NASA Contractor Report 187125

Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems

Task 5 – Unsteady Counterrotation Ducted Propfan Analysis
Computer Program User's Manual

Edward J. Hall and Robert A. Delaney
Allison Gas Turbine Division of General Motors
Indianapolis, Indiana

January 1993

Prepared for
Lewis Research Center
Under Contract NAS3–25270

Preface

This manual was prepared by Edward J. Hall and Robert A. Delaney of the Allison Gas Turbine Division, General Motors Corporation, Indianapolis, IN. The work was performed under NASA Contract NAS3-25270 from April, 1991 to September, 1992. The flow code theory, and programming modifications necessary for the aerodynamic analysis were performed by Edward J. Hall. The Allison program manager for this contract was Robert A. Delaney. The NASA project manager for this contract was Christopher J. Miller.

Acknowledgements

The authors would like to express their appreciation to the following NASA personnel who contributed to this program:

Dr. John. J. Adamczyk for the many helpful technical discussions concerning the development of the computer codes,

Dr. Christopher J. Miller for his suggestions and critical review of the program,

Dr. Andrea Arnone and Dr. Charles Swanson for their assistance during the development of the multigrid algorithm.

The services of the NASA Numerical Aerodynamic Simulation (NAS) facilities and personnel are gratefully acknowledged.

UNIX is a trademark of AT&T

IRIS is a trademark of Silicon Graphics, Inc.

AIX is a trademark of IBM

UNICOS is a trademark of Cray Computers

PostScript is a trademark of Adobe Systems, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

NOTATION

A list of the symbols used throughout this document and their definitions is provided below for convenience.

Roman Symbols

e ... total internal energy

i ... first grid index of numerical solution

j ... second grid index of numerical solution

k ... third grid index of numerical solution or thermal conductivity

n ... rotational speed (revolutions per second) or time step level

r ... radius or radial coordinate

t ... time

u ... velocity in the axial direction

v ... velocity in the radial direction

w ... velocity in the circumferential direction

x ... Cartesian coordinate system coordinate

y ... Cartesian coordinate system coordinate

z ... Cartesian coordinate or cylindrical coordinate system axial coordinate

$ADPAC$ ... Advanced Ducted Propfan Analysis Codes

$AOACR$ ... Angle of attack/coupled row aerodynamic analysis code

$ADPOST$ ... ADPAC post processing program

$ASCII$ ... American Standard Code for Information Interchange

CFL ... Courant-Freidrichs-Lewy number $(\Delta t / \Delta t_{max,stable})$

$CHGRIDV2$ ... Ducted propfan grid generation code

D ... Propfan diameter (units of length)

$FULLPLOT$ ... PostScript x-y plotting program

J ... advance ratio $(J = U/nD)$

M ... Mach number

$MAKEADGRID$ ... ADPAC multiple-block mesh assembly program

$MULAC$ ... NASA-Lewis Compressor Mesh Generation Program

N ... Number of blades

R ... gas constant or residual or maximum radius

$ROTGRID$ ... ADPAC-AOACR mesh rotation program

$SETUP$ ... ADPAC file setup program

$SDBLIB$ ... Scientific DataBase Library (binary file I/O routines)

$TIGG3D$ ... NASA-Lewis multiple splitter mesh generation program

U ... Freestream or flight velocity (units of length/time)

V ... volume

Greek Symbols

γ ... specific heat ratio

Δ ... calculation increment

ρ ... density

μ ... coefficient of viscosity

Subscripts

$[\ ]_{coarse}$ ... coarse mesh value

$[\ ]_{fine}$ ... fine mesh value

$[\ ]_{i,j,k}$ ... grid point index of variable

$[\ ]_{max}$ ... maximum value

$[\ ]_{min}$ ... minimum value

$[\ ]_{non-dimensional}$ ... non-dimensional value

$[\ ]_{ref}$ ... reference value

$[\ ]_{stable}$ ... value implied by linear stability

$[\ ]_{total}$ ... total (stagnation) value

1. SUMMARY

The primary objective of this study was the development of a time-marching three-dimensional Euler/Navier-Stokes aerodynamic analysis to predict steady and unsteady compressible transonic flows about ducted and unducted propfan propulsion systems employing multiple blade rows. The computer codes resulting from this study are referred to as *ADPAC-AOACR* (Advanced Ducted Propfan Analysis Codes-Angle of Attack/Coupled Row). This report is intended to serve as a computer program user's manual for the *ADPAC-AOACR* codes developed under Task 5 of NASA Contract NAS3-25270, Unsteady Counterrotating Ducted Propfan Analysis.

The *ADPAC-AOACR* program is based on a flexible multiple-block grid discretization scheme permitting coupled 2-D/3-D mesh block solutions with application to a wide variety of geometries. For convenience, several standard mesh block structures are described for turbomachinery applications. Aerodynamic calculations are based on a four-stage Runge-Kutta time-marching finite volume solution technique with added numerical dissipation. Steady flow predictions are accelerated by a multigrid procedure. Numerical calculations are compared with experimental data for several test cases to demonstrate the utility of this approach for predicting the aerodynamics of modern turbomachinery configurations employing multiple blade rows.

2. INTRODUCTION

This document contains the Computer Program User's Manual for the *ADPAC-AOACR* (Advanced Ducted Propfan Analysis Codes - Angle of Attack/Coupled Row) 3-D Euler/Navier-Stokes analysis developed by the Allison Gas Turbine Division of the General Motors Corporation under Task V of NASA Contract NAS3-25270. The objective of this study was to develop a three-dimensional time-dependent Euler/Navier-Stokes analysis for high-speed ducted propfan aircraft propulsion systems employing multiple blade rows. The analysis is capable of predicting both steady state and time-dependent flowfields using coupled 2-D/3-D solution concepts (described in detail in Section 1.3). Throughout the rest of this document, the aerodynamic analysis is referred to as *ADPAC-AOACR* to signify that it is an *ADPAC* code developed for Angle of Attack/Coupled Row configurations.

A complete theoretical development of the *ADPAC-AOACR* program is outlined in a companion Final Report for Task V of NASA Contract NAS3-25270 [1]. In brief, the program utilizes a finite-volume, time-marching numerical procedure in conjunction with a flexible, coupled 2-D/3-D multiple grid block geometric representation to permit detailed aerodynamic simulations about complex configurations. The analysis has been tested and results verified for both turbomachinery and non-turbomachinery based applications. The ability to accurately predict the aerodynamics due to the interactions between adjacent blade rows of modern, high speed turbomachinery was of particular interest in this program, and therefore, emphasis is given to these types of calculations throughout the remainder of this document. It should be emphasized at this point that although the *ADPAC-AOACR* program was developed to analyze the steady and unsteady aerodynamics of high-bypass ducted fans employing multiple blade rows, the code possesses many features which make it practical to compute a number of other complicated flow configurations as well.

3

2.1 Multiple-Block Solution Domain Concepts

In order to appreciate and utilize the features of the *ADPAC-AOACR* solution system, the concept of a multiple-block grid system must be fully understood. It is expected that the reader possesses at least some understanding of the concepts of computational fluid dynamics (CFD), so the use of a numerical grid to discretize a flow domain should not be foreign. Many CFD analyses rely on a single structured ordering of grid points upon which the numerical solution is performed (the authors are aware of a growing number of unstructured grid solution techniques as well, but resist the temptation to mention them in this discussion). Multiple-block grid systems are different only in that several structured grid systems are used in harmony to generate the numerical solution. This concept is illustrated graphically in two dimensions for the flow through a nozzle in Figures 2.1-2.3.

The grid system in Figure 2.1 employs a single structured ordering, resulting in a single computational space to contend with. The mesh system in Figure 2.2 is comprised of two, separate structured grid blocks, and consequently, the numerical solution consists of two unique computational domains. In theory, the nozzle flowpath could be subdivided into any number of domains employing structured grid blocks resulting in an identical number of computational domains to contend with, as shown in the 20 block decomposition illustrated in Figure 2.3. The complicating factor in this domain decomposition approach is that the numerical solution must provide a means for the isolated computational domains to communicate with each other in order to satisfy the conservation laws governing the desired aerodynamic solution. Hence, as the number of subdomains used to complete the aerodynamic solution grows larger, the number of inter-domain communication paths increases in a corresponding manner. (It should be noted that this domain decomposition/communication overhead relationship is also a key concept in parallel processing for large scale computations, and thus, the *ADPAC-AOACR* code appears to be a viable candidate for parallelization via the natural domain decomposition division afforded by the multiple-block grid data structure.) For the simple nozzle case illustrated in Figure 2.1 it would seem that there is no real advantage in using a multiple-block grid, and this is probably true. For more complicated geometries, such as the turbine vane coupled O-H

4

ADPAC–AOACR 2–D Single Block Mesh Structure Illustration
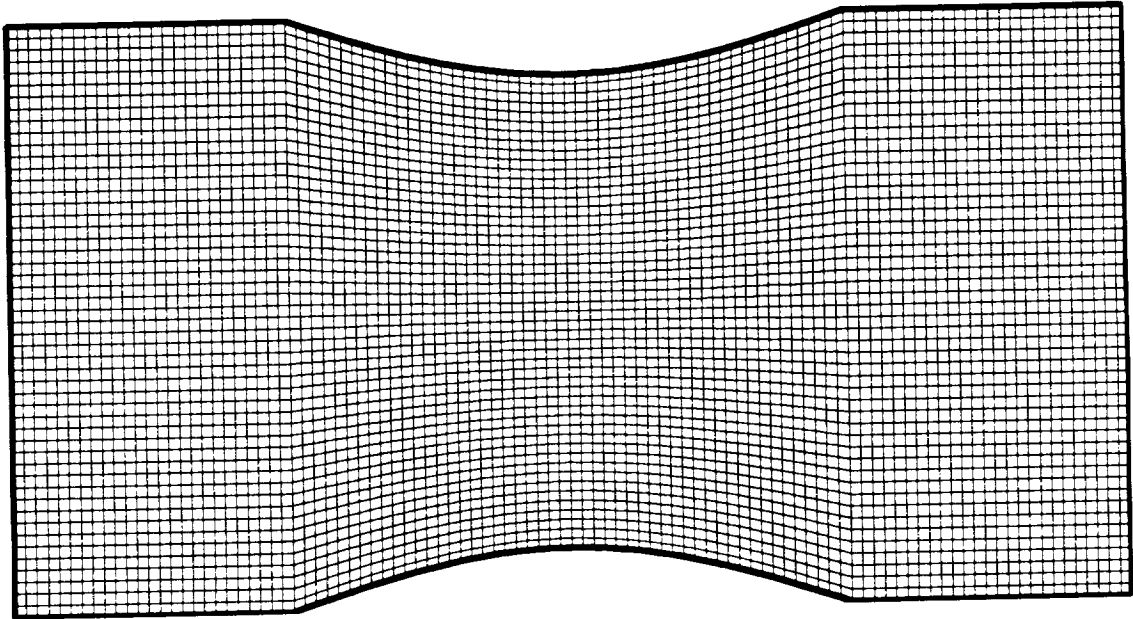
Physical Domain

