE(*,'(A)')'ENTER JMAX'
      READ(*,*) JMAX
      GO TO 10
   65 WRITE(*,'(A)')'ENTER STRETCHING FACTOR IN BODY DIRECTION'
      READ(*,*) P
      GO TO 10
   70 WRITE(*,'(A)')'ENTER STRETCHING FACTOR NORMAL TO BODY DIRECTION'
      READ(*,*) Q
      GO TO 10
   75 WRITE(*,'(A)')'ENTER UPSTREAM BOUNDARY'
      READ(*,*) IPLUS
      GO TO 10
   80 WRITE(*,'(A)')'ENTER SEMI-MAJOR AXIS FOR OUTER BOUNDARY ELLIPSE'
      READ(*,*) MAJOR
      GO TO 10
   85 WRITE(*,'(A)')'ENTER SEMI-MINOR AXIS FOR OUTER BOUNDARY ELLIPSE'
      READ(*,*) MINOR
      GO TO 10
   90 CENTRA=.NOT.CENTRA
      GO TO 10
   95 WRITE(*,'(A)')'ENTER NUMBER OF ITERATIONS'
      READ(*,*) ITERAT
      GO TO 10
  100 TEST=.NOT.TEST
      GO TO 10
  105 CONVEX=.NOT.CONVEX
      GO TO 10
  110 WRITE(*,'(A)')'ENTER EXTRA POINTS TO BE PLACED ON WING TIP (EVEN I
     +NTEGER)'
      READ(*,*) IPM
      GO TO 10
  115 ORTHO=.NOT.ORTHO
      GO TO 10
  120 WRITE(*,'(A)')'ENTER POINT LOCATION WHERE TANH IS TO BE CENTERED'
      READ(*,*) JCENT
      GO TO 10
      END
      SUBROUTINE SHOCK
      COMMON/SET/DIMEN,RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,P,Q,IPLUS,MIN
     +OR,MAJOR,CENTRA,ITERAT,TEST,CONVEX,IPN,ORTHO,JCENT
      COMMON/BETA/BETA,PI
      REAL MAJOR,MINOR
      GAMMA=1.4
C
C INITIAL GUESS AT THE SHOCK ANGLE
C
      BETA=ASIN(((GAMMA+1.)*RMACH**2*DELTA*PI/(360.*(RMACH**2-1.)**.5)+1
     +.)/RMACH**2)
      DELTAR=DELTA*PI/180.
C
C ITERATE TO GET SHOCK ANGLE
C
      DO 5 I=1,15
    5 BETA=ASIN(((GAMMA+1.)*SIN(BETA)*SIN(DELTAR)/(2*COS(BETA-DELTAR))+1
     +./RMACH**2)**.5)
      BETA=BETA*180./PI
      RETURN
      END
      SUBROUTINE TRIDIG(MAX)
      COMMON/TRI/A(100),C(100),D(100)
      COMMON ANS(100)
C
C USED MODIFIED THOMAS ALGORITHM FOR TRIDIAGONAL SOLVER
C
      DO 5 I=2,MAX
      AC=1.0-A(I)*C(I-1)
      D(I)=(D(I)-A(I)*D(I-1))/AC
    5 C(I)=C(I)/AC
      DO 10 II=2,MAX
      I=MAX-II+2
   10 ANS(I)=D(I)-C(I)*ANS(I+1)
      RETURN
      END
      SUBROUTINE GEOM(PHI,RB,RFS,RE,SWITCH)
      COMMON/SET/DIMEN,RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,P,Q,IPLUS,MIN
     +OR,MAJOR,CENTRA,ITERAT,TEST,CONVEX,IPM,ORTHO,JCENT
      COMMON/BETA/BETA,PI
      REAL MAJOR,MINOR
      LOGICAL SWITCH
      RPHI=(PHI+90.)*PI/180.
      IF(SWITCH) GO TO 5
      SIGMA=BETA/DELTA
      ANGLE=PHI/PHIL
C
C RADII FOR LOWER BODY, UPPER BODY AND ELLIPSE GEOMETRY
C
      RB=((SIGMA**2-DELTA1**2)*ANGLE**N+DELTA1**2)**.5
      RFS=SIGMA*((SIGMA**2-DELTA1**2)*ANGLE**N/(SIGMA**2-1.)+(DELTA1**2-
     +1.)/(SIGMA**2-1.))**.5
    5 RE=(MAJOR**2*MINOR**2/(MAJOR**2*SIN(RPHI)**2+MINOR**2*COS(RPHI)**2
     +))**.5
      RETURN
      END
      SUBROUTINE ARCLEN
      COMMON/SET/RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,KMAX,P,Q,R,IPLUS,KP
     +LUS,MINOR,MAJOR,CENTRA,ITERAT,IPM,JCENT
      COMMON/ARC/APHI(0:180),ARB(0:180),ARFS(0:180),ARCB(0:180),ARCFS(0:
     +180),ARE(0:180),ARCE(0:180)
      COMMON/BETA/BETA,PI,RADIAN
      COMMON/LENGTH/BARC,FSARC,EARC
      REAL MAJOR,MINOR
      LOGICAL SWITCH
      SWITCH=.FALSE.
C
C LOAD TABLE OF ANGLE, RADII AND ARCLENGTH FOR LOWER, UPPER BODY AND ELLIPSE
C GEOMETRY
C
      PHI=0.0
      CALL GEOM(PHI,RB,RFS,RE,SWITCH)
      APHI(0)=0.0
      ARB(0)=RB
      ARFS(0)=RFS
      ARCB(0)=0.0
      ARCFS(0)=0.0
      ARE(0)=RE
      ARCE(0)=0.0
      IPHIL=PHIL
C
C LOAD TABLE FROM 0 TO DIHEDRAL ANGLE, THIS CREATES THE WAVERIDER GEOMETRY
C AND PART OF OUTER BOUNDARY ELLIPSE
C
      DO 5 I=1,IPHIL
      PHI=I
      CALL GEOM(PHI,RB,RFS,RE,SWITCH)
      APHI(I)=PHI
      ARB(I)=RB
      ARFS(I)=RFS
      ARE(I)=RE
      ANGLE1=APHI(I)*RADIAN
      ANGLE2=APHI(I-1)*RADIAN
      ARCB(I)=((ARB(I)*COS(ANGLE1)-ARB(I-1)*COS(ANGLE2))**2+(ARB(I)*SIN(
     +ANGLE1)-ARB(I-1)*SIN(ANGLE2))**2)**.5+ARCB(I-1)
      ARCFS(I)=((ARFS(I)*COS(ANGLE1)-ARFS(I-1)*COS(ANGLE2))**2+(ARFS(I)*
     +SIN(ANGLE1)-ARFS(I-1)*SIN(ANGLE2))**2)**.5+ARCFS(I-1)
    5 ARCE(I)=((ARE(I)*COS(ANGLE1)-ARE(I-1)*COS(ANGLE2))**2+(ARE(I)*SIN(
     +ANGLE1)-ARE(I-1)*SIN(ANGLE2))**2)**.5+ARCE(I-1)
      BARC=ARCB(IPHIL)
      FSARC=ARCFS(IPHIL)
      SWITCH=.TRUE.
C
C LOAD ELLIPSE VALUES FROM DIHEDRAL ANGLE TO 180 DEGREES
C
      DO 10 I=IPHIL+1,180
      PHI=I
      CALL GEOM(PHI,RB,RFS,RE,SWITCH)
      APHI(I)=PHI
      ARE(I)=RE
      ANGLE1=APHI(I)*RADIAN
      ANGLE2=APHI(I-1)*RADIAN
   10 ARCE(I)=((ARE(I)*COS(ANGLE1)-ARE(I-1)*COS(ANGLE2))**2+(ARE(I)*SIN(
     +ANGLE1)-ARE(I-1)*SIN(ANGLE2))**2)**.5+ARCE(I-1)
      EARC=ARCE(180)
C
C IF THE DIHEDRAL ANGLE IS NOT AN INTEGER FINISH LOADING TABLE FOR WAVERIDER
C WING TIP POINT
C
      IF(IPHIL.EQ.PHIL) THEN
       RETURN
      ELSE
       CALL GEOM(PHIL,RB,RFS,RE,SWITCH)
       I=IPHIL
       APHI(I+1)=PHIL
       ARB(I+1)=RB
       ARFS(I+1)=RFS
       ARE(I+1)=RE
       ANGLE1=APHI(I+1)*RADIAN
       ANGLE2=APHI(I)*RADIAN
       ARCB(I+1)=((ARB(I+1)*COS(ANGLE1)-ARB(I)*COS(ANGLE2))**2+(ARB(I+1)
     + *SIN(ANGLE1)-ARB(I)*SIN(ANGLE2))**2)**.5+ARCB(I)
       ARCFS(I+1)=((ARFS(I+1)*COS(ANGLE1)-ARFS(I)*COS(ANGLE2))**2+(ARFS(
     + I+1)*SIN(ANGLE1)-ARFS(I)*SIN(ANGLE2))**2)**.5+ARCFS(I)
       BARC=ARCB(I+1)
       FSARC=ARCFS(I+1)
       RETURN
      ENDIF
      END
      SUBROUTINE INTERP(MAX,I)
      COMMON/SET/RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,KMAX,P,Q,R,IPLUS,KP
     +LUS,MINOR,MAJOR,CENTRA,ITERAT,IPM,JCENT
      COMMON ANS(100)
      COMMON/ARC/APHI(0:180),ARB(0:180),ARFS(0:180),ARCB(0:180),ARCFS(0:
     +180),ARE(0:180),ARCE(0:180)
      COMMON/SBOUND/XYZS1(100,2),XYZS2(100,2)
      COMMON/BETA/BETA,PI,RADIAN
      COMMON/LENGTH/BARC,FSARC,EARC
      K=0
      GO TO (35,5,20),I
    5 ITAB=MAX+IPM/2
C
C IF USING REFINED POINTS, CALCULATE THOSE POINTS NEAR THE WING
C TIP
C
      IF(IPM .GT. 0) THEN
       MMAX=MAX+IPM/2
       ANS(MMAX)=ANS(MMAX-IPM/2)
       DO 6 J=1,IPM/2
    6  ANS(MMAX-J)=BARC-J/10000.
      ENDIF
C
C INTERPOLATE TO FIND VALUES OF LOWER BODY THAT CORRESPOND TO THE ARCLENGTH
C DISTRIBUTION
C
      DO 15 J=1,MMAX
   10 IF(ARCB(K) .LE. ANS(J) .AND. ARCB(K+1) .GE. ANS(J)) THEN
       FACTOR=ABS((ANS(J)-ARCB(K))/(ARCB(K+1)-ARCB(K)))
       RR=FACTOR*ABS(ARB(K+1)-ARB(K))+ARB(K)
       PHI=FACTOR*ABS(APHI(K+1)-APHI(K))+APHI(K)
       XYZS1(J,1)=RR*COS(PHI*RADIAN)
       XYZS1(J,2)=RR*SIN(PHI*RADIAN)
      ELSE
       K=K+1
       GO TO 10
      ENDIF
   15 CONTINUE
      RETURN
C
C AGAIN CALCULATE REFINED POINTS ON UPPER SURFACE NEAR WING TIP
C
   20 IF(IPM .GT. 0) THEN
       MMAX=MAX+IPM/2
       ANS(MMAX)=ANS(MMAX-IPM/2)
       DO 21 J=1,IPM/2
   21  ANS(MMAX-J)=FSARC-J/10000.
      ENDIF
C
C INTERPOLATE TO FIND VALUES OF UPPER BODY THAT CORRESPOND TO THE ARCLENGTH
C DISTRIBUTION
C
      DO 30 J=1,MMAX
   25 IF(ARCFS(K) .LE. ANS(J) .AND. ARCFS(K+1) .GE. ANS(J)) THEN
       FACTOR=ABS((ANS(J)-ARCFS(K))/(ARCFS(K+1)-ARCFS(K)))
       RR=FACTOR*ABS(ARFS(K+1)-ARFS(K))+ARFS(K)
       PHI=FACTOR*ABS(APHI(K+1)-APHI(K))+APHI(K)
C
C LOAD POINTS BELOW LOWER BODY POINTS
C
       XYZS1(ITAB+MMAX-J,1)=RR*COS(PHI*RADIAN)
       XYZS1(ITAB+MMAX-J,2)=RR*SIN(PHI*RADIAN)
      ELSE
       K=K+1
       GO TO 25
      ENDIF
   30 CONTINUE
      RETURN
   35 DO 45 J=1,MAX
C
C INTERPOLATE TO FIND VALUES ON ELLIPSE THAT CORRESPOND TO THE ARCLENGTH
C DISTRIBUTION
C
   40 IF(ARCE(K) .LE. ANS(J) .AND. ARCE(K+1) .GE. ANS(J)) THEN
       FACTOR=ABS((ANS(J)-ARCE(K))/(ARCE(K+1)-ARCE(K)))
       RR=FACTOR*ABS(ARE(K+1)-ARE(K))+ARE(K)
       PHI=FACTOR*ABS(APHI(K+1)-APHI(K))+APHI(K)
       XYZS2(J,1)=RR*COS(PHI*RADIAN)
       XYZS2(J,2)=RR*SIN(PHI*RADIAN)
      ELSE
       K=K+1
       GO TO 40
      ENDIF
   45 CONTINUE
      RETURN
      END
```

Appendix C Three-Dimensional Wrapped Grid Computer Code

Subroutines SHOCK, TRIDIG, GEOM, and ARCLEN are the same as those found in Appendix B.