Computational Domain
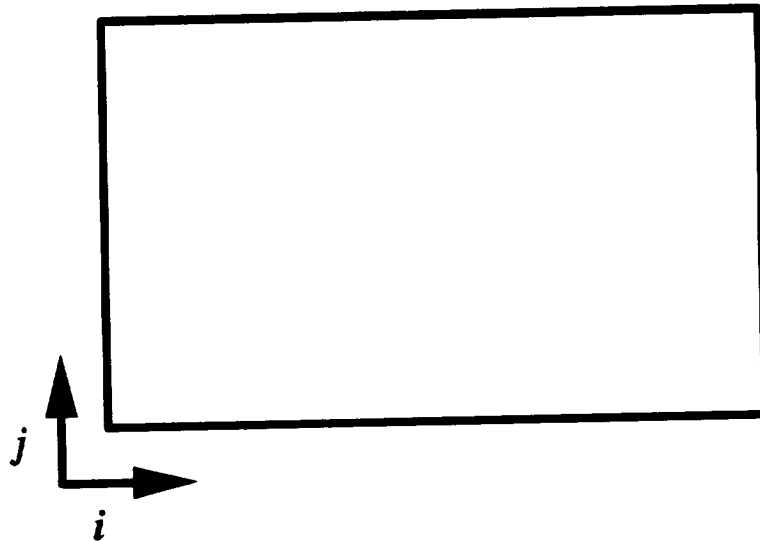
Figure 2.1: *ADPAC-AOACR* 2-D Single Block Mesh Structure Illustration

ADPAC-AOACR 2-D Two Block Mesh Structure Illustration
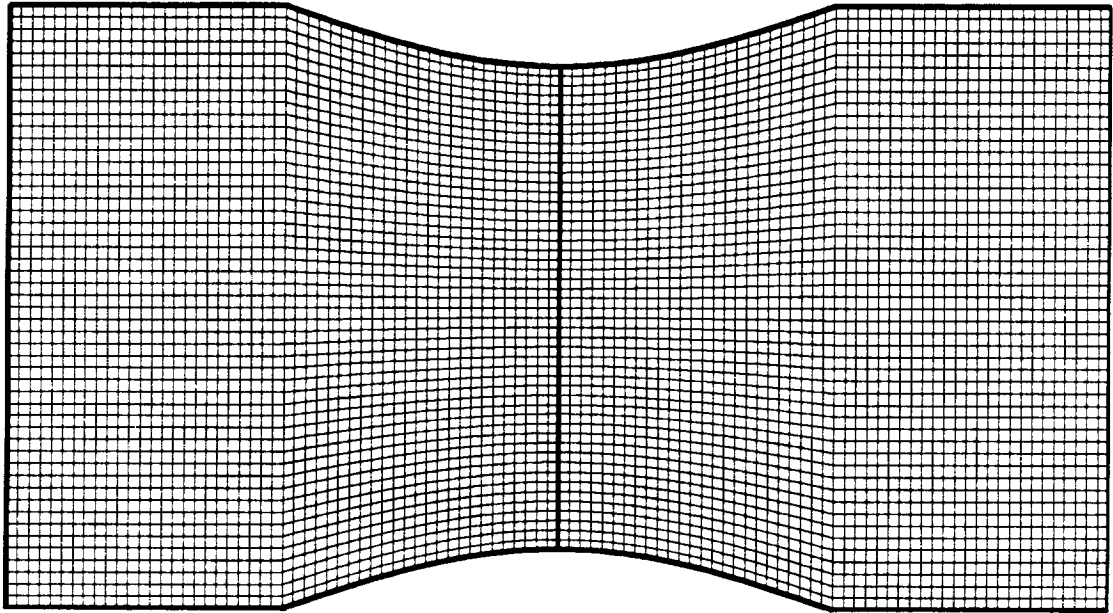
Physical Domain

Computational Domain
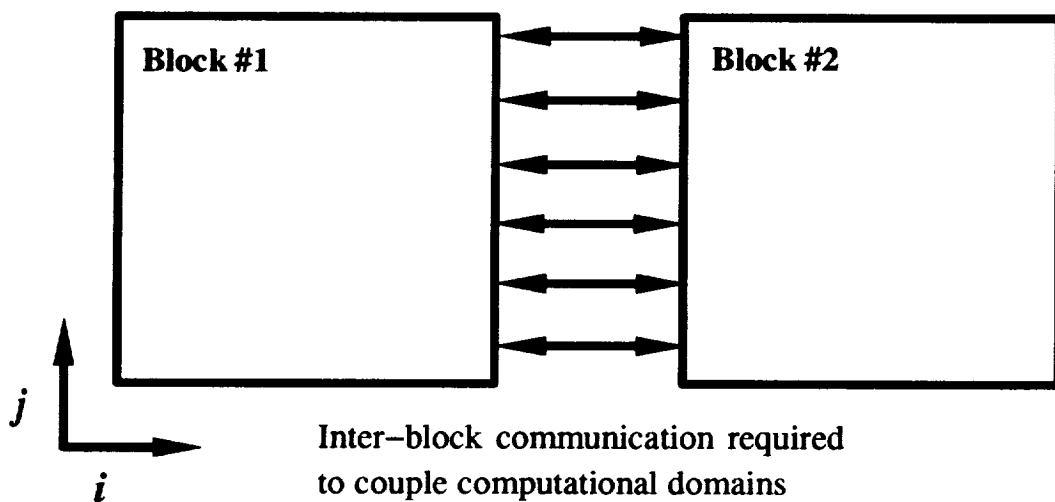
Figure 2.2: *ADPAC-AOACR* 2-D Two Block Mesh Structure Illustration

ADPAC-AOACR   2-D Multiple Block Mesh Structure Illustration
Physical Domain

Computational Domain

Inter-block communication required
to couple computational domains
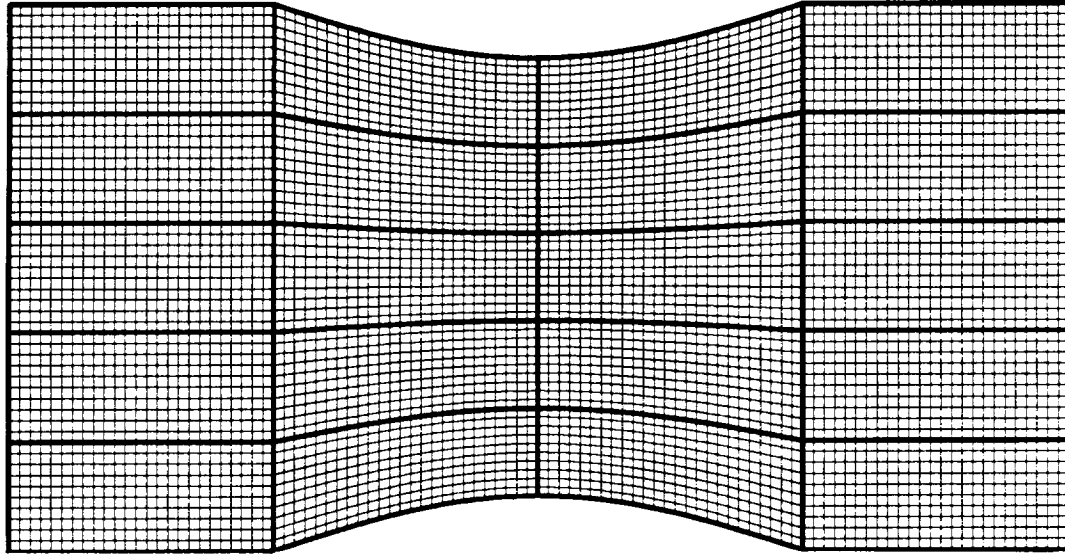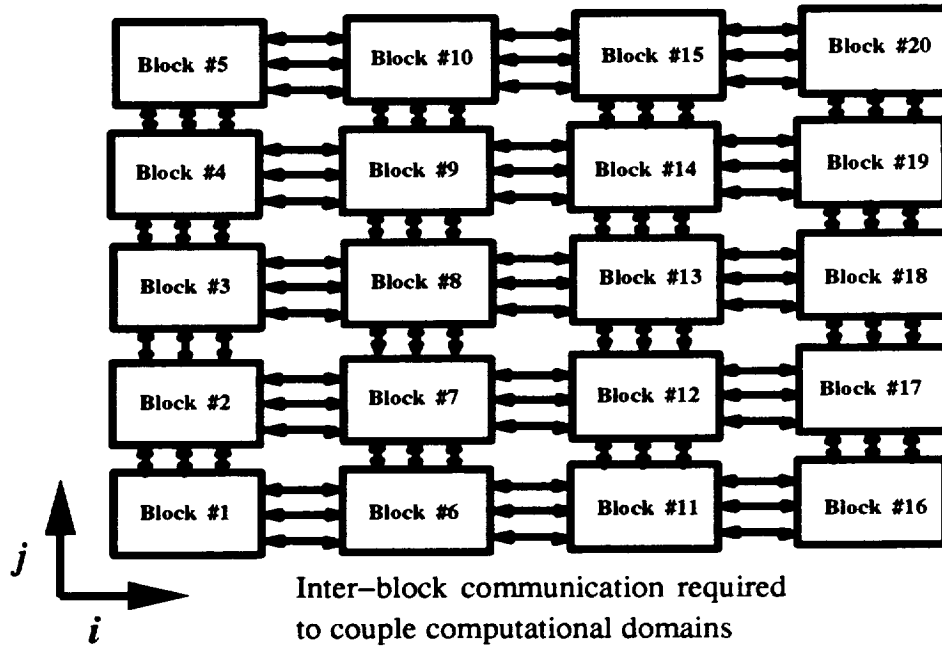
Figure 2.3:   *ADPAC-AOACR* 2-D Multiple Block Mesh Structure Illustration

grid system shown in Figure 2.4 and the corresponding computational domain communication scheme shown in Figure 2.5, it may not be possible to generate a single structured grid to encompass the domain of interest without sacrificing grid quality, and therefore, a multiple-block grid system has significant advantages.

The *ADPAC-AOACR* code utilizes the multiple-block grid concept to full extent by permitting an arbitrary number of structured grid blocks with user specifiable communication paths between blocks. The inter-block communication paths are implemented as a series of boundary conditions on each block which, in some cases, communicate flow information from one block to another. The advantages of the multiple-block solution concept are exploited throughout the remainder of this document as a means of treating complicated geometries with multiple blade rows of varying blade number, and to exploit computational enhancements such as multigrid.

2.2 Multiple Blade Row Solution Concepts

Armed with an understanding of the multiple-block mesh solution concept discussed in the previous section, it is now possible to describe how this numerical solution technique can be applied to predict the flow through rotating machinery with multiple blade rows. Historically, the prediction of three-dimensional flows through multistage turbomachinery has been based on one of three solution schemes. These schemes are briefly illustrated and described in Figure 2.6.

The first scheme involves predicting the time-resolved unsteady aerodynamics resulting from the interactions occurring between relatively rotating blade rows. Examples of this type of calculation are given by Rao and Delaney [4], Jorgensen and Chima [5], and Rai [9]. This approach requires either the simulation of multiple blade passages per blade row, or the incorporation of a phase-lagged boundary condition to account for the differences in spatial periodicity for blade rows with dissimilar blade counts. Calculations of this type are typically computationally expensive, and are presently impractical for machines with more than 2-3 blade rows.