```
      PROGRAM GRID3D
      COMMON/SET/RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,KMAX,P,Q,R,IPLUS,KP
     +LUS,MINOR,MAJOR,CENTRA,ITERAT,IPM,JCENT
      COMMON/TRI/A(100),C(100),D(100)
      COMMON/BETA/BETA,PI,RADIAN
      COMMON/BOUND/XYZ1(100,25,3),XYZ2(100,25,3)
      COMMON ANS(100)
      COMMON/SBOUND/XYZS1(100,2),XYZS2(100,2)
      COMMON/ARC/APHI(0:180),ARB(0:180),ARFS(0:180),ARCB(0:180),ARCFS(0:
     +180),ARE(0:180),ARCE(0:180)
      COMMON/LENGTH/BARC,FSARC,EARC
      DIMENSION A3D(100),C3D(100),D3D(100,3)
      DIMENSION XYZM1(0:100,50,3),XYZ(0:100,50,3),XYZPRO(0:100,50,3)
      DIMENSION NODE(3),ARC(100)
      REAL MAJOR,MINOR,LENGTH(3),INTER1,INTER2,K
      LOGICAL CENTRA,TEST
      REWIND 7
      PI=ACOS(-1.0)
      RADIAN=PI/180.
C
C GET INITIAL VALUES FOR RUN
C
      CALL SETUP
C
C CALCULATE SHOCK ANGLE
C
      CALL SHOCK(BETA)
      IMXM1=IMAX-1
      JMXM1=JMAX-1
      KMXM1=KMAX-1
C
C CALCULATE WHERE NOSE OF WAVERIDER IS LOCATED
C
      XNOSE=0.0
      YNOSE=0.0
      ZNOSE=0.0
C
C CALCULATE LENGTH OF WAVERIDER
C
      RLENG=BETA/(DELTA*TAN(BETA*RADIAN))
C
C Y-COORDINATE OF NOSE IS ZERO, X-COORDINATE OF NOSE IS EQUAL TO THE UPPER
C POINT IN THE BASE PLANE BECAUSE THE UPPER SURFACE OF WAVERIDER IS ALIGNED
C WITH THE FREESTREAM, THE Z-COORDINATE COMES FROM THE CONE EQUATION
C
      IF(DELTA1 .NE. DELTA) THEN
       SLOPE1=1./TAN(BETA*RADIAN)
       XNOSE=BETA/DELTA*((DELTA1**2-1.)/((BETA/DELTA)**2-1.))**.5
       ZNOSE=SLOPE1*XNOSE
      ENDIF
C
C CALCULATE THE Z LOCATIONS OF THE WAVERIDER CROSS-SECTIONS
C
```

```
      BB=1.+EXP(-ABS(R))
      AA=-1./BB
      CC=-EXP(-ABS(R))/BB
      A(1)=0.0
      A(KMAX)=0.0
      C(1)=0.0
      C(KMAX)=0.0
      D(1)=ZNOSE
      D(KMAX)=RLENG
      ANS(1)=ZNOSE
      ANS(KMAX)=RLENG
      IF(R .LE. 0.0) THEN
       DO 5 I=2,KMXM1
       A(I)=AA
       C(I)=CC
    5  D(I)=0.0
      ELSE
       DO 10 I=2,KMXM1
       A(I)=CC
       C(I)=AA
   10  D(I)=0.0
      ENDIF
      CALL TRIDIG(KMXM1)
      ANS(KMAX+1)=2*ANS(KMAX)-ANS(KMAX-1)
C
C LOAD Z-COORDINATES INTO THE INNER AND OUTER BOUNDARY POINTS
C
      DO 20 J=1,KMAX+1
      DO 15 I=1,IMAX
      XYZ1(I,J,3)=ANS(J)
   15 XYZ2(I,J,3)=ANS(J)
   20 CONTINUE
C
C STORE Z CROSS-SECTION LOCATIONS
C
      DO 25 I=1,KMAX+1
   25 ARC(I)=ANS(I)
C
C THE BOUNDARY POINTS OF THE INNER AND OUTER BOUNDARY POINTS AT THE FIRST
C CROSS SECTION IS THE NOSE OF THE WAVERIDER WHERE IT INTERSECTS THE SHOCK
C WAVE
C
      DO 30 J=1,IMAX
      XYZ1(J,1,1)=XNOSE
      XYZ1(J,1,2)=YNOSE
      XYZ2(J,1,1)=XNOSE
   30 XYZ2(J,1,2)=YNOSE
C
C CALCULATE WAVERIDER, ELLIPSE BOUNDARY POINTS
C
      CALL ARCLEN
C
C CALCULATE ARCLENGTH OF OUTER BOUNDARY ELLIPSE, AND LOWER AND UPPER SURFACES
```

```
C OF WAVERIDER   EARC, ARCLENGTH OF ELLIPSE   BARC, ARCLENGTH OF LOWER SURFACE
C FSARC, ARCLENGTH OF FREESTREAM OR UPPER SURFACE OF WAVERIDER
C
      I=0
      LENGTH(1)=EARC
      LENGTH(2)=BARC
      LENGTH(3)=FSARC
      NODE(1)=IMAX
   35 I=I+1
C
C GENERATE COEFFICIENTS FOR 1-D ELLIPTIC EQUATION
C
      BB=1.+EXP(-ABS(P))
      AA=-1./BB
      CC=-EXP(-ABS(P))/BB
      A(1)=0.0
      A(NODE(I))=0.0
      C(1)=0.0
      C(NODE(I))=0.0
      D(1)=0.0
      ANS(1)=0.0
      D(NODE(I))=LENGTH(I)
      ANS(NODE(I))=LENGTH(I)
      IF(P .LE. 0.0) THEN
       DO 40 L=2,NODE(I)-1
       A(L)=AA
       C(L)=CC
   40  D(L)=0.0
      ELSE
       DO 45 L=2,NODE(I)-1
       A(L)=CC
       C(L)=AA
   45  D(L)=0.0
      ENDIF
C
C SOLVE 1-D ELLIPTIC EQUATION FOR DISTRIBUTION P AND FOR NODE(I) POINTS.
C
      CALL TRIDIG(NODE(I)-1)
C
C NOW THAT THE ARCLENGTH DISTRIBUTION IS KNOWN, CONVERT ARCLENGTH INTO ELLIPSE
C AND WAVERIDER BOUNDARY POINTS.
C
      CALL INTERP(NODE(I),I,XNOSE)
C
C CALCULATE THE INTERSECTION OF LINE BISECTING UPPER AND LOWER SURFACES OF
C WAVERIDER AND THE OUTER BOUNDARY.
C
      P=0.0
      IF(I .EQ. 1) THEN
       SLIMIT=10000.
C
C GO 5 DEGREES BACK FROM THE WING TIP AT PHIL AND GET X3,Y3 ON LOWER BODY
C SURFACE AND X4,Y4 ON UPPER BODY SURFACE.  AVERAGE X3,Y3 AND X4,Y4 TO GET
```