The second solution technique is based on the average-passage equation system developed by Adamczyk [6]. In this approach, separate 3-D solution domains are

ADPAC-AOACR Multiple Block Mesh System for a Turbine Blade Row (Combination O-H Mesh)
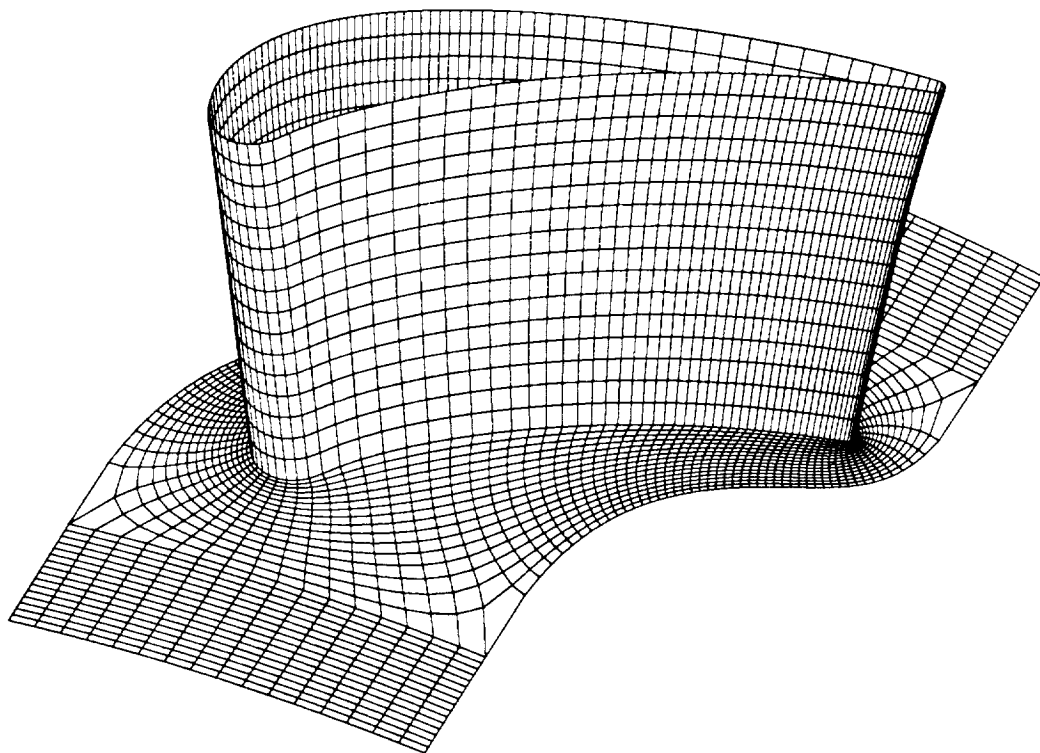
Figure 2.4: Coupled O-H Grid System for a Turbine Vane Cascade

ADPAC-AOACR 3-D O-H Turbine Grid Mesh Structure Illustration
Computational Domain

Block #2
(Forward H-Grid)

Block #3
(Rearward H-Grid)
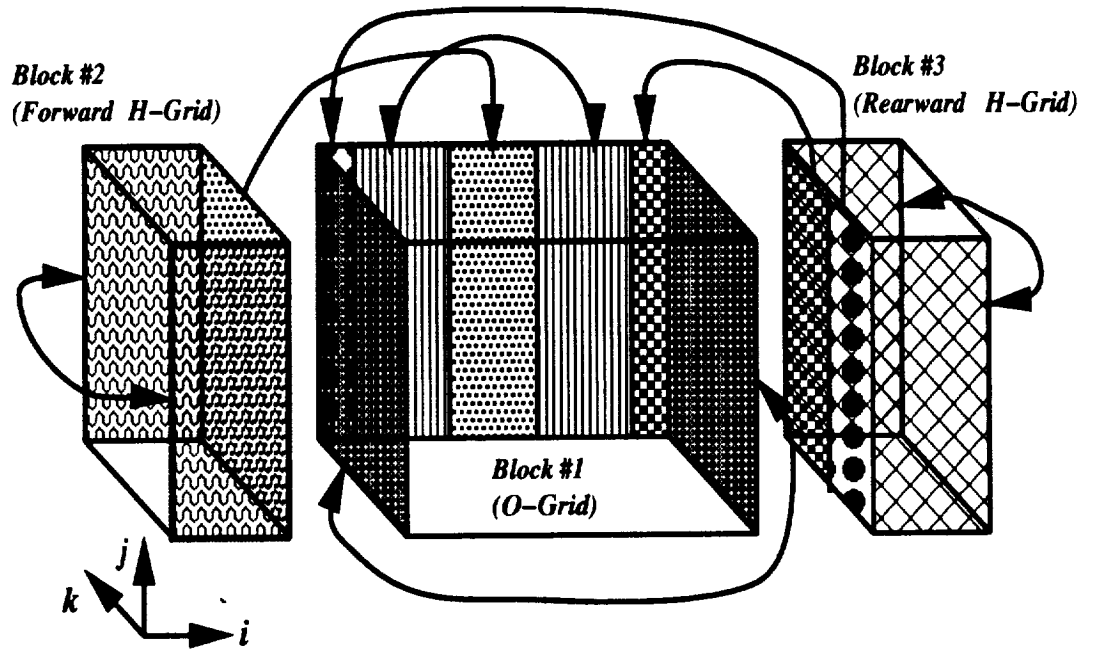
Block #1
(O-Grid)

Block to Block Communication Patches

Forward H-Grid
Periodic Boundary

Rearward H-Grid
Periodic Boundary

Rearward H-Grid
O-Grid Connection

Forward H-Grid
O-Grid Connection

Rearward H-Grid
O-Grid Connection

O-grid cut

O-Grid Periodic
Boundary

Other boundaries (walls,
inflow/outflow, etc.
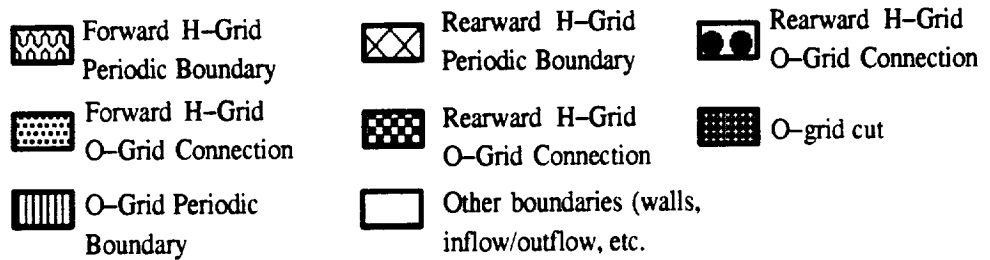
Figure 2.5: Computational Domain Communication Scheme for Turbine Vane O-H Grid System

10

Multiple Blade Row Numerical Solution Concepts

3-D Rotor/Stator Interaction

Average-Passage Simulation

Circumferential Mixing Plane

Mixing Plane

Axisymmetric Representation
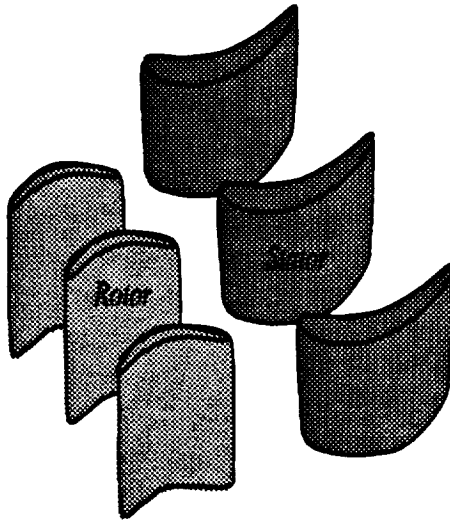
3-D time-dependent Navier-Stokes equations

Multiple blade passages for each blade row or phase-lagged boundaries

Time-dependent coupling of individual blade passage domains

Computationally expensive
  Multiple blade passages per blade row
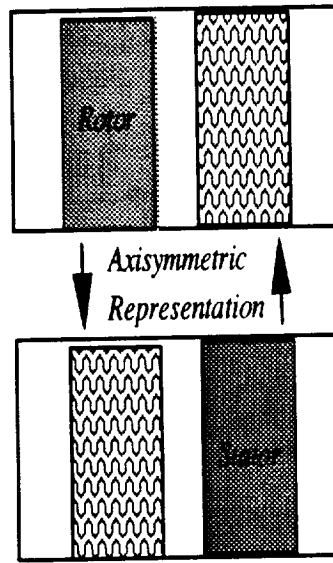
Average-passage equation system

3-D steady solution of entire domain for each blade row

Adjacent blade rows represented by blockage/body forces in 3-D solution

Solutions have common axisymmetric flowfield

Correlation model for mixing terms

Computational cost still rather high
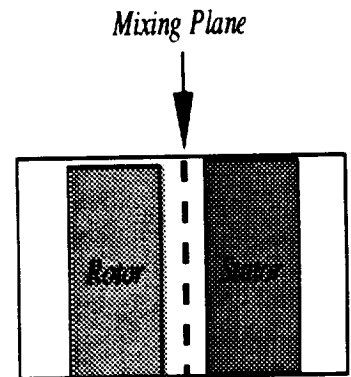
Steady Navier Stokes solution

Computational domain limited to near blade region

Circumferential mixing plane provides inter-blade row communication

Lower computational cost

Figure 2.6:   Multiple Blade Row Numerical Solution Schemes

11

defined for each blade row which encompasses the overall domain for the entire turbomachine. The individual solution domains are specific to a particular blade row, although all blade row domains share a common axisymmetric flow. In the solution for the flow through a specific blade passage, adjacent blade rows are represented by their time and space-averaged blockage, body force, and energy source contributions to the overall flow. A correlation model is used to represent the time and space-averaged flow fluctuations representing the interactions between blade rows. The advantage of the average-passage approach is that the temporally and spatially averaged equations system reduce the solution to a steady flow environment; and, within the accuracy of the correlation model, the solution is representative of the average aerodynamic condition experienced by a given blade row under the influence of all other blade rows in the machine. The disadvantage of the average-passage approach is that the solution complexity and cost grow rapidly as the number of blade passages increases, and the validity of the correlation model is as yet unverified.

The third approach for the prediction of flow through multistage turbomachinery is based on the mixing plane concept. A mixing plane is an arbitrarily imposed boundary inserted between adjacent blade rows across which the flow is "mixed out" circumferentially. This circumferential mixing approximates the time-averaged condition at the mixing plane and allows the aerodynamic solution for each blade passage to be performed in a steady flow environment. The mixing plane concept was recently applied to realistic turbofan engine configurations by Dawes [7]. Flow variables on either side of the mixing plane are circumferentially averaged and passed to the neighboring blade row as a means of smearing out the circumferential nonuniformities resulting from dissimilar blade counts. The mixing plane concept is a much more cost-effective approach computationally because the flow is steady, and the individual blade passage domains are limited to a near-blade region. Unfortunately, the accuracy of this approach is clearly questionable under some circumstances because of the placement of the mixing plane and the loss of spatial information resulting from the circumferential averaging operator.

The *ADPAC-AOACR* program possesses features which permit multiple blade row solutions using either the time-dependent interaction approach or the mixing plane concept, described above. Average-passage simulations for realistic turbofan

engine configurations were recently reported under Task 4 of this contract, and further details on this approach can be found in Reference [8]. *ADPAC-AOACR* predictions utilizing the time-accurate rotor/stator interaction technique requires that a sufficient number of blade passages be represented in each row such that the circumferential distance represented in each blade row is constant. This limits the blade counts which can be effectively simulated through this technique. For example, for the simple single-stage calculation suggested in Figure 2.6, if the rotor has 36 blades and the stator has 48 blades, a time dependent solution would require, as a minimum, 3 rotor blade passages and 4 stator blade passages to accomodate the common circumferential pitch requirement. If the rotor has 35 blades, and the stator has 47 blades, however, then both blade rows would require that every blade passage be modeled as no simpler reduction in blade count is possible. This restriction will appear quite often, as turbomachinery designers often do not like to design neighboring blade rows with blade counts which have a common integer factor. Ultimately, this type of problem will require the incorporation of a phase-lagged boundary condition which would permit time-dependent interaction solutions for neighboring blades using only one blade passage per blade row.