```
C XMID,YMID POINT
C
      IPHIL=PHIL
      IPOS=IPHIL-5
      ANGLE=FLOAT(IPOS)*RADIAN
      X3=ARB(IPOS)*COS(ANGLE)
      Y3=ARB(IPOS)*SIN(ANGLE)
      X4=ARFS(IPOS)*COS(ANGLE)
      Y4=ARFS(IPOS)*SIN(ANGLE)
C
C CALCULATE POINT ON WING TIP
C
      XMID=(X3+X4)/2.
      YMID=(Y3+Y4)/2.
      IF(IPHIL .EQ. PHIL) THEN
       ANGLE=FLOAT(IPHIL)*RADIAN
       XEND=ARB(IPHIL)*COS(ANGLE)
       YEND=ARB(IPHIL)*SIN(ANGLE)
      ELSE
       ANGLE=PHIL*RADIAN
       XEND=ARB(IPHIL+1)*COS(ANGLE)
       YEND=ARB(IPHIL+1)*SIN(ANGLE)
      ENDIF
C
C CALCULATE SLOPE AND Y-INTERSECT OF LINE BISECTING WAVERIDER GEOMETRY.
C THERE ARE TWO SPECIAL CASES WHEN THE SLOPE IS EQUAL TO ZERO OR INFINITY.
C
      IF(XEND-XMID .EQ. 0.0) THEN
       XCROSS=0.0
       YCROSS=MAJOR
      ELSE
       SLOPE1=(YEND-YMID)/(XEND-XMID)
       IF(SLOPE1 .EQ. 0.0) THEN
        XCROSS=MINOR
        YCROSS=0.0
       ELSE
        INTER1=YMID-SLOPE1*XMID
C
C CALCULATE INTERSECTION POINT OF BISECTING LINE AND OUTER BOUNDARY
C ONLY THE POSITIVE RADICAL TERM OF THE QUADRATIC EQUATION IS NEEDED
C
        XCROSS=(2.*(MAJOR**2*XNOSE-MINOR**2*SLOPE1*INTER1)+(4.*(MINOR**
     +  2*SLOPE1*INTER1-MAJOR**2*XNOSE)**2-4.*(MAJOR**2+MINOR**2*SLOPE1
     +  **2)*(MAJOR**2*XNOSE**2+MINOR**2*INTER1**2-MAJOR**2*MINOR**2))*
     +  *.5)/(2.*(MAJOR**2+MINOR**2*SLOPE1**2))
        YCROSS=SLOPE1*XCROSS+INTER1
       ENDIF
      ENDIF
C
C DETERMINE WHICH POINT ON THE OUTER BOUNDARY IS CLOSEST TO THE INTERSECTION
C POINT
C
      DO 50 L=1,IMAX
```

```
       XDELTA=ABS(XYZS2(L,1)-XCROSS)
       YDELTA=ABS(XYZS2(L,2)-YCROSS)
       DISTAN=(XDELTA**2+YDELTA**2)**.5
       IF(DISTAN .LE. SLIMIT) THEN
        SLIMIT=DISTAN
        NODE(2)=L
       ENDIF
   50  CONTINUE
C
C NODE(2) IS THE NUMBER OF POINTS TO BE PLACED ON LOWER SURFACE.  IF REFINED
C POINTS ARE USED, SUBTRACT HALF THE ADDED POINTS FROM THE LOWER SURFACE TO
C GET THE PROPER DISTRIBUTION OF POINTS ON THAT SURFACE.
C
       NODE(2)=NODE(2)-IPM/2
       NODE(3)=IMAX-NODE(2)+1-IPM
      ENDIF
C
C INITIALIZE INNER BOUNDARY POINT ON OTHER SIDE OF PLANE OF SYMMETRY, TO BE USED
C IN ORTHOGONALITY GENERATION.
C
      IF(I .LT. 3) GO TO 35
C
C CALCULATE EQUATION OF LINE FROM NOSE OF WAVERIDER TO THE WING TIP IN
C THE BASE PLANE
C
      SLOPE1=(RLENG-ZNOSE)/(BETA/DELTA*COS(PHIL*RADIAN)-XNOSE)
      INTER1=RLENG-SLOPE1*BETA/DELTA*COS(PHIL*RADIAN)
      DO 56 I=2,KMAX+1
C
C CALCULATE THE X AND Y LOCATIONS WHERE THE WING TIP OF THE CROSS SECTION
C SHOULD INTERSECT THE WAVERIDER AND GENERATE SCALING FACTORS SO THE BASE
C PLANE GEOMETRY CAN BE SCALED ACCORDINGLY
C
      XSHOCK=(ARC(I)-INTER1)/SLOPE1
      FACT1=(XSHOCK-XNOSE)/(BETA/DELTA*COS(PHIL*RADIAN)-XNOSE)
      YSHOCK=BETA/DELTA*((ARC(I)/RLENG)**2-(XSHOCK*DELTA/BETA)**2)**.5
      FACT2=YSHOCK/YEND
      FACT4=FACT2
      DO 55 J=1,IMAX
C
C SCALE THE INNER AND OUTER BOUNDARIES OF THE BASE PLANE GEOMETRY WITH RESPECT
C TO THE SCALE FACTORS JUST CALCULATED
C
      XYZ1(J,I,1)=(XYZS1(J,1)-XNOSE)*FACT1+XNOSE
      XYZ1(J,I,2)=XYZS1(J,2)*FACT2
      XYZ2(J,I,1)=(XYZS2(J,1)-XNOSE)*FACT4+XNOSE
   55 XYZ2(J,I,2)=XYZS2(J,2)*FACT4
      XYZ1(IMAX+1,I,1)=XYZ1(IMAX-1,I,1)
      XYZ1(IMAX+1,I,2)=-XYZ1(IMAX-1,I,2)
   56 XYZ1(IMAX+1,I,3)=XYZ1(IMAX-1,I,3)
C
C GENERATE COEFFICIENTS FOR 1-D ELLIPTIC EQUATION
C
```

```
      CALL SECOND(CPI)
C
C CALCULATE Q DISTRIBUTION OF POINTS AND STORE
C
      A(1)=0.0
      A(JMAX)=0.0
      C(1)=0.0
      C(JMAX)=0.0
      D(1)=0.0
      ANS(1)=D(1)
      D(JMAX)=10.0
      ANS(JMAX)=D(JMAX)
      BB=1.+EXP(-ABS(Q))
      AA=-1./BB
      CC=-EXP(-ABS(Q))/BB
      IF(Q .LE. 0.0) THEN
       DO 60 J=2,JMXM1
       A(J)=AA
       C(J)=CC
   60  D(J)=0.0
      ELSE
       DO 65 J=2,JMXM1
       A(J)=CC
       C(J)=AA
   65  D(J)=0.0
      ENDIF
C
C SOLVE 1-D ELLIPTIC EQUATION
C
      CALL TRIDIG(JMXM1)
C
C SINCE Q IS CONSTANT THROUGHOUT GRID, SOLVE 1-D ELLIPTIC EQUATION ONCE AND
C STORE IN ARC AND THEN CAN SCALE FOR BOTH INITIAL GUESS AT I AND AT IPLUS, THE
C PROJECTED SOLUTION
C
      DO 70 J=1,JMAX
   70 ARC(J)=ANS(J)
C
C INITIALIZE AQ, EQ, AND K
C
      AQ=ABS(Q)
      IF(AQ .LT. 1.E-6) THEN
       EQ=1.-AQ+AQ**2/2.-AQ**3/6.+AQ**4/24.-AQ**5/120.
       K=1./(1.-AQ/2.+AQ**2/6.-AQ**3/24.+AQ**4/120.)
      ELSE
       EQ=EXP(-AQ)
       K=AQ/(1-EQ)
      ENDIF
C
C BEGINNING OF 3-D GRID GENERATION LOOP, INCREMENT COUNTER, JPLUS IS THE
C LOCATION OF PROJECTED SOLUTION IN THE XI DIRECTION, JPLSM1 IS IPLUS-I, LPLUS
C IS THE LOCATION OF THE PROJECTED SOLUTION IS THE ZETA DIRECTION, AND LPLSM1
C IS KPLUS-K
```

```
C
      KCOUNT=0
   75 KCOUNT=KCOUNT+1
C
C CALCULATE POSITON OF PROJECTED SOLUTION IN THE ZETA DIRECTION
C
      LPLUS=KCOUNT+KPLUS
      LPLSM1=KPLUS
      IF(LPLUS .GT. KMAX) THEN
       LPLUS=KMAX
       LPLSM1=LPLUS-KCOUNT
      ENDIF
      IF(KCOUNT .EQ. KMAX) THEN
       LPLUS=KMAX+1
       LPLSM1=1
      ENDIF
      IF(KCOUNT .EQ. 1) THEN
C
C LOAD NOSE POINTS INTO THE MINUS ONE 2-D CROSSECTION
C
       DO 85 I=0,IMAX+1
       DO 80 J=1,JMAX
       XYZM1(I,J,1)=XYZ1(1,1,1)
       XYZM1(I,J,2)=XYZ1(1,1,2)
       XYZM1(I,J,3)=XYZ1(1,1,3)
C
C THE INITIAL PLANE IS A POINT, SO WRITE IT OUT INTO THE OUTPUT FILE
C
       IF(I .GT. 0 .AND. I .LT. IMAX+1) THEN
        WRITE(7,'(3(F13.8,1X))')XYZM1(I,J,1),XYZM1(I,J,2),XYZM1(I,J,3)
       ENDIF
   80  CONTINUE
   85  CONTINUE
       GO TO 75
      ENDIF
C
C IF KPLUS IS EQUAL TO ONE AND AT SECOND MARCHING POSITION ONLY THE NEW
C PROJECTED PLANE NEEDS TO BE FILLED WITH NEARLY ORTHOGONAL CURVES.  THE
C NEW SOLUTION PLANE IS THE OLD PROJECTED SOLUTION PLANE AND THE NEARLY
C ORTHOGONAL CURVES CAN BE ROTATED IN
C
      IF(KCOUNT .GT. 2 .AND. KPLUS .EQ. 1) THEN
       DO 95 I=0,IMAX+1
       DO 90 J=1,JMAX
       XYZ(I,J,1)=XYZPRO(I,J,1)
       XYZ(I,J,2)=XYZPRO(I,J,2)
   90  XYZ(I,J,3)=XYZPRO(I,J,3)
   95  CONTINUE
       JCOUNT=1
       GO TO 106
      ENDIF
C
C IF AT THE FIRST SOLUTION STATION OR IF KPLUS IS GREATER THAN ONE, BOTH THE
```

```
C SOLUTION AND PROJECTED SOLUTION PLANES NEED TO BE FILLED WITH NEARLY
C ORTHOGONAL CURVES.  THE STORED 1-D Q DISTRIBUTION OF POINTS IS SCALED
C BETWEEN ALL THE BOUNDARY POINTS IN THESE TWO PLANES
C
      JCOUNT=0
C
C THIS PROCEDURE CALCULATES THE NEARLY ORTHOGONAL CURVES IN THE SOLUTION PLANE
C
      DO 105 I=1,IMAX
      XYZ(I,1,1)=XYZ1(I,KCOUNT,1)
      XYZ(I,1,2)=XYZ1(I,KCOUNT,2)
      XYZ(I,1,3)=XYZ1(I,KCOUNT,3)
      XYZ(I,JMAX,1)=XYZ2(I,KCOUNT,1)
      XYZ(I,JMAX,2)=XYZ2(I,KCOUNT,2)
      XYZ(I,JMAX,3)=XYZ2(I,KCOUNT,3)
      FACT1=(XYZ(I,JMAX,1)-XYZ(I,1,1))/10.
      FACT2=(XYZ(I,JMAX,2)-XYZ(I,1,2))/10.
      FACT3=(XYZ(I,JMAX,3)-XYZ(I,1,3))/10.
      DO 100 J=2,JMXM1
      DIFF=ARC(J)-ARC(J-1)
      XYZ(I,J,1)=XYZ(I,J-1,1)+DIFF*FACT1
      XYZ(I,J,2)=XYZ(I,J-1,2)+DIFF*FACT2
  100 XYZ(I,J,3)=XYZ(I,J-1,3)+DIFF*FACT3
  105 CONTINUE
C
C THIS PROCEDURE GENERATES THE NEARLY ORTHOGONAL CURVES IN THE PROJECTED
C SOLUTION PLANE
C
  106 DO 115 I=1,IMAX
      XYZPRO(I,1,1)=XYZ1(I,LPLUS,1)
      XYZPRO(I,1,2)=XYZ1(I,LPLUS,2)
      XYZPRO(I,1,3)=XYZ1(I,LPLUS,3)
      XYZPRO(I,JMAX,1)=XYZ2(I,LPLUS,1)
      XYZPRO(I,JMAX,2)=XYZ2(I,LPLUS,2)
      XYZPRO(I,JMAX,3)=XYZ2(I,LPLUS,3)
      FACT1=(XYZPRO(I,JMAX,1)-XYZPRO(I,1,1))/10.
      FACT2=(XYZPRO(I,JMAX,2)-XYZPRO(I,1,2))/10.
      FACT3=(XYZPRO(I,JMAX,3)-XYZPRO(I,1,3))/10.
      DO 110 J=2,JMXM1
      DIFF=ARC(J)-ARC(J-1)
      XYZPRO(I,J,1)=XYZPRO(I,J-1,1)+DIFF*FACT1
      XYZPRO(I,J,2)=XYZPRO(I,J-1,2)+DIFF*FACT2
  110 XYZPRO(I,J,3)=XYZPRO(I,J-1,3)+DIFF*FACT3
  115 CONTINUE
C
C THE FOLLOWING DETERMINES WHICH CROSS SECTION IN THE Z DIRECTION IS BEING USED
C
  120 JCOUNT=JCOUNT+1
      IF(JCOUNT .EQ. 1) THEN
       KPOS=KCOUNT
      ELSE
       KPOS=LPLUS
      ENDIF
```

```
C
C CALCULATE THETA2 OF THE NODE POINTS BEFORE AND AFTER THE REFINED POINTS
C FOR USE IN NEARLY ORTHOGONAL CURVE GENERATION AND CALCULATE THE INTERVAL
C TO BE ADDED TO EACH REFINED POINT
C
      TEST=.TRUE.
      LCOUNT=0
  121 LCOUNT=LCOUNT+1
      IF(LCOUNT .EQ. 3) THEN
       TEST=.FALSE.
       IF(ANG1 .GT. PI) ANG1=ANG1-2.*PI
       ANG3=(ANG2-ANG1)/FLOAT(IPM)
       GO TO 129
      ENDIF
      IF(LCOUNT .EQ. 1) THEN
       ICOUNT=NODE(2)-1
      ELSE
       ICOUNT=NODE(2)+IPM+1
      ENDIF
      GO TO 133
  129 ICOUNT=0
  130 ICOUNT=ICOUNT+1
C
C IF ORTHOGONALITY SWITCH IS ON, TURN THE PROJECTED SOLUTION FROM A STRAIGHT
C TO A CURVE THAT STARTS ORTHOGONAL TO SURFACE AND ENDS UP AT OUTER BOUNDARY
C SURFACE
C
      IF(ICOUNT .EQ. 1 .OR. ICOUNT .EQ. IMAX) THEN
       THETA1=0.0
       THETA2=0.0
       ATH1=1.
       GO TO 131
      ENDIF
C
C CALCULATE ANGLE OF LINE BETWEEN INNER AND OUTER BOUNDARIES
C
       THETA1=ATAN((XYZ2(ICOUNT,KPOS,2)-XYZ1(ICOUNT,KPOS,2))/(XYZ2(ICOUN
     + T,KPOS,1)-XYZ1(ICOUNT,KPOS,1)))
C
C PUT ANGLE IN PROPER QUADRAT, HAVE SPECIAL CONSIDERATIONS WHEN SLOPES ARE
C EITHER ZERO OR INFINITY
C
       IF(XYZ2(ICOUNT,KPOS,1)-XYZ1(ICOUNT,KPOS,1) .EQ. 0.0) THEN
        FACT1=1.
       ELSE
        FACT1=(XYZ2(ICOUNT,KPOS,1)-XYZ1(ICOUNT,KPOS,1))/ABS(XYZ2(ICOUNT,
     +  KPOS,1)-XYZ1(ICOUNT,KPOS,1))
        ENDIF
        IF(XYZ2(ICOUNT,KPOS,2)-XYZ1(ICOUNT,KPOS,2) .EQ. 0.0) THEN
         FACT2=1.
        ELSE
         FACT2=(XYZ2(ICOUNT,KPOS,2)-XYZ1(ICOUNT,KPOS,2))/ABS(XYZ2(ICOUNT
     +   ,KPOS,2)-XYZ1(ICOUNT,KPOS,2))
```

```
       ENDIF
C
C SINCE QUADRAT NOW KNOWN, CHANGE ANGLE TO REFLECT QUADRAT
C
      IF(FACT1 .EQ. -1. .AND. FACT2 .EQ. 1.) THETA1=PI+THETA1
      IF(FACT1 .EQ. -1. .AND. FACT2 .EQ. -1.) THETA1=PI+THETA1
      IF(FACT1 .EQ. 1. .AND. FACT2 .EQ. -1.) THETA1=2.*PI+THETA1
C
C CALCULATE ANGLE OF NORMAL LINE TO BODY SURFACE USING AN AVERAGE OF PANELS ON
C EACH SIDE OF THE PROJECTED SOLUTION
C
      IF(ICOUNT .GT. NODE(2)-1 .AND. ICOUNT .LT. NODE(2)+IPM+1) THEN
       THETA2=ANG1+ANG3*FLOAT(ICOUNT-NODE(2))
       GO TO 132
      ENDIF
  133 THETA2=(ATAN((XYZ1(ICOUNT-1,KPOS,1)-XYZ1(ICOUNT,KPOS,1))/(XYZ1(
     + ICOUNT,KPOS,2)-XYZ1(ICOUNT-1,KPOS,2)))+ATAN((XYZ1(ICOUNT,KPOS,1)-
     + XYZ1(ICOUNT+1,KPOS,1))/(XYZ1(ICOUNT+1,KPOS,2)-XYZ1(ICOUNT,KPOS,2)
     + )))/2.
      SLOPE1=TAN(THETA2)
C
C HAVE A QUADRAT PROBLEM AGAIN.  HAVE SLOPE AND POINT AND THEREFORE EQUATION
C OF LINE.  INTERSECT THIS LINE WITH OUTER BOUNDARY WITH BOTH POSITIVE AND
C NEGATIVE SIGNS ON THE RADICAL OF THE QUADRATIC EQUATION.  SPECIAL PROBLEMS
C AGAIN WITH SLOPES EQUAL TO ZERO AND INFINITY
C
      FACT5=FACT4*MAJOR
      FACT6=FACT4*MINOR
      IF(SLOPE1 .EQ. 0.0) THEN
       YQUAD1=XYZ1(ICOUNT,KPOS,2)
       YQUAD2=XYZ1(ICOUNT,KPOS,2)
       XQUAD1=((1.-XYZ1(ICOUNT,KPOS,2)**2/FACT5**2)*FACT6**2)**.5
       XQUAD2=-XQUAD1
      ELSE
       IF(SLOPE1 .GT. 10000.) THEN
        XQUAD1=XYZ1(ICOUNT,KPOS,1)
        XQUAD2=XYZ1(ICOUNT,KPOS,1)
        YQUAD1=((1.-XYZ1(ICOUNT,KPOS,1)**2/FACT6**2)*FACT4**2)**.5
        YQUAD2=-YQUAD1
       ELSE
        INTER1=XYZ1(ICOUNT,KPOS,2)-SLOPE1*XYZ1(ICOUNT,KPOS,1)
        FACT1=-2.*FACT6**2*SLOPE1*INTER1+2.*FACT5**2*XNOSE
        FACT2=2.*(FACT5**2+FACT6**2*SLOPE1**2)
        RADICA=((-2.*FACT5**2*XNOSE+2.*FACT6**2*SLOPE1*INTER1)**2-4.*(F
     +  ACT5**2+FACT6**2*SLOPE1**2)*(FACT5**2*XNOSE**2+FACT6**2*INTER1*
     +  *2-FACT5**2*FACT6**2))**.5
        XQUAD1=(FACT1+RADICA)/FACT2
        XQUAD2=(FACT1-RADICA)/FACT2
        YQUAD1=SLOPE1*XQUAD1+INTER1
        YQUAD2=SLOPE1*XQUAD2+INTER1
       ENDIF
      ENDIF
C
```