If, instead, a mixing plane type of calculation is desired, then the multiple block scheme may again be invoked by utilizing a single blade passage per blade row, where each grid block has a common mating surface with a neighboring blade row. The only special requirement here is that boundary condition routines be available to adequately perform the circumferential averaging between blade rows and supply the block-to-block communication of this information in the multiple-block mesh solution algorithm. Section 3.7 describes the techniques for applying this type of boundary condition.

2.3 2-D/3-D Solution Zooming Concepts

A third unique feature of the *ADPAC-AOACR* solution system involves the concept of coupling two-dimensional and three-dimensional solution domains to obtain representative simulations of realistic high bypass ducted fan engine concepts. A complicating factor in the analysis of flows through turbofan engine systems results from

13

the interactions between adjacent blade rows, and, in the case of a ducted fan, the effects of downstream blade rows on the aerodynamics of the upstream fan rotor. Historically, in the design of multistage turbomachinery, an axisymmetric representation of the flow through a given blade row has been used to effectively reduce the complexity of the overall problems to a managable level. Similarly, an efficient approach to the numerical simulation of downstream blade rows could naturally utilize an axisymmetric representation of the effects of these rows through a two-dimensional grid system, with blade blockage, body force, and energy terms representing the axisymmetric averaged aerodynamic influence imparted by the embedded blade row. This concept is illustrated graphically in Figure 2.7 for a representative turbofan engine fan section.

A numerical solution of the flow through the fan rotor is complicated by the presence of the core stator, bypass stator, and bypass splitter. It is undesirable to restrict the solution domain to the fan rotor alone as this approach neglects the potential interactions between the fan rotor and the downstream geometry. The *ADPAC-AOACR* program permits coupled solutions of 3-D and 2-D mesh blocks with embedded blade row blockage, body force, and energy terms as a means of efficiently treating these more complicated configurations. Blade force terms may be determined from a separate 3-D solution, or may be directly specified based on simpler design system analyses. Neighboring 2-D and 3-D mesh blocks are numerically coupled through a circumferential averaging procedure which attempts to globally satisfy the conservation of mass, momentum and energy across the solution domain interface. The "dimensional zooming" capability permitted by the 2-D/3-D mesh coupling scheme is considered a vital asset for the accurate prediction of the flow through modern high speed turbofan engine systems.

2.4 Multigrid Convergence Acceleration Concepts

For completeness, a brief paragraph is included here to discuss the multigrid convergence acceleration solution technique incorporated into the *ADPAC-AOACR* code. Multigrid (please do not confuse this with a multiple-block grid!) is a numerical solution technique which attempts to accelerate the convergence of an iterative pro-

2-D Axisymmetric Blade Row Representation

3-D Geometry

2-D Axisymmetric Representation

3-D Computational
Domain

Rotor

Stator

Stator

Rotor

2-D Axisymmetric Representation
of Stator Blade Row – Includes the
Effects of Blockage, Body Forces and
Energy Sources
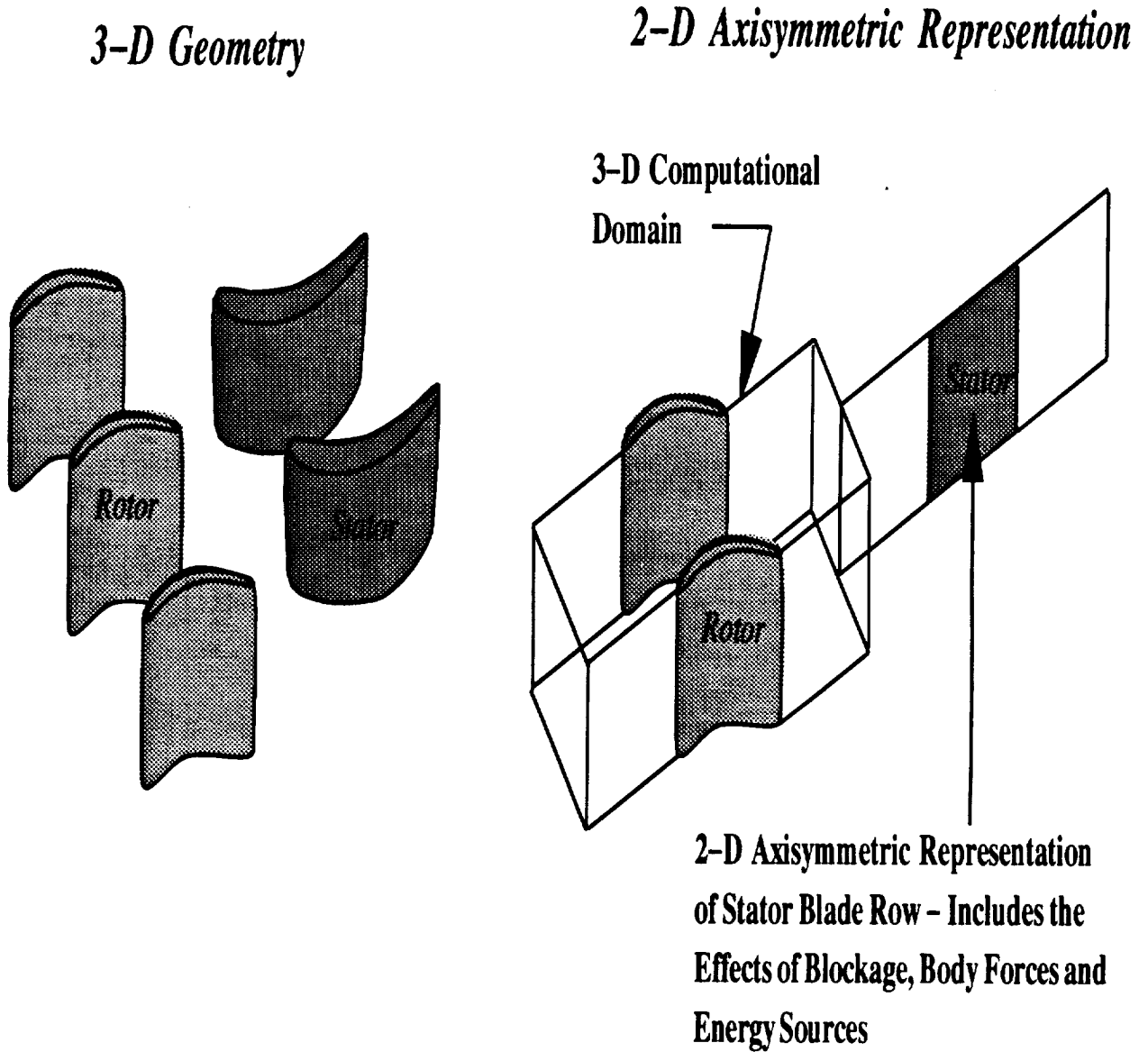
Figure 2.7:   2-D Axisymmetric Flow Representation of a Turbomachinery Blade Row

15

cess (such as a steady flow prediction using a time-marching scheme) by computing corrections to the solution on coarser meshes and propagating these changes to the fine mesh through interpolation. This operation may be recursively applied to several coarsenings of the original mesh to effectively enhance the overall convergence. Coarse meshes are derived from the preceding finer mesh by eliminating every other mesh line in each coordinate direction as shown in Figure 2.8. As a result, the number of multigrid levels (coarse mesh divisions) is controlled by the mesh size, and, in the case of the *ADPAC-AOACR* code, by the mesh indices of the boundary patches used to define the boundary conditions on a given mesh block (see Figure 2.8). These restrictions suggest that mesh blocks should be constructed such that the internal boundaries and overall size coincide with numbers which are compatible with the multigrid solution procedure (i.e., the mesh size should be 1 greater than any number which can be divided by 2 several times and remain whole numbers; e.g. 9, 17, 33, 65 etc.) Further details on the application of the *ADPAC-AOACR* multigrid scheme are given in Section 3.6 and in the companion Final Report [1].

A second multigrid concept which should be discussed is the so-called "full" multigrid startup procedure. The "full" multigrid method is used to start up a solution be initiating the calculation on a coarse mesh, performing several time-marching iterations on that mesh (which, by the way could be multigrid iterations if successively coarser meshes are available), and then interpolating the solution at that point to the next finer mesh, and repeating the entire process until the finest mesh level is reached. The intent here is to generate a reasonably approximate solution on the coarser meshes before undergoing the expense of the fine mesh multigrid cycles. Again, the "full" multigrid technique only applies to starting up a solution, and therefore, it is not normally advisable to utilize this scheme when the solution is restarted from a previous solution as the information provided by the restart data will likely be lost in the coarse mesh initialization.

2.5 General Solution Procedure Sequence

The *ADPAC-AOACR* code is distributed as a compressed *tar* file which must be processed before the code may be utilized. The instructions in Appendix A describe

Multigrid Algorithm Mesh Level Description

Fine Mesh
Level 1

Coarse Mesh
Level 2

Coarse Mesh
Level 3

Every other mesh line removed to define next mesh level

Grid lines defining mesh boundaries and internal boundaries (blade leading edges, trailing edges, etc.) must be consistent with the mesh coarsening process (cannot remove a mesh line defining a boundary for the given coordinate direction)
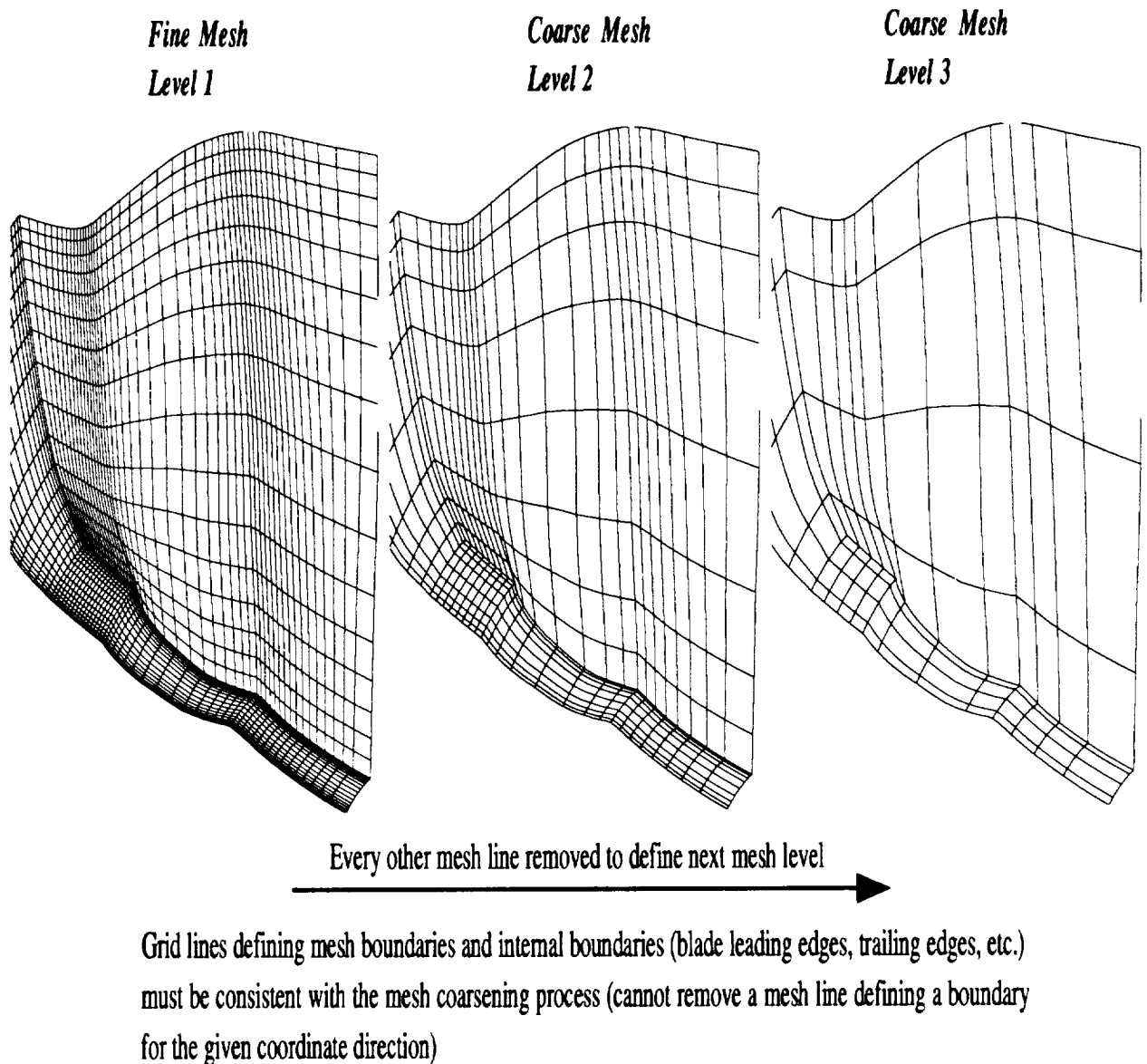
Figure 2.8: Multigrid Mesh Coarsening Strategy and Mesh Index Relation