```
C CALCULATE DISTANCE BETWEEN INTERSECTIONS AND OUTER BOUNDARY POINT.  THE
C SHORTER DISTANCE DETERMINES THE PROPER QUADRAT
C
       QUAD1=((XQUAD1-XYZ2(ICOUNT,KPOS,1))**2+(YQUAD1-XYZ2(ICOUNT,KPOS,2
     + ))**2)**.5
       QUAD2=((XQUAD2-XYZ2(ICOUNT,KPOS,1))**2+(YQUAD2-XYZ2(ICOUNT,KPOS,2
     + ))**2)**.5
       IF(QUAD1 .LT. QUAD2) THEN
        XCROSS=XQUAD1
        YCROSS=YQUAD1
       ELSE
        XCROSS=XQUAD2
        YCROSS=YQUAD2
       ENDIF
       IF(XCROSS-XYZ1(ICOUNT,KPOS,1) .EQ. 0.0) THEN
        FACT1=1.
       ELSE
        FACT1=(XCROSS-XYZ1(ICOUNT,KPOS,1))/ABS(XCROSS-XYZ1(ICOUNT,KPOS,1
     +  ))
       ENDIF
       IF(YCROSS-XYZ1(ICOUNT,KPOS,2) .EQ. 0.0) THEN
        FACT2=1.
       ELSE
        FACT2=(YCROSS-XYZ1(ICOUNT,KPOS,2))/ABS(YCROSS-XYZ1(ICOUNT,KPOS,2
     +  ))
       ENDIF
C
C CHANGE ANGLE TO REFLECT QUADRAT
C
       IF(FACT1 .EQ. -1. .AND. FACT2 .EQ. 1.) THETA2=PI+THETA2
       IF(FACT1 .EQ. -1. .AND. FACT2 .EQ. -1.) THETA2=PI+THETA2
       IF(FACT1 .EQ. 1. .AND. FACT2 .EQ. -1.) THETA2=2.*PI+THETA2
       IF(TEST) THEN
        IF(LCOUNT .EQ. 1) THEN
         ANG1=THETA2
        ELSE
         ANG2=THETA2
        ENDIF
        GO TO 121
       ENDIF
C
C FACTOR USED TO ACCELERATE OR RETARD HYPERBOLIC TANGENT
C
  132  IF(THETA2-THETA1 .EQ. 0.0) THEN
        ATH1=25.
       ELSE
        ATH1=1./ABS(THETA2-THETA1)
        IF(ATH1 .GT. 25.) ATH1=25.
       ENDIF
C
C CALCULATE ANGLE OF LINE BETWEEN INNER AND OUTER BOUNDARIES
C
  131  PSI1=0.0
```

```
C
C CALCULATE ANGLE OF NORMAL LINE TO BODY SURFACE USING AN AVERAGE OF PANELS ON
C EACH SIDE OF THE PROJECTED SOLUTION
C
       PSI2=ATAN((XYZ1(ICOUNT,KPOS-1,1)-XYZ1(ICOUNT,KPOS,1))/(XYZ1(ICOUN
     + T,KPOS,3)-XYZ1(ICOUNT,KPOS-1,3)))
C
C FACTOR USED TO ACCELERATE OR RETARD HYPERBOLIC TANGENT
C
       IF(PSI2-PSI1 .EQ. 0.0) THEN
        ATH2=25.
       ELSE
        ATH2=1./ABS(PSI2-PSI1)
        IF(ATH2 .GT. 25.) ATH2=25.
       ENDIF
C
C LENGTH OF PROJECTED SOLUTION
C
       SPAN=((XYZ2(ICOUNT,KPOS,1)-XYZ1(ICOUNT,KPOS,1))**2+(XYZ2(ICOUNT,K
     + POS,2)-XYZ1(ICOUNT,KPOS,2))**2+(XYZ2(ICOUNT,KPOS,3)-XYZ1(ICOUNT,K
     + POS,3))**2)**.5
C
C CENTER OF PROJECTED SOLUTION WITH RESPECT TO THE MIDDLE POINT
C
       IF(JCOUNT .EQ. 1) THEN
        CENTER=(((XYZ(ICOUNT,JCENT,1)-XYZ1(ICOUNT,KPOS,1))**2+(XYZ(ICOUN
     +  T,JCENT,2)-XYZ1(ICOUNT,KPOS,2))**2+(XYZ(ICOUNT,JCENT,3)-XYZ1(ICO
     +  UNT,KPOS,3))**2)**.5)/SPAN
       ELSE
        CENTER=(((XYZPRO(ICOUNT,JCENT,1)-XYZ1(ICOUNT,KPOS,1))**2+(XYZPRO
     +  (ICOUNT,JCENT,2)-XYZ1(ICOUNT,KPOS,2))**2+(XYZPRO(ICOUNT,JCENT,3)
     +  -XYZ1(ICOUNT,KPOS,3))**2)**.5)/SPAN
       ENDIF
C
C CHECK TO SEE WHERE THE HYPERBOLIC TANGENT SHOULD BE CENTERED, SO THERE WON'T
C BE GRID CROSS OVER DUE TO THE ORTHOGONALITY.  CALCULATE EQUATION OF
C ORTHOGONAL IPLUS LINE AND AT PRIOR STATION.  CALCULATE INTERSECTION AND USE
C MORE RESTRICTIVE CENTERING CRITERION.
C
       CHANGE=10000.
       IF(ICOUNT .GT. 2 .AND. ICOUNT .LT. NODE(2)+1 .OR. ICOUNT .GT. NOD
     + E(2)+IPN+1 .AND. ICOUNT .LT. IMAX) THEN
        SLOPE1=TAN(THETA4)
        SLOPE2=TAN(THETA2)
        INTER1=XYZ1(ICOUNT-1,KPOS,2)-SLOPE1*XYZ1(ICOUNT-1,KPOS,1)
        INTER2=XYZ1(ICOUNT,KPOS,2)-SLOPE2*XYZ1(ICOUNT,KPOS,1)
        XCROSS=(INTER2-INTER1)/(SLOPE1-SLOPE2)
        YCROSS=SLOPE2*XCROSS+INTER2
        IF(XYZ2(ICOUNT,KPOS,1) .GT. XYZ1(ICOUNT,KPOS,1)) THEN
         IF(XCROSS .GT. XYZ1(ICOUNT,KPOS,1) .AND. XCROSS .LT. XYZ2(ICOUN
     +   T,KPOS,1)) THEN
          CHANGE=((XCROSS-XYZ1(ICOUNT,KPOS,1))**2+(YCROSS-XYZ1(ICOUNT,KP
     +    OS,2))**2)**.5/SPAN-.15/SPAN
```

```
        ENDIF
       ELSE
        IF(XCROSS .LT. XYZ1(ICOUNT,KPOS,1) .AND. XCROSS .GT. XYZ2(ICOUN
     +  T,KPOS,1)) THEN
         CHANGE=((XCROSS-XYZ1(ICOUNT,KPOS,1))**2+(YCROSS-XYZ1(ICOUNT,KP
     +   OS,2))**2)**.5/SPAN-.15/SPAN
        ENDIF
       ENDIF
      ENDIF
      IF(CHANGE .LE. CENTER) CENTER=CHANGE
      THETA4=THETA2
C
C GENERATE THE ORTHOGONAL CURVE IN 3-D
C
      DO 135 I=2,JMXM1
      IF(JCOUNT .EQ. 1) THEN
       DISTAN=((XYZ(ICOUNT,I,1)-XYZ1(ICOUNT,KPOS,1))**2+(XYZ(ICOUNT,I,2
     + )-XYZ1(ICOUNT,KPOS,2))**2+(XYZ(ICOUNT,I,3)-XYZ1(ICOUNT,KPOS,3))*
     + *2)**.5
      ELSE
       DISTAN=((XYZPRO(ICOUNT,I,1)-XYZ1(ICOUNT,KPOS,1))**2+(XYZPRO(ICOU
     + NT,I,2)-XYZ1(ICOUNT,KPOS,2))**2+(XYZPRO(ICOUNT,I,3)-XYZ1(ICOUNT,
     + KPOS,3))**2)**.5
      ENDIF
      ANGLE1=ATH1*(DISTAN/SPAN-CENTER)
      ANGLE2=ATH2*(DISTAN/SPAN-CENTER)
      IF(I .EQ. 2) THEN
       ANGLE3=TANH(ANGLE1)
       ANGLE4=TANH(-ATH1*(1.-CENTER))
       ANGLE5=TANH(ANGLE2)
       ANGLE6=TANH(-ATH2*(1.-CENTER))
      ENDIF
      FACT1=FLOAT(JMXM1-I)*ANGLE3/FLOAT(JMXM1-2)+ANGLE4*((FLOAT(I-JMXM1
     +)/FLOAT(JMXM1-2))+1.)
      FACT2=FLOAT(JMXM1-I)*ANGLE5/FLOAT(JMXM1-2)+ANGLE6*((FLOAT(I-JMXM1
     +)/FLOAT(JMXM1-2))+1.)
C
C ANGLE THAT STARTS OUT AT THE ORTHOGONAL ANGLE AND GOES TO THE STRAIGHT LINE
C ANGLE
C
      IF(ABS(THETA2-THETA1) .GT. PI*.5) THEN
C
C FOR ANGLES THAT ARE IN BOTH FIRST AND FOURTH QUADRAT
C
       THETA3=(TANH(ANGLE1)/FACT1*(THETA2-2.*PI-THETA1)+THETA2-2.*PI+TH
     + ETA1)/2.
      ELSE
       THETA3=(TANH(ANGLE1)/FACT1*(THETA2-THETA1)+THETA2+THETA1)/2.
      ENDIF
C
C ANGLE THAT STARTS OUT AT THE ORTHOGONAL ANGLE AND GOES TO THE STRAIGHT LINE
C ANGLE
C
```

```
       IF(ABS(PSI2-PSI1) .GT. PI*.5) THEN
C
C FOR ANGLES THAT ARE IN BOTH FIRST AND FOURTH QUADRAT
C
        PSI3=(TANH(ANGLE2)/FACT2*(PSI2-2.*PI-PSI1)+PSI2-2.*PI+PSI1)/2.
       ELSE
        PSI3=(TANH(ANGLE2)/FACT2*(PSI2-PSI1)+PSI2+PSI1)/2.
       ENDIF
C
C NEW ORTHOGONAL PROJECTED SOLUTION
C
       IF(JCOUNT .EQ. 1) THEN
        XYZ(ICOUNT,I,1)=DISTAN*COS(THETA3)+XYZ1(ICOUNT,KPOS,1)
        XYZ(ICOUNT,I,2)=DISTAN*SIN(THETA3)+XYZ1(ICOUNT,KPOS,2)
        XYZ(ICOUNT,I,3)=DISTAN*SIN(PSI3)+XYZ1(ICOUNT,KPOS,3)
       ELSE
        XYZPRO(ICOUNT,I,1)=DISTAN*COS(THETA3)+XYZ1(ICOUNT,KPOS
     +  ,1)
        XYZPRO(ICOUNT,I,2)=DISTAN*SIN(THETA3)+XYZ1(ICOUNT,KPOS
     +  ,2)
        XYZPRO(ICOUNT,I,3)=DISTAN*SIN(PSI3)+XYZ1(ICOUNT,KPOS,3)
       ENDIF
  135  CONTINUE
C
C LOAD INITIAL SOLVED SOLUTION ON OTHER SIDE OF THE PLANE OF SYMMETRY
C
      IF(ICOUNT .LT. IMAX-1) GO TO 130
      IF(JCOUNT .EQ. 1) THEN
       DO 140 I=1,JMAX
       XYZ(0,I,1)=XYZ(2,I,1)
       XYZ(0,I,2)=-XYZ(2,I,2)
  140  XYZ(0,I,3)=XYZ(2,I,3)
      ELSE
       DO 145 I=1,JMAX
       XYZPRO(0,I,1)=XYZPRO(2,I,1)
       XYZPRO(0,I,2)=-XYZPRO(2,I,2)
       XYZPRO(0,I,3)=XYZPRO(2,I,3)
       XYZPRO(IMAX+1,I,1)=XYZPRO(IMAX-1,I,1)
       XYZPRO(IMAX+1,I,2)=-XYZPRO(IMAX-1,I,2)
  145  XYZPRO(IMAX+1,I,3)=XYZPRO(IMAX-1,I,3)
      ENDIF
      IF(JCOUNT .EQ. 1) GO TO 120
C
C ITERATION LOOP
C
      DO 185 I=1,ITERAT
      DO 180 J=1,IMAX
      JPLUS=J+IPLUS
      JPLSM1=IPLUS
      IF(J .EQ. 1) THEN
       JPLUS=2
       JPLSM1=1
      ENDIF
      IF(JPLUS .GT. IMAX) THEN
       JPLUS=IMAX
       JPLSM1=JPLUS-J
      ENDIF
      IF(J .EQ. IMAX) THEN
       JPLUS=IMAX+1
       JPLSM1=1
      ENDIF
C
C DO A CHECK ON CONVEXITY IF NOT SOLVING EITHER PLANE OF SYMMETRY AND THE
C CONVEXITY SWITCH IS ON
C
      IF(J .GT. 1 .AND. J .LT. IMAX) THEN
C
C GET EQUATION OF LINE FROM THE SECOND POINT OF PROJECTED SOLUTION TO THE SECOND
C POINT OF THE PRIOR SOLVED SOLUTION
C
  150  SLOPE1=(XYZ(JPLUS,2,2)-XYZ(J-1,2,2))/(XYZ(JPLUS,2,1)-XYZ(J-1,2,1)
     + )
       INTER1=XYZ(J-1,2,2)-SLOPE1*XYZ(J-1,2,1)
C
C CHECK IF BODY INTERSECTS THE LINE JUST CALCULATED.  CALCULATE EQUATION OF LINE
C FOR EACH GRID LINE BETWEEN PRIOR SOLUTION AND PROJECTED SOLUTION AND DETERMINE
C WHETHER THE SURFACE IS TOO CONVEX
C
       IF(JPLSM1 .EQ. 1) GO TO 160
       DO 155 L=JPLUS-1,J,-1
       SLOPE2=(XYZ2(L,KCOUNT,2)-XYZ1(L,KCOUNT,2))/(XYZ2(L,KCOUNT,1)-XYZ1
     + (L,KCOUNT,1))
       INTER2=XYZ1(L,KCOUNT,2)-SLOPE2*XYZ1(L,KCOUNT,1)
       XCROSS=(INTER2-INTER1)/(SLOPE1-SLOPE2)
       IF(XYZ1(L,KCOUNT,1) .LT. XYZ2(L,KCOUNT,1)) THEN
        IF(XCROSS .GT. XYZ1(L,KCOUNT,1) .AND. XCROSS .LT. XYZ2(L,KCOUNT,
     + 1))THEN
        ELSE
C
C SURFACE TOO CONVEX, MOVE PROJECTED SOLUTION BACK ONE
C
         JPLUS=JPLUS-1
         JPLSM1=JPLSM1-1
         IF(JPLUS-J .EQ. 1) GO TO 160
          GO TO 150
        ENDIF
       ELSE
        IF(XCROSS .LT. XYZ1(L,KCOUNT,1) .AND. XCROSS .GT. XYZ2(L,KCOUNT,
     + 1)) THEN
        ELSE
         JPLUS=JPLUS-1
         JPLSM1=JPLSM1-1
         IF(JPLUS-J .EQ. 1) GO TO 160
          GO TO 150
        ENDIF
       ENDIF
  155 CONTINUE
      ENDIF
C
C FIRST TIME THROUGH THE LOOP, IT CALCULATES THE X-COMPONENT. THE SECOND TIME
C THROUGH THE LOOP, IT CALCULATES THE Y-COMPONENT. THE THIRD TIME THROUGH THE
C LOOP IT CALCULATES THE Z-COMPONENT.
C
  160 A3D(1)=0.0
      A3D(JMAX)=0.0
      C3D(1)=0.0
      C3D(JMAX)=0.0
      D3D(1,1)=XYZ(J,1,1)
      D3D(1,2)=XYZ(J,1,2)
      D3D(1,3)=XYZ(J,1,3)
      D3D(JMAX,1)=XYZ(J,JMAX,1)
      D3D(JMAX,2)=XYZ(J,JMAX,2)
      D3D(JMAX,3)=XYZ(J,JMAX,3)
      DO 165 M=2,JMXM1
      MP1=M+1
      MM1=M-1
C
C AT BOTH PLANES OF SYMMETRY, THE METRICS HAVE TO BE CALCULATED WITH
C CENTRAL DIFFERENCE.
C
      IF(J .EQ. 1 .OR. J .EQ. IMAX .OR. CENTRA) THEN
C
C DO CENTRAL DIFFERENCE ON METRICS IF IPLUS IS EQUAL TO 1 AND
C CENTRAL DIFFERENCING SWITCH IS ON.
C
       XXSI=(XYZ(JPLUS,M,1)-XYZ(J-1,M,1))/2.
       YXSI=(XYZ(JPLUS,M,2)-XYZ(J-1,M,2))/2.
       ZXSI=(XYZ(JPLUS,M,3)-XYZ(J-1,M,3))/2.
       XETA=(XYZ(J,MP1,1)-XYZ(J,MM1,1))/2.
       YETA=(XYZ(J,MP1,2)-XYZ(J,MM1,2))/2.
       ZETA=(XYZ(J,MP1,3)-XYZ(J,MM1,3))/2.
       XZETA=(XYZPRO(J,M,1)-XYZM1(J,M,1))/(2.*FLOAT(LPLSM1))
       YZETA=(XYZPRO(J,M,2)-XYZM1(J,M,2))/(2.*FLOAT(LPLSM1))
       ZZETA=(XYZPRO(J,M,3)-XYZM1(J,M,3))/(2.*FLOAT(LPLSM1))
      ELSE
C
C DO FIRST ORDER BACKWARDS DIFFERENCING IF CENTRAL DIFFERENCING
C SWITCH IS OFF.
C
       XXSI=XYZ(J,M,1)-XYZ(J-1,M,1)
       YXSI=XYZ(J,M,2)-XYZ(J-1,M,2)
       ZXSI=XYZ(J,M,3)-XYZ(J-1,M,3)
       XETA=(XYZ(J,MP1,1)-XYZ(J,MM1,1))/2.
       YETA=(XYZ(J,MP1,2)-XYZ(J,MM1,2))/2.
       ZETA=(XYZ(J,MP1,3)-XYZ(J,MM1,3))/2.
       XZETA=XYZ(J,M,1)-XYZM1(J,M,1)
       YZETA=XYZ(J,M,2)-XYZM1(J,M,2)
       ZZETA=XYZ(J,M,3)-XYZM1(J,M,3)
      ENDIF
      C11=YETA*ZZETA-YZETA*ZETA
      C22=XXSI*ZZETA-XZETA*ZXSI
      C33=XXSI*YETA-XETA*YXSI
      C12=-YXSI*ZZETA-YZETA*ZXSI
      C13=YXSI*ZETA-YETA*ZXSI
      C23=-XXSI*ZETA+XETA*ZXSI
      C21=-XETA*ZZETA+XZETA*ZETA
      C31=XETA*YZETA-XZETA*YETA
      C32=-XXSI*YZETA+XZETA*YXSI
      ALPH11=C11**2+C21**2+C31**2
      ALPH12=C11*C12+C21*C22+C31*C32
      ALPH13=C11*C13+C21*C23+C31*C33
      ALPH22=C12**2+C22**2+C32**2
      ALPH23=C12*C13+C22*C23+C32*C33
      ALPH33=C13**2+C23**2+C33**2
      BBB=ALPH11*(1.+ABS(P)+1./FLOAT(JPLSM1))+ALPH33*(1.+ABS(R)+1./FLOAT
     +(LPLSM1))+ALPH22*K*(1.+EQ)
      AAA=-ALPH22*K/BBB
      CCC=-ALPH22*K*EQ/BBB
      IF(Q .LE. 0.0) THEN
       A3D(M)=AAA
       C3D(M)=CCC
      ELSE
       A3D(M)=CCC
       C3D(M)=AAA
      ENDIF
      DO 164 L=1,3
      IF(P .LE. 0.0) THEN
       DDD1=ALPH11*((1.+ABS(P))*XYZ(J-1,M,L)+1./FLOAT(JPLSM1)*XYZ(JPLUS,
     + M,L))
      ELSE
       DDD1=ALPH11*(XYZ(J-1,M,L)+(1.+ABS(P))/FLOAT(JPLSM1)*XYZ(JPLUS,M,L
     + ))
      ENDIF
      IF(R .LE. 0.0) THEN
       DDD2=ALPH33*((1.+ABS(R))*XYZM1(J,M,L)+1./FLOAT(LPLSM1)*XYZPRO(J,M
     + ,L))
      ELSE
       DDD2=ALPH33*(XYZM1(J,M,L)+(1.+ABS(R))/FLOAT(LPLSM1)*XYZPRO(J,M,L)
     + )
      ENDIF
      D3D(M,L)=(DDD1+DDD2+ALPH12/FLOAT(JPLSM1+1)*(XYZ(JPLUS,MP1,L)-XYZ(J
     +-1,MP1,L)-XYZ(JPLUS,MM1,L)+XYZ(J-1,MM1,L))+2.*ALPH13/(FLOAT(JPLSM1
     ++1)*FLOAT(LPLSM1+1))*(XYZPRO(JPLUS,M,L)-XYZM1(JPLUS,M,L)-XYZPRO(J-
     +1,M,L)+XYZM1(J-1,M,L))+ALPH23/FLOAT(LPLSM1+1)*(XYZPRO(J,MP1,L)-XYZ
     +PRO(J,MM1,L)-XYZM1(J,MP1,L)+XYZM1(J,MM1,L)))/BBB
  164 CONTINUE
  165 CONTINUE
      DO 175 L=1,3
      DO 166 M=1,JMAX
      A(M)=A3D(M)
      C(M)=C3D(M)
      D(M)=D3D(M,L)
  166 CONTINUE
      ANS(1)=D(1)
      ANS(JMAX)=D(JMAX)
C
C SOLVE 3-D PARABOLIC EQUATION
C
      CALL TRIDIG(JMXM1)
C
C LOAD SOLUTION INTO X AND Y VECTORS
C
      DO 170 M=2,JMXM1
  170 XYZ(J,M,L)=ANS(M)
  175 CONTINUE
      IF(J .EQ. IMAX-1) THEN
       DO 176 M=1,JMAX
       XYZ(IMAX+1,M,1)=XYZ(IMAX-1,M,1)
       XYZ(IMAX+1,M,2)=-XYZ(IMAX-1,M,2)
  176  XYZ(IMAX+1,M,3)=XYZ(IMAX-1,M,3)
      ENDIF
  180 CONTINUE
  185 CONTINUE
C
C WRITE OUT SOLUTION
C
      DO 200 I=1,IMAX
      DO 195 J=1,JMAX
      DO 190 L=1,3
  190 XYZM1(I,J,L)=XYZ(I,J,L)
      WRITE(7,'(3(F13.8,1X))')XYZ(I,J,1),XYZ(I,J,2),XYZ(I,J,3)
  195 CONTINUE
  200 CONTINUE
      IF(KCOUNT .LT. KMAX) GO TO 75
      CALL SECOND(CPF)
      CPU=CPF-CPI
      WRITE(*,'(A,F8.4)')'3-D GRID GENERATION TIME = ',CPU
      STOP
      END
      SUBROUTINE SETUP
      COMMON/SET/RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,KMAX,P,Q,R,IPLUS,KP
     +LUS,MINOR,MAJOR,CENTRA,ITERAT,IPM,JCENT
      LOGICAL CENTRA
      REAL MAJOR,MINOR
C
C INITIALIZE VARIABLES
C
      RMACH=5.0
      PHIL=60.0
      DELTA=5.0
      DELTA1=1.05
      N=8
      IMAX=60
      JMAX=20
      KMAX=15
      P=0.0
      Q=0.0
      R=0.0
      IPLUS=1
      KPLUS=1
      MAJOR=4.0
      CENTRA=.TRUE.
      ITERAT=1
      MINOR=4.0
      IPM=8
      JCENT=10
C
C WRITE OUT MENU
C
    5 WRITE(*,'(A)')'MENU TO GENERATE GRID AROUND A WAVERIDER'
      WRITE(*,'(A)')'0) GENERATE GRID'
      WRITE(*,'(A)')'1) RE-PRINT MENU'
      WRITE(*,'(A,F6.3)')'2) MACH NUMBER = ',RMACH
      WRITE(*,'(A,F6.3)')'3) DIHEDRAL ANGLE = ',PHIL
      WRITE(*,'(A,F6.3)')'4) BASIC-CONE SEMI-VERTEX ANGLE = ',DELTA
      WRITE(*,'(A,F6.3)')'5) BASIC-CONE SEMI-VERTEX ANGLE < DELTA1 < SHO
     +CK ANGLE, DELTA1 = ',DELTA1
      WRITE(*,'(A,I4)')'6) SHAPE FACTOR, AN EVEN INTEGER = ',N
      WRITE(*,'(A,I4)')'7) NUMBER OF GRID POINTS ALONG BODY SURFACE = ',
     +IMAX
      WRITE(*,'(A,I4)')'8) NUMBER OF GRID POINTS NORMAL FROM BODY SURFAC
     +E = ',JMAX
      WRITE(*,'(A,I4)')'9) NUMBER OF GRID POINTS IN STREAMWISE DIRECTION
     + = ',KMAX
      WRITE(*,'(A,F6.3)')'10) STRETCHING IN BODY DIRECTION ON OUTER BOUN
     +DARY ONLY] = ',P
      WRITE(*,'(A,F6.3)')'11) STRETCHING IN DIRECTION NORMAL TO BODY SUR
     +FACE = ',Q
      WRITE(*,'(A,F6.3)')'12) STRETCHING IN STREAMWISE DIRECTION = ',R
      WRITE(*,'(A,I4)')'13) PARABOLIC UPSTREAM BOUNDARY IN BODY DIRECTIO
     +N = ',IPLUS
      WRITE(*,'(A,I4)')'14) PARABOLIC UPSTREAM BOUNDARY IN STREAMWISE DI
     +RECTION =',KPLUS
      WRITE(*,'(A,F6.3)')'15) SEMI-MAJOR AXIS FOR OUTER BOUNDARY ELLIPSE
     + = ',MAJOR
      WRITE(*,'(A,F6.3)')'16) SEMI-MINOR AXIS FOR OUTER BOUNDARY ELLIPSE
     + = ',MINOR
      IF(CENTRA) THEN
       WRITE(*,'(A)')'17) SECOND ORDER CENTRAL DIFFERENCING ON METRICS'
      ELSE
       WRITE(*,'(A)')'17) FIRST ORDER BACKWARDS DIFFERENCING ON METRICS'
      ENDIF
      WRITE(*,'(A,I4)')'18) NUMBER OF ITERATIONS = ',ITERAT
      WRITE(*,'(A)')'19) NUMBER OF EXTRA POINTS ON WING TIP FOR A PRANDT
     +L-MEYER TYPE EXPANSION'
      WRITE(*,'(A,I4)')'    (EVEN INTEGER) = ',IPM
      WRITE(*,'(A,I4)')'20) POINT AT WHICH HYPERBOLIC TANGENT IS CENTER
     +FOR ORTHOGONALITY = ',JCENT
   10 WRITE(*,'(A)')'ENTER DIRECTIVE'
      READ(*,*) IDIR
      IDIR=IDIR+1
C
C MODIFY VARIABLES
C
      GO TO (15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,
     +110,115),IDIR
   15 RETURN
   20 GO TO 5
   25 WRITE(*,'(A)')'ENTER MACH NUMBER'
      READ(*,*) RMACH
      GO TO 10
   30 WRITE(*,'(A)')'ENTER DIHEDRAL ANGLE'
      READ(*,*) PHIL
      GO TO 10
   35 WRITE(*,'(A)')'ENTER BASIC-CONE SEMI-VERTEX ANGLE'
      READ(*,*) DELTA
      GO TO 10
   40 WRITE(*,'(A)')'ENTER DELTA1'
      READ(*,*) DELTA1
      GO TO 10
   45 WRITE(*,'(A)')'ENTER SHAPE FACTOR'
      READ(*,*) N
      GO TO 10
   50 WRITE(*,'(A)')'ENTER IMAX'
      READ(*,*) IMAX
      GO TO 10
   55 WRITE(*,'(A)')'ENTER JMAX'
      READ(*,*) JMAX
      GO TO 10
   60 WRITE(*,'(A)')' ENTER KMAX'
      READ(*,*) KMAX
      GO TO 10
   65 WRITE(*,'(A)')'ENTER STRETCHING FACTOR IN BODY DIRECTION'
      READ(*,*) P
      GO TO 10
```