17

how to obtain the distribution file, and extract the necessary data to run the code. This operation is typically required only once when the initial distribution is received. Once the source files have been extracted, the sequence of tasks listed below are typical of the events required to perform a successful analysis using the *ADPAC-AOACR* code.

Step 1.)   Define the problem:

This step normally involves selecting the geometry and flow conditions, and defining which specific results are desired from the analysis.The definition of the problem must involve specifying whether steady state or time-dependent data are required, whether an inviscid calculation is sufficient, or whether a viscous flow solution is required, and some idea of the relative merits of solution accuracy versus solution cost (CPU time) must be considered.

Step 2.)   Define the geometry and flow domain:

.   Typically, geometric features such as airfoils, ducts, and flowpath endwalls are required to geometrically define a ducted fan. The solution domain may be chosen to include the external flow, internal engine passage flows, and/or leakage flows. The flow domain is normally defined large enough such that the region of interest is far enough away from the external boundaries of the problem to ensure that the solution is not unduly influenced by the external boundary conditions.

Step 3.)   Define a block structure:

Once the geometry and solution domain has been numerically defined, the implementation of the solution mesh structure must be considered. This process begins with a determination of the domain block structure, if and when more than one mesh block is required for a given solution. The possibility of incorporating 2-D mesh blocks should be considered whenever possible due to the computational savings afforded by this approach (see Section 2.3). For convenience, several standard block structures for high bypass ducted fans have been constructed and will be discussed later in this document (see Chapter 5). This should not discourage users from developing their own block structures should specific needs be unresolved by the standard block structures.

Step 4.)   Generate a numerical grid for the domain of interest:

Most of the standard grid block structures defined in this document can be

18

adequately handled through either the *TIGG3D* [15] or the *CHGRIDV2* [2] grid generation programs. Other grid generation programs may be equally useful, and a conversion program called *MAKEADGRID* (described in Chapter 7.0) is included to convert non-standard meshes into *ADPAC-AOACR* format.

Step 5.)   Generate a standard input file:

The standard input file controls operations specific to a particular run of the *ADPAC-AOACR* code. Options such as the number of iterations, damping parameters, and input/output control of the code execution may all be governed by the values specified in the standard input file. For the standard block structures described above, a program called *SETUP* (see Chapter 8.0) is provided which will interactively query the user about the desired run for one of the standard block configurations, and construct an appropriate input file.

Step 6.)   Generate a boundary data file:

The boundary data file controls the application of boundary conditions on the grid block structure provided to the flow code. The boundary data specifications are specific to the mesh being used in a given calculation. For the standard block structures described above, a program called *SETUP* (see Chapter 8.0) is provided which will interactively query the user about the desired run for one of the standard block configurations, and construct an appropriate boundary data file. For other block configurations, the user must construct the boundary data file by hand according to the format described in Section 3.7.

Step 7.)   Run *ADPAC-AOACR* to predict the aerodynamics.

Chapter 3 is available to describe the commands necessary to perform this task. A sample test case is also completely outlined in Appendix A. In many cases, a given calculation will involve several applications of the *ADPAC-AOACR* code, restarted from the previous calculation as a means of breaking up a large problem into several shorter calculations.

Step 8.)   Plot and process the results:

An interactive post processing program called *ADPOST* is provided to handle tasks such as mass-averaging flow variables to simplify the interpretation of the computed results (see Chapter 4). Output data is also provided for widely available plotting programs such as *PLOT3D* [11], and *FAST* [13].

19

A condensed description of the commands involved in the steps described above beginning with extracting the source code from the distribution, compiling the codes, setting up a case, and running a case, is given in the Appendix. Separate sections are provided in the chapters which follow to describe in detail the basis and operation of the codes used in the steps above.

It is worthwhile mentioning that the development and application of the codes described in this manual were performed on Unix-based computers. All files are stored in machine-independent format. Small files utilize standard ASCII format, while larger files, which benefit from some type of binary storage format, are based on the Scientific DataBase Library (SDBLIB) format [10]. The SDBLIB format utilizes machine-dependent input/output routines which permit machine independence of the binary data file. The SDBLIB routines are under development at the NASA Lewis Research Center.

Most of the plotting and graphical postprocessing of the solutions was performed on graphics workstations. The *PLOT3D* [11], and *FAST* [13] graphics software packages developed at NASA Ames Research Center were extensively used for this purpose, and data files for these plotting packages are generated automatically. These data files are written in what is known as *PLOT3D* multiple-grid format. (See AD-PAC File Description, Section 3.5).

3. *ADPAC-AOACR*: 3-D EULER/NAVIER-STOKES FLOW SOLVER OPERATING INSTRUCTIONS

3.1 Introduction

This chapter contains the operating instructions for the time-dependent multiple grid block 3-D Euler/Navier-Stokes aerodynamic analysis *ADPAC-AOACR*. These instructions include some general information concerning the execution of the code, defining array limits, compilation of the flow solver, setting up input files, running the code, and examining output data. The *ADPAC-AOACR* flow solver source programs are written in FORTRAN 77, and have been used successfully on Cray UNICOS and IBM VM/CMS mainframe computer systems as well as IBM AIX Operating System and Silicon Graphics 4D workstations using a UNIX operating system.

3.2 General Information Concerning the Operation of the *ADPAC-AOACR* Code

Approximate computational storage and CPU requirements for the *ADPAC-AOACR* code can be conservatively estimated from the following formulas:

$$\text{CPU sec} \approx 1.0x10^{-4}(\text{\# grid points})(\text{\# iterations})$$

$$\text{Memory MW} \approx 6.8x10^{-5}(\text{\# grid points})$$

These formulas are valid for a Cray-YMP computer operating under the UNICOS environment and the cf77 compiler, version 6.0.11 and above. The times reported are for a single processor only, and are not indicative of any parallelization available through the Cray autotasking or microtasking facilities. Without multigrid, steady inviscid flow calculations normally require approximately 2000 iterations to reduce the maximum residual by three orders of magnitude (10^3) which is normally an acceptable

21

level of convergence for most calculations. Viscous flow calculations generally require 3000 or more iterations to converge. When multigrid is used, the number of iterations required to obtain a converged solution is often one third to one fourth the number of iterations required for a non-multigrid calculation. Convergence for a viscous flow case is generally less well behaved than a corresponding inviscid flow calculation, and in many cases, it is not possible to reduce the maximum residual by three orders of magnitude due to oscillations resulting from vortex shedding, shear layers, etc. A determination of convergence for a viscous flow case must often be based on observing the mass flow rate or other global parameter, and terminating the calculation when these variables no longer change. The number of iterations required for an unsteady flow calculation is highly case-dependent, and may be based on mesh spacing, overall time-period, complexity of the flow, etc.

The *ADPAC-AOACR* program produces output files suitable for plotting using the *PLOT3D* [11], *SURF* [12], and *FAST* [13] graphics software packages developed at the NASA Ames Research Center. *PLOT3D* format data files are written for both absolute and relative flows (see Chapter 2 for a description of the *PLOT3D* format). The user may also elect to have additional *PLOT3D* absolute flow data files output at constant iteration intervals during the course of the solution. These files may be used as instantaneous flow "snapshots" of an unsteady flow prediction.

3.3 Configuring Maximum Array Dimensions

The first step required before attempting to run the *ADPAC-AOACR* program is to set the maximum array size required for the analysis prior to the compilation process. The maximum array size will ultimately determine the largest problem (in terms of total number of mesh points) which can be run with the code. The larger the array limits, the larger the number of grid points which may be used. Unfortunately, setting larger array limits also increases the total amount of memory required by the program, and hence, can impede the execution of the code on memory-limited computing systems. Ideally, the code should be dimensioned just large enough to fit the problem at hand. Array dimensions are specified in the *ADPAC-AOACR* program by a PARAMETER statement in each subroutine. The array limits are specified in

the source code file **parameter.inc** by the statements:

PARAMETER(NBMAX = 10)
PARAMETER(NRA3D = 250000)
PARAMETER(NBL2D = 20000)
PARAMETER(NRA1D = 1000)
PARAMETER(NBCPBL = 15)
PARAMETER(NRAINT = 4000)
PARAMETER(NBFRA = 12000)
PARAMETER(LGRAFX = 4000)

which will ultimately appear in every subroutine through a FORTRAN **include** statement.

The PARAMETER variables are defined as:

NBMAX   Maximum number of grid blocks allowed

NRA3D   Overall number of all 3-D grid block elements

NBL2D   Maximum linear dimension squared of any 2-D or 3-D grid block

NRA1D   Overall sum of linear dimensions of all 2-D and 3-D grid blocks

NBCPBL   Maximum number of boundary specifications per block

NRAINT   Maximum number of mesh points involved in non-contiguous mesh patching boundary condition (**BCPINT** subroutine, see Section 3.7).

NBFRA   Sum of all 2-D grid block elements

LGRAFX   Graphics array size

The minimum values for each of these array limits for a given grid may be calculated as follows:

$$NRA3D \geq \sum_{i=1}^{i=NBLKS} [(IMX)_i + 1][(JMX)_i + 1][(KMX)_i + 1]$$

$$NBL2D \geq max\left([(IMX)_i + 1, (JMX)_i + 1, (KMX)_i + 1]\right)^2$$

$$NRA1D \geq \sum_{i=1}^{i=NBLKS} max[(IMX)_i + 1), (JMX)_i + 1, (KMX)_i + 1]$$

23

$$NBFRA \geq \sum_{i=1}^{i=NBLKS} [(IMX)_i + 1][(JMX)_i + 1]L_{2D}(i)$$

where $(IMX)_i$, $(JMX)_i$, and $(KMX)_i$ indicate the size of grid block i, $L_{2D}(i)$ is a trigger to indicate whether the grid block i is 2-D (1) or 3-D (0), and $NBLKS$ is the total number of grid blocks. The limits on **NRA1D** and **NBL2D** utilize the largest of all the dimensions of each of the individual grid blocks to eliminate any restrictions on the size of an individual grid block. PARAMETER **NBL2D** is used for those two-dimensional arrays which are used repeatedly for every grid block, and must therefore be dimensioned large enough to contain the largest 2-D grid block. PARAMETER **NBCPBL** determines array limits for storing boundary condition specifications by setting the maximum number of boundary conditions per block. The total number of boundary conditions which may be specified for a given calculation is then limited by the product **NBMAX * NBCPBL**. The PARAMETER **NRAINT** is used to provide extra storage for the application of the non-contiguous mesh patching scheme **BCPINT**. The **BCPINT** subroutine utilizes an interpolation scheme to effect block to block communication for meshes which have a common mating surface defined by non-contiguous mesh points. The PARAMETER **NRAINT** must be dimensioned large enough to contain all mesh points involved in the non-contiguous mating surfaces for all mesh blocks (more details are available in the description of **PINT** in Section 3.7). The PARAMETER **NBFRA** is utilized to store blockage and body force source terms for 2-D flow simulations with embedded blade rows, and must be large enough to store all of these elements for all 2-D mesh blocks. The PARAMETER **LGRAFX** is used to define storage for graphics arrays used in the real time interactive graphics display described in Chapter 9. If graphics is not used, the **LGRAFX** should be set to 1. If graphics is desired, then **LGRAFX** should be set equal to the value used for **NRA3D**, above.