```
 70 WRITE(*,'(A)')'ENTER STRETCHING FACTOR NORMAL TO BODY DIRECTION'
    READ(*,*) Q
    GO TO 10
 75 WRITE(*,'(A)')'ENTER STRETCHING FACTOR IN STREAMWISE DIRECTION'
    READ(*,*) R
    GO TO 10
 80 WRITE(*,'(A)')'ENTER UPSTREAM BOUNDARY IN BODY DIRECTION'
    READ(*,*) IPLUS
    GO TO 10
 85 WRITE(*,'(A)')'ENTER UPSTREAM BOUNDARY IN STREAMWISE DIRECTION'
    READ(*,*) KPLUS
    GO TO 10
 90 WRITE(*,'(A)')'ENTER SEMI-MAJOR AXIS FOR OUTER BOUNDARY ELLIPSE'
    READ(*,*) MAJOR
    GO TO 10
 95 WRITE(*,'(A)')'ENTER SEMI-MINOR AXIS FOR OUTER BOUNDARY ELLIPSE'
    READ(*,*) MINOR
    GO TO 10
100 CENTRA=.NOT.CENTRA
    GO TO 10
105 WRITE(*,'(A)')'ENTER NUMBER OF ITERATIONS'
    READ(*,*) ITERAT
    GO TO 10
110 WRITE(*,'(A)')'ENTER EXTRA POINTS TO BE PLACED ON WING TIP (EVEN I
   +NTEGER)'
    READ(*,*) IPN
    GO TO 10
115 WRITE(*,'(A)')'ENTER POINT LOCATION WHERE TANH IS TO BE CENTERED'
    READ(*,*) JCENT
    GO TO 10
    END
```

```
      SUBROUTINE INTERP(MAX,I,XNOSE)
      COMMON/SET/RMACH,PHIL,DELTA,DELTA1,N,IMAX,JMAX,KMAX,P,Q,R,IPLUS,KP
     +LUS,MINOR,MAJOR,CENTRA,ITERAT,IPM,JCENT
      COMMON ANS(100)
      COMMON/ARC/APHI(0:180),ARB(0:180),ARFS(0:180),ARCB(0:180),ARCFS(0:
     +180),ARE(0:180),ARCE(0:180)
      COMMON/SBOUND/XYZS1(100,2),XYZS2(100,2)
      COMMON/BETA/BETA,PI,RADIAN
      COMMON/LENGTH/BARC,FSARC,EARC
      K=0
      GO TO (35,5,20),I
    5 ITAB=MAX+IPM/2
C
C IF USING REFINED POINTS, CALCULATE THOSE POINTS NEAR THE WING
C TIP
C
      IF(IPM .GT. 0) THEN
       MMAX=MAX+IPM/2
       ANS(MMAX)=ANS(MMAX-IPM/2)
       DO 6 J=1,IPM/2
    6  ANS(MMAX-J)=BARC-J/10000.
      ENDIF
C
C INTERPOLATE TO FIND VALUES OF LOWER BODY THAT CORRESPOND TO THE ARCLENGTH
C DISTRIBUTION
C
      DO 15 J=1,MMAX
   10 IF(ARCB(K) .LE. ANS(J) .AND. ARCB(K+1) .GE. ANS(J)) THEN
       FACTOR=ABS((ANS(J)-ARCB(K))/(ARCB(K+1)-ARCB(K)))
       RR=FACTOR*ABS(ARB(K+1)-ARB(K))+ARB(K)
       PHI=FACTOR*ABS(APHI(K+1)-APHI(K))+APHI(K)
       XYZS1(J,1)=RR*COS(PHI*RADIAN)
       XYZS1(J,2)=RR*SIN(PHI*RADIAN)
      ELSE
       K=K+1
       GO TO 10
      ENDIF
   15 CONTINUE
      RETURN
C
C AGAIN CALCULATE REFINED POINTS ON UPPER SURFACE NEAR WING TIP
C
   20 IF(IPM .GT. 0) THEN
       MMAX=MAX+IPM/2
       ANS(MMAX)=ANS(MMAX-IPM/2)
       DO 21 J=1,IPM/2
   21  ANS(MMAX-J)=FSARC-J/10000.
      ENDIF
C
C INTERPOLATE TO FIND VALUES OF UPPER BODY THAT CORRESPOND TO THE ARCLENGTH
C DISTRIBUTION
C
      DO 30 J=1,MMAX
   25 IF(ARCFS(K) .LE. ANS(J) .AND. ARCFS(K+1) .GE. ANS(J)) THEN
        FACTOR=ABS((ANS(J)-ARCFS(K))/(ARCFS(K+1)-ARCFS(K)))
        RR=FACTOR*ABS(ARFS(K+1)-ARFS(K))+ARFS(K)
        PHI=FACTOR*ABS(APHI(K+1)-APHI(K))+APHI(K)
C
C LOAD POINTS BELOW LOWER BODY POINTS
C
        XYZS1(ITAB+MMAX-J,1)=RR*COS(PHI*RADIAN)
        XYZS1(ITAB+MMAX-J,2)=RR*SIN(PHI*RADIAN)
      ELSE
        K=K+1
        GO TO 25
      ENDIF
   30 CONTINUE
      RETURN
   35 DO 45 J=1,MAX
C
C INTERPOLATE TO FIND VALUES ON ELLIPSE THAT CORRESPOND TO THE ARCLENGTH
C DISTRIBUTION
C
   40 IF(ARCE(K) .LE. ANS(J) .AND. ARCE(K+1) .GE. ANS(J)) THEN
        FACTOR=ABS((ANS(J)-ARCE(K))/(ARCE(K+1)-ARCE(K)))
        RR=FACTOR*ABS(ARE(K+1)-ARE(K))+ARE(K)
        PHI=FACTOR*ABS(APHI(K+1)-APHI(K))+APHI(K)
        XYZS2(J,1)=RR*COS(PHI*RADIAN)+XNOSE
        XYZS2(J,2)=RR*SIN(PHI*RADIAN)
      ELSE
        K=K+1
        GO TO 40
      ENDIF
   45 CONTINUE
      RETURN
      END
```

Vita

1Lt Steven Glenn Miller was born on 25 October 1959 in Minneapolis, Minnesota. He graduated from high school in Shelton, Connecticut, in 1977. He received his Bachelor of Science in Aerospace Engineering and Mechanics in December 1981 from the University of Minnesota. Upon graduation, he received a commission in the USAF through the ROTC scholarship program. He was then assigned to the Flight Dynamics Laboratory as an Aeronautical Engineer, where he was in charge of utilizing a linearized full potential flow field solver called PANAIR. He generated computer models of prospective advanced aircraft configurations and compared wind tunnel results with those generated by PANAIR. During this time he co-authored two papers on the numerical aerodynamics analysis of canard/wing configurations, AIAA-83-0009 and AIAA-83-1829. He wrote pre- and post-processor programs to take advantage of color raster terminals, and color graphics stroke/refresh terminals, until entering the School of Engineering, Air Force Institute of Technology, in June 1984.

Permanent Address: 23 Cold Spring Circle
Shelton, Connecticut
06484

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release;
distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GA/AA/85D-7

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Engineering

6b. OFFICE SYMBOL *(If applicable)*
AFIT/ENG

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State and ZIP Code)*
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

7b. ADDRESS *(City, State and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
Flight Dynamics Laboratory

8b. OFFICE SYMBOL *(If applicable)*
AFWAL/FIMM

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State and ZIP Code)*
AFWAL/FIMM
Wright-Patterson AFB, Ohio 45433

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*
See Box 19

12. PERSONAL AUTHOR(S)
Steven G. Miller, B.S., 1Lt, USAF

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT *(Yr., Mo., Day)*
1985 December

15. PAGE COUNT
141

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.
01	01	
20	04	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Grids, Fluid Mechanics, Navier Stokes Equations, Three Dimensional Flow.

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Title: NUMERICAL GRID GENERATION FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS USING MARCHING TECHNIQUES

Thesis Chairman: Dr. James K. Hodge, Major, USAF
Associate Professor of Aeronautical & Astronautical Engineering

Approved for public release: IAW AFR 190-1.
Lynn Wolaver 16 JAN 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Dr. James K. Hodge, Major, USAF

22b. TELEPHONE NUMBER *(Include Area Code)*
513-255-3517

22c. OFFICE SYMBOL
AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

Two- and three-dimensional surface normal grids are generated in Cartesian coordinates around a supersonic/hypersonic waverider configuration using parabolic partial differential equations. The elliptic partial differential equations for grid generation are parabolized in the ξ direction in two dimensions, and in the ξ and ζ directions in three dimensions. This is consistent with spatial marching flow solutions. The parabolized grid equations march in the ξ direction for two dimensions and in both the ξ and ζ directions for three dimensions, without iteration. The following problems are investigated: describing the boundary points, generating grids around the waverider's wing tip, using approximations to the elliptic grid generation equations too far downstream around a convex corner, and grid crossover in concave regions when orthogonality is specified. The degree of grid smoothing in the marching directions is related to the positioning of the approximations to the elliptic grid generation equations. Highly stretched surface orthogonal grids are accurately and efficiently generated without grid embedding for high Reynold's number flows. Keywords: [illegible]

END

FILMED

3-86

DTIC