The requirement that the PARAMETER variables **NRA3D, NBL2D, NRA1D** be based on array sizes 1 element larger than the grid dimensions results from the use of phantom points outside the computational domain to impose the numerical boundary conditions. The *ADPAC-AOACR* program automatically checks to make sure enough storage is available for all the blocks and issues a fatal error message if the array size is exceeded.

Approximate computational storage and CPU requirements can be estimated for the aerodynamic analysis from formulas listed in Section 3.2.

3.4 ADPAC Compilation Using Makefile

Compilation of the *ADPAC-AOACR* source code into an executable form is handled through a UNIX-based Makefile facility. A Makefile is included with the standard distribution which permits automatic compilation of the code for several operational capabilities and computer systems. The format of the *Makefile* compiling command is described below.

In the directory containing the FORTRAN source of the *ADPAC-AOACR* code, compilation is performed by executing the command:

make *option*

The **make** command is standard on UNIX systems and automatically interrogates the file *Makefile* for instructions on how to perform the compilation. The *option* argument may be any of the variables listed below:

No argument - same as *link* below.

link   This is the standard UNIX system compilation. All non-standard programming constructs are avoided (such as graphics, or multi-processor features). This option will deliver a working executable on most UNIX systems which support standard naming conventions (*f77* as the standard compiler, etc.). The compilation includes basic compiler optimization (f77 -O).

pfa   This option is used on Silicon Graphics computers supporting the Power FORTRAN option for multiprocessor execution. The compilation includes basic compiler optimization (f77 -O).

graphics   This option compiles *ADPAC-AOACR* with the necessary routines needed to permit interactive graphics between network connected Silicon Graphics workstations. This option will only work when compiling on a Silicon Graphics workstation with IRIX operating system 4.0.1 or above. The full Silicon Graphics shared graphics libraries and X-windows system graphics libraries must be installed on the compiling workstation in order for this option to work.

pfagraphics   This option combines the features of *pfa* and *graphics* described above.

cray   This option is utilized when compiling the standard code on a Cray computer. For best performance, the aggressive optimization option of the Cray compiler has been invoked (cf77 -Zv -Wf"-o aggress").

aix   This option is used when compiling the standard code on an IBM RS-6000 workstation running the AIX operating system.

craygraphics   This option compiles *ADPAC-AOACR* with the necessary routines needed to permit execution on a remote, network connected Cray computer with interactive graphics displayed on a local Silicon Graphics workstation. This option requires the compilation of the Cray Graphics Libraries (CGL) (developed at NASA-Ames and included in the source code), and that the program *AGTPLT-LCL* be running on the local workstation during execution (see Chapter 9.0 on Graphics for more details).

aixgraphics   This option compiles *ADPAC-AOACR* with the necessary routines needed to permit execution on a remote, network connected IBM workstation computer running the AIX operating system with interactive graphics displayed on a local Silicon Graphics workstation. This option requires the compilation of the Cray Graphics Libraries (CGL) (developed at NASA-Ames and included with the source code), and that the program *AGTPLT-LCL* be running on the local workstation during execution (see Chapter 9.0 on Graphics for more details).

sgidbx   This option is used for generating an executable version of the code which is compatible with the standard UNIX *dbx*-based debugging facility. This should work on any standard UNIX machine which supports *dbx* (Note: the code will run much more slowly when compiled in this fashion.) This option is used mainly for code development or debugging.

craydbx   This option is used for generating an executable version of the code which is compatible with the Cray *cdbx* debugging facility. (Note: the code will run much more slowly when compiled in this fashion.) This option is used mainly for code development or debugging.

aixdbx   This option is used for generating an executable version of the code which is compatible with the IBM AIX *dbx* debugging facility. (Note: the code will run

much more slowly when compiled in this fashion.) This option is used mainly for code development or debugging.

At the completion of the compilation process on any system, an executable version of the code, named simply **adpac** is written in the source directory (see Appendix A for an application of the compilation and execution processes for a sample test case).

3.5   Input/Output Files

In this section, the various input/output data files related to a calculation using the *ADPAC-AOACR* program are described. In order to understand the file naming convention, the concept of a case name must first be detailed. All files used in an *ADPAC-AOACR* calculation are named according to a standard naming convention of the form:

case.extension

where *case* is a unique, user-specifiable name identifying the geometry or flow condition being investigated, and *extension* is a name describing the type of file. The *case* name must be specified in the standard input file described below. A list and description of each of the files used or generated by *ADPAC-AOACR* is given in Table 3.1.

The standard input, standard output, boundary data, and convergence history files are stored in ASCII format. All other files utilize the Scientific DataBase Library (SDBLIB) [10] format. The mesh file and *PLOT3D* plot output files are compatible with the *PLOT3D* multiple grid, binary definition (see Sections 3.8 and 3.11 for a description and coding examples of the SDBLIB binary format).

The standard input and standard output files are directed at runtime using the standard UNIX redirection syntax as:

adpac < inputfile > outputfile

ADPAC-AOACR Coordinate System Reference
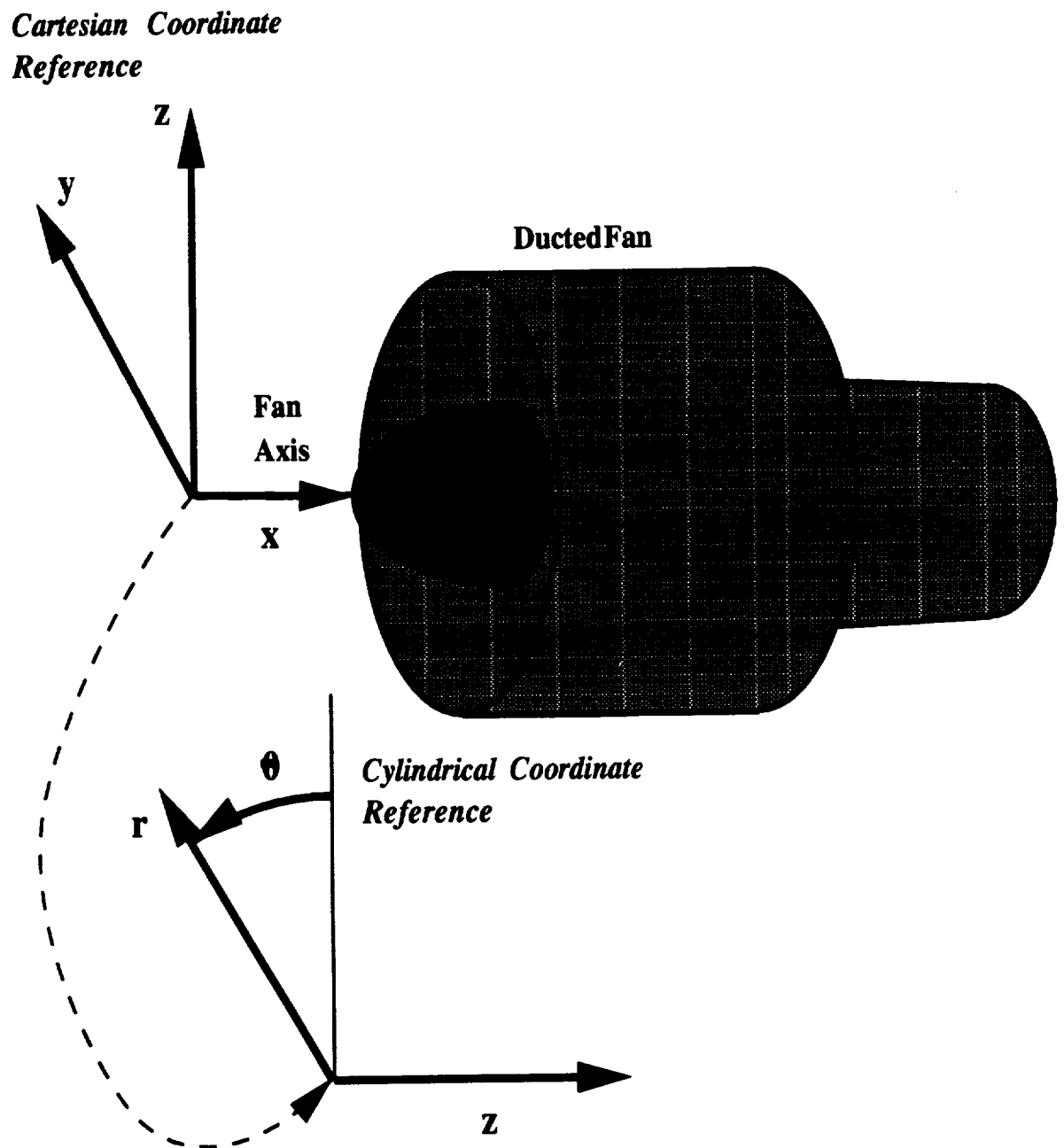
Figure 3.1: *ADPAC-AOACR* Geometry Coordinate Reference

Table 3.1: Description of input/output files and UNIX-based filenames for *ADPAC-AOACR* Euler/Navier-Stokes solver

Name	Description
case.input	Standard input file
case.boundata	Block boundary definition file
case.output	Standard output file
case.mesh	Mesh file (*PLOT3D* compatible)
case.p3dabs	Final *PLOT3D* output file (absolute flow)
case.p3drel	Final *PLOT3D* output file (relative flow)
case.bf.#	2-D blockage/body force file for block number #
case.p3fr.#	Instantaneous *PLOT3D* interval output file (absolute flow) The frame number is given by #.
case.img.#	Instantaneous Silicon Graphics image file for graphics interactive display) The frame number is given by #.
case.restart.new	New restart file (output by code)
case.restart.old	Old restart file (used as input for restart runs)
case.converge	Solution residual convergence history file

If a restart run is desired, the user must move the most current output restart file from

<p style="text-align:center">case.restart.new</p>

to the default input restart file name

<p style="text-align:center">case.restart.old</p>

each time the code is restarted. A more detailed description of the use and format of the *ADPAC-AOACR* files is presented in the sections which follow.

3.6 Standard Input File Description

The standard *ADPAC-AOACR* input file contains a number of user-specifiable parameters which control the basic operation of the code during execution. These parameters tend to be less case dependent (as opposed to the boundary data file

which is entirely case dependent). During code execution, the input file is read one line at a time as a character string, and each string is parsed sequentially to determine the specific program action in each case. The standard input file utilizes a keyword input format, such that any line which does not contain a recognizable keyword is treated as a comment line. Therefore, the user may place any number of comments in the file (so long as the line does not contain a keyword input string in the form described below), and code execution is unaltered. Comments may also be placed after the variable assigned to the keyword as long as there are one or more blanks separating the keyword value from the comment string. All input file lines are echoed to the standard output, and the program response to each line is listed when a specific action is taken.

All keyword input lines are given in the following format:

        KEYWORD = Value   Comment String

where KEYWORD is one of the recognized keywords described below, and Value is the specific value to be assigned to that variable. The input line must contain the equals sign (=) with one or more blanks on both sides in order to be recognized. The Comment String must be separated by one or more blank spaces from the Value. Therefore, the lines

```
DIAM    = 10.000
DIAM                        =               10.000
DIAM    = 10.000    This is the diameter.
```

are valid keyword input lines assigning the value 10.0 to the variable associated with the keyword DIAM. Conversely, the lines

```
DIAM= 10.000
DIAM =10.000
DIAM=10.000
```

are not recognizable keyword input lines (in spite of the presence of the keyword DIAM), because of the lack of proper placement of the blanks about the equals sign. The purpose for this restriction is to permit keyword variables in comment lines,

and to help users to generate readable input files. All keyword values are either real numbers (which, in many cases, are converted to integers in the code) or character strings.

A sample *ADPAC-AOACR* standard input file containing all possible keywords is listed below:

ADPAC-AOACR Sample Standard Input File

```
----------------------------------------------------
ADPAC Input File Generated by SETUP-Version 1.0
Apr 15 1992, at 17:36:38
----------------------------------------------------
                      JOB TITLE
----------------------------------------------------
NASA 1.15 PRESSURE RATIO FAN - 2 BLADE ROWS
MULTIGRID INTERFACE AVERAGING CALCULATION
----------------------------------------------------
                     INPUT DATA
----------------------------------------------------
VARNAME  = VARIABLE VALUE     COMMENT
----------------------------------------------------
    CASENAME   = nasa            The case name is "nasa"
    FMULTI     = 3.0             Three mesh levels for multigrid
    FSUBIT     = 1.0             1 subiteration on each coarse mesh level
    FFULMG     = 1.0             Use "full" multigrid
    FCOAG1     = 3.0             Start "full" multigrid on 3rd mesh level
    FCOAG2     = 2.0             End "full" multigrid on 2nd mesh level
    FITFMG     = 150.0           150 "full" multigrid iterations
    RMACH      =     0.750000    Reference Mach Number
    FINVVI     =     0.000000    Inviscid Flow
    GAMMA      =     1.400000    Specific heat ratio
    PREF       =  2116.800049    Reference Total Pressure (lbf/ft**2)
    TREF       =   518.700012    Reference Total Temperature (Deg. R)
    RGAS       =  1716.260010    Gas constant (ft-lbf/slug-deg R)
    DIAM       =     9.000000    Reference diameter (ft.)
    EPSX       =     1.500000    Residual smoothing in i direction
```

31

```
EPSY        =     1.500000    Residual smoothing in j direction
EPSZ        =     1.500000    Residual smoothing in k direction
VIS2        =     0.500000    Fine mesh 2nd order dissipation coefficient
VIS4        =     0.015625    Fine mesh 4th order dissipation coefficient
VISCG2      =     0.125000    Coarse mesh dissipation coefficient
CFL         =    -5.000000    Steady flow, CFL=5.0
FNCMAX      =   150.000000    150 iterations on fine mesh level
FTIMEI      =     1.000000    Update time step every 1 iteration
FTURBI      =     1.000000    Update turbulence model every 1 iteration
FTURBB      = 99999.000000    Begin turbulence model at iteration 99999
PRNO        =     0.700000    Gas Prandtl number = 0.7
PRTNO       =     0.900000    Turbulent Prandtl number = 0.9
FSOLVE      =     0.000000    Use solution scheme #1
FFILT       =     5.000000    Dissipation is turned on
FRESID      =     4.000000    Residual smoothing is turned on
FREST       =     0.000000    No restart file is read in
FUNINT      = 99999.000000    Intermediate PLOT3D file each 99999 it.
FRDMUL      =     0.0         Calculate multigrid boundary conditions
FITCHK      =   100.0         100 iterations between job checkpoint output
FTOTSM      =     0.0         No post multigrid global smoothing
EPSTOT      =     0.0         Post multigrid global smoothing coefficient
P3DPRT      =     1.0         Output PLOT3D files
FVTSFAC     =     2.5
FGRAFIX     =     0.0
FGRAFINT    =     0.0
FIMGSAV     =     0.0
FIMGINT     = 99999.0
WBF(1)      =     0.0
WBF(2)      =     0.0
WBF(3)      =     0.0
WBF(4)      =     0.0
L2D(1)      =     0.0
L2D(2)      =     0.0
L2D(3)      =     0.0
L2D(4)      =     0.0
ADVR(1)     =    -2.780000    Advance ratio for block #1 is -2.78
ADVR(2)     =    -2.780000    Advance ratio for block #2 is -2.78
ADVR(3)     =     0.000000    Advance ratio for block #3 is  0.00
ADVR(4)     =     0.000000    Advance ratio for block #4 is  0.00
```

`ENDINPUT`

It is unnecessary to specify all possible keywords in every input file. The *ADPAC-AOACR* code is programmed with a default set of input variables such that the only input variable which must be present is the **CASENAME** (described below) which is used to assign input/output file names. A list and description of all input keywords and their default values are listed below. A quick reference to the input file keywords is provided in Appendix B.

ADPAC-AOACR Standard Input File Keyword Description

Keyword: CASENAME
Format: (No Default Value)

> `CASENAME = case`

> Description: The **CASENAME** keyword value is used to set the case name which is used to define all input/output file names during an *ADPAC-AOACR* run (see Section 3.5 for details). The case name is limited to an 8 character string, and cannot contain embedded blanks. The case name has no default value, and as such, all input files must contain the **CASENAME** keyword.

Keyword: RMACH
Format: (Default Value = 0.5)

> `RMACH = 0.5`

> Description: The **RMACH** keyword value is intended to be a reference flow Mach number, used primarily to set the initial freestream values for a given calculation. The freestream values are used to initialize the flowfield prior to the execution of the time-marching solver in the absence of a restart file. It should be mentioned

that the initial data values are assigned based on the assumption that the nondimensional freestream total pressure and total temperature are 1.0. This implies that it is advisable to set up all input variables (in particular PREF and TREF, described below, and PEXIT, described in Section 3.7 on boundary data file specifications) such that the imposed inlet and exit flow boundary conditions are compatible with the initial conditions derived from RMACH, based on the assumed global nondimensional total pressure and temperature. For example, suppose that the desired solution for an internal stage compressor rotor has an inlet total pressure of 24 psia, and an exit static pressure of 23.5 psia. For compressor designers, these numbers would commonly be referenced to standard atmospheric pressure (14.7 psia), resulting in nondimensional upstream total and exit static pressures of 1.6326 and 1.5986, respectively. If RMACH is set to 0.5, and the reference pressure is 14.7 psia, then the interior mesh points will be initiated with a static pressure value of 0.84302. It is unlikely that a stable solution will result when the exit static pressure is 1.5986, and the interior static pressure is 0.84302 (reversed flow at the exit boundary will result). A better approach is to specify 24 psia as the reference pressure, such that the nondimensional inlet total and exit static pressures are 1.0, and 0.97917, and the initial nondimensional static pressure at the interior cells is 0.84302. With these values, it is much more likely that a stable solution will result. In addition, the value of **RMACH** is used in conjunction with the value of advance ratio specified by the keyword **ADVR**, when the rotational speed is defined in this manner. In this case, the value of **RMACH** must be the freestream Mach number associated with the advance ratio specified by **ADVR** or an incorrect rotational speed will be calculated.

Keyword: FINVVI

Format:(Default Value = 1.0)

> FINVVI = 1.0

Description: The **FINVVI** keyword is a simple trigger to determine whether the solution mode is for inviscid flow (**FINVVI** = 0.0) or for viscous flow (**FINVVI** = 1.0). This trigger controls whether the viscous stress flux contributions are calculated during the time-marching process. This does not affect the application of boundary conditions, as this is completely controlled by the specifications in the boundary data file (see Section 3.7). As such, it is possible to run viscous boundary conditions in an inviscid flow solution, and inviscid boundary conditions in a viscous flow solution.

Keyword: GAMMA
Format:(Default Value = 1.4)

> GAMMA = 1.4

Description: The **GAMMA** keyword sets the value for the gas specific heat rato. For most cases involving air at moderate pressures and temperatures, a value of 1.4 is adequate. For cases involving combustion products, this value may be quite different, and should be considered appropriately. Extreme care must be taken when post-processing a calculation which is based on a value of **GAMMA** other than 1.4 as many post processors use an assumed value of the specific heat ratio equal to 1.4 (*PLOT3D* is a common example). It should be mentioned that the present version of the code does not permit user specification of the fluid viscosity, as the formula for air is hardwired into the code.

Keyword: PREF
Format:(Default Value = 2116.0)

> PREF = 2116.0

Description: The **PREF** keyword sets the dimensional value (in pounds force per square foot) of the reference total pressure used to nondimensionalize the flowfield. For viscous flows, this value must be accurately specified in order to properly set the nondimensional flow viscosity, (and hence, the Reynolds number). For inviscid flow predictions, this value has no real significance because of the similarity of inviscid flows with Mach number. It is very important to choose an average representative value for this variable, such that the nondimensional total pressure at any point in the flow is near a value of 1.0. An extended discussion on the reason for this choice is given in the description of **RMACH**, above.

Keyword: TREF
Format:(Default Value = 518.7)

```
TREF = 518.7
```

Description: The **TREF** keyword sets the dimensional value (in degrees Rankine) of the reference total temperature used to nondimensionalize the flowfield. For viscous flows, this value must be accurately specified in order to properly set the nondimensional flow viscosity, (and hence, the Reynolds number). This value is also important for the specification of wall temperature used in the viscous wall boundary condition **SSVI, SS2DVI** (see the description of the boundary data file, Section 3.7). For inviscid flow predictions, this value has no real significance because of the similarity of inviscid flows with Mach number. It is very important to choose an average representative value for this variable, such that the nondimensional total temperature at any point in the flow is near a value of 1.0. An extended discussion on the reason for this choice is given in the description of **RMACH**, above.

Keyword: RGAS
Format:(Default Value = 1716.26)

36

```
RGAS = 1716.26
```

Description: The **RGAS** keyword sets the dimensional value (in foot-pounds force per slug-degree Rankine) of the gas constant. The default value is for atmospheric air at standard pressure and temperature. This value is used in conjunction with **GAMMA** above in determining the gas specific heats at constant pressure and constant volume.

Keyword: DIAM
Format:(Default Value = 1.0)

```
DIAM = 1.0
```

Description: The **DIAM** keyword sets the dimensional value (in feet) of the reference diameter. This value should be equal to the value used to nondimensionalize the mesh coordinates (see Section 3.8). This value is used in determining the flow Reynolds number and the flow nondimensional rotational speed, and many problems can be traced to improperly specifying this value.

Keyword: EPSX, EPSY, EPSZ
Format:(Default Value = 1.0)

```
EPSX = 1.0
EPSY = 1.0
EPSZ = 1.0
```

Description: The **EPSX, EPSY, EPSZ** keywords set the value of the implicit residual smoothing coefficient multipliers in the i, j, and k coordinate directions, respectively. The values of **EPSX, EPSY,** and **EPSZ** are used as simple multipliers for the residual smoothing coefficients calculated by the eigenvalue scaling residual smoothing scheme described in the Final Report [1]. If **EPSX, EPSY** or **EPSZ** = 0.0, then no smoothing is applied for the given coordinate direction. The default value is 1.0. Any value larger than 1.0

simply implies excess smoothing and may be useful for cases with poor convergence or undesirable mesh quality. If a value larger than 3.0 is required to stabilize a solution, this generally indicates some sort of problem in the calculation (such as poor mesh aspect ratio, bad boundary specification, etc.). Values less than 1.0 will likely cause code instabilities.

Keyword: VIS2
Format:(Default Value = 1/2)

```
VIS2 = 0.5
```

Description: The **VIS2** keyword defines the value of the second order added dissipation term used in the fine mesh time-marching solver (see the Final Report, reference [1]). This value is a simple multiplier of the second order dissipation term, and hence, larger values imply more added dissipation. The recommended value is 0.5, but values from 0.0 (no second order dissipation) to 2.0 may be necessary. Any value larger than 2.0 is of questionable use, as the added dissipation will likely dominate the solution.

Keyword: VIS4
Format:(Default Value = 1/64)

```
VIS4 = 0.015625
```

Description: The **VIS4** keyword defines the value of the fourth order added dissipation term used in the fine mesh time-marching solver (see the Final Report, reference [1]). This value is a simple multiplier of the fourth order dissipation term, and hence, larger values imply more added dissipation. The recommended value is 0.015625 (1/64), but values from 0.0 (no fourth order dissipation) to 0.0625 (1/16) may be necessary. Any value larger than 0.0625 is of questionable use, as the added dissipation will likely dominate the solution.

Keyword: CFL
Format:(Default Value = -5.0)

```
CFL = -5.0
```

Description: The **CFL** keyword defines the value of the time step multiplier used in the time-marching solver. The algorithm is sensitive to the sign of the value used for **CFL** in determining the manner in which the time-marching solver is applied. If **CFL** < 0.0, local time stepping is used (steady flow only) and each cell is advanced in time according to it's own maximum allowable time step. If **CFL** > 0.0, then a time-accurate time-marching algorithm is applied, and the code calculates the smallest of all calculated maximum time steps and applies this value uniformly in the time-marching scheme at each cell. The absolute value of **CFL** is used as a multiplier for the time step (larger absolute values indicate larger time steps). A value of -5.0 is used for steady flow calculations, and a value of 7.0 is recommended for time-accurate calculations. The value of **CFL** is also used implicitly in the eigenvalue scaling terms in the implicit residual smoothing algorithm, such that larger values of **CFL** imply increased residual smoothing (see the description of the implicit residual smoothing algorithm in the companion Final Report [1]).

Keyword: FNCMAX
Format:(Default Value = 10.0)

```
FNCMAX = 200.0
```

Description: The **FNCMAX** keyword controls the total number of iterations for a non-multigrid calculation (**FMULTI** $\leq$ 1.0), or the number of global iterations on the finest mesh for a multigrid calculation (**FMULTI** > 1.0). The total number of iterations performed on all meshes for a multigrid run is controlled by a combination

of **FNCMAX, FMULTI, FCOAG1, FCOAG2, FFULMG,
FITFMG,** and **FSUBIT**. For example, the values

```
FNCMAX = 200.0
FMULTI = 1.0
FITFMG = 0.0
FFULMG = 0.0
FSUBIT = 0.0
```

would prescribe 200 iterations of a non-multigrid run (only the fine mesh is used). The values

```
FNCMAX = 200.0
FMULTI = 3.0
FITFMG = 0.0
FFULMG = 0.0
FSUBIT = 1.0
```

would prescribe 200 multigrid iterations using 3 mesh levels (but still only 200 global iterations, where each iteration involves a single subiteration on each of 3 mesh levels). And finally, the values

```
FNCMAX = 200.0
FMULTI = 3.0
FITFMG = 50.0
FFULMG = 1.0
FSUBIT = 1.0
FCOAG1 = 3
FCOAG2 = 2
```

would prescribe an initial pass of 50 iterations on the third mesh level, followed by 50 multigrid iterations on the second mesh level, and finally 200 global multigrid iterations on the finest mesh level. See the descriptions of the variables **FNCMAX, FMULTI, FCOAG1, FCOAG2, FFULMG, FITFMG,** and **FSUBIT** for further details. A flowchart of the *ADPAC-AOACR* iteration and multigrid control algorithm is given in Figure 3.2.

40

ADPAC–AOACR Input Keyword Multigrid Cycle and Time–Marching Iteration Management Flowchart

Figure 3.2: *ADPAC-AOACR* Input Keyword Multigrid Cycle and Time-Marching Iteration Management Flowchart

41

Keyword: FMULTI

Format:(Default Value = 1.0)

```
FMULTI = 1.0
```

Description: The **FMULTI** keyword assigns the number of multigrid levels to be used during the calculation (for a description of the multigrid algorithm, see Reference [1]). The code will analyze the dimensions of the fine mesh to determine whether it can be properly subdivided according to the number of multigrid levels requested. If **FMULTI** $\leq$ 1.0, then the number of multigrid levels is set to 1, and the calculation is performed on the finest mesh only without multigrid acceleration. For unsteady flows, multigrid is not valid, and **FMULTI** should be set to 1.0. A flowchart of the *ADPAC-AOACR* iteration and multigrid control algorithm is given in Figure 3.2.

Keyword: FFULMG

Format:(Default Value = 0.0)

```
FFULMG = 0.0
```

Description: The **FFULMG** keyword assigns a trigger which determines whether the "full" multigrid solution procedure is applied or whether the standard multigrid procedure is used. The use of "full" multigrid is advisable (but not required) when a new calculation is being started as a means of rapidly generating a better initial guess for the final flowfield. If the solution is being restarted from a previous calculation (FREST=1.0), it is usually advisable to set **FFULMG** to 0.0 to avoid destroying the initial data read from the restart file (a warning message is issued when this combination is specified). A flowchart of the *ADPAC-AOACR* iteration and multigrid control algorithm is given in Figure 3.2.

Keyword: FTIMEI

Format:(Default Value = 1.0)

```
FTIMEI = 1.0
```

Description: The **FTIMEI** keyword assigns a trigger which determines the number of iterations between time step evaluations. For best results, this should be 1.0, which implies that the time step is reevaluated at every iteration. However, this value can be increased (< 10) to reduce CPU time by reevaluating the time step every **FTIMEI** iterations instead.

Keyword: FTURBI
Format:(Default Value = 1.0)

```
FTURBI = 0.0
```

Description: The **FTURBI** keyword assigns a trigger which determines the number of iterations between turbulence model evaluations. For best results, this should be 1.0, which implies that the turbulence parameters are reevaluated at every iteration. However, this value can be increased (< 10) to reduce CPU time by reevaluating the turbulence quantities every **FTURBI** iterations instead.

Keyword: FTURBB
Format:(Default Value = 50.0)

```
FTURBB = 50.0
```

Description: The **FTURBB** keyword assigns a trigger which determines the number of iterations before the turbulence model is activated. For laminar flow, set **FTURBB** to a very large number (**FTURBB > FNCMAX + (FMULTI-1) * FITFMG) * FFULMG**) so the turbulence model is never called. For turbulent flow, the value should be large enough (e.g., ≥ 50) to ensure that the solution has developed adequately enough to permit stable implementation of the turbulence model (i.e. - the flowfield should at least exhibit the gross characteristics (correct flow direction, some boundary

43

layer development) of the expected final flow before the turbulence model is activated).

Keyword: PRNO
Format:(Default Value = 0.7)

```
PRNO = 0.7
```

Description: The **PRNO** keyword assigns the value of the gas Prandtl number. For air (and many other gases) at moderate pressures and temperatures, a value of 0.7 is appropriate.

Keyword: PRTNO
Format:(Default Value = 0.9)

```
PRTNO = 0.9
```

Description: The **PRTNO** keyword assigns the value of the gas turbulent Prandtl number. The turbulence model in *ADPAC-AOACR* determines the turbulent thermal conductivity via a turbulent Prandtl number and the calculated turbulent viscosity (see the Final Report, Reference [1]). The recommended value is 0.9.

Keyword: FSOLVE
Format:(Default Value = 0.0)

```
FSOLVE = 0.0
```

Description: The **FSOLVE** keyword assigns a trigger which determines which type of time-marching strategy is employed on both fine and coarse meshes. For **FSOLVE** = 0.0, the standard 4-stage Runge Kutta time-marching scheme is used with a single added dissipation evaluation, and implicit residual smoothing at alternating stages. For **FSOLVE** = 1.0, a modified 4-stage Runge Kutta time-marching scheme is used with a two evaluations of the added dissipation, and implicit residual smoothing at every stage. For

FSOLVE = 2.0, a 5-stage Runge Kutta time-marching scheme is used with three weighted added dissipation evaluations, and implicit residual smoothing at every stage. **FSOLVE = 1.0** is the recommended value, although the other schemes may provide improved convergence at a different computational cost.

Keyword: FFILT

Format:(Default Value = 1.0)

```
FFILT = 1.0
```

Description: The **FFILT** keyword assigns a trigger which determines directly whether the added dissipation routines are called during the time-marching process. If **FFILT = 0.0**, then no added dissipation is calculated. It is also possible to turn off the added dissipation by setting the values of **VIS2** and **VIS4** to 0.0; however, the use of **FFILT** avoids the calculation of the dissipation terms entirely. It is unlikely that any value other than 1.0 is required except for code debugging purposes.

Keyword: FRESID

Format:(Default Value = 1.0)

```
FRESID = 1.0
```

Description: The **FRESID** keyword assigns a trigger which determines directly whether the implicit residual smoothing routines are called during the time-marching process. If **FFILT = 0.0**, then no added dissipation is calculated. It is also possible to turn off the residual smoothing by setting the values of **EPSX**, **EPSY**, and **EPSZ** to 0.0; however, the use of **FRESID** avoids the calculation of the smoothed residuals entirely. It is unlikely that any value other than 1.0 is required except for code debugging purposes.

Keyword: FREST

Format:(Default Value = 0.0)

```
FREST = 0.0
```

Description: The **FREST** keyword assigns a trigger which controls the restart characteristics of the *ADPAC-AOACR* code. If **FREST** = 0.0, then no restart file is used, and the flow variables are initialized according to the scheme described by the input keyword **RMACH**. If **FREST** = 1.0, then the code attempts to open a restart file described by the file naming convention (see Section 3.5), and the flow variables are initialized by the values given in the restart file. Restarting a calculation from a previous calculation is often useful for breaking up large calculations into smaller computational pieces, and may also provide faster convergence for cases which involve minor changes to a previous calculation.

Keyword: FUNINT

Format:(Default Value = 99999.0)

```
FUNINT = 99999.0
```

Description: The **FUNINT** keyword is used to determine the number of iterations between instantaneous *PLOT3D* absolute flow file output. For a time-dependent calculation, it is often desirable to print out data at several intervals during the course of the solution to examine the time-dependent nature of the flow. The *ADPAC-AOACR* program provides a mechanism by which a *PLOT3D* format flow file can be printed at a fixed iteration interval (the interval defined by the value of **FUNINT**) as a means of extracting time-dependent data during a calculation. For steady flow calculations, it is normally desirable to set **FUNINT** to a very large number, and simply use the final output *PLOT3D* format files if needed. For unsteady flow calculations, the value of **FUNINT** can be highly case dependent, and some numerical experimentation may be required to prevent excessive output, or a deficiency in data. The file naming convention for the unsteady output files is given in Section 3.5.

Keyword: FSUBIT
Format:(Default Value = 1.0)

```
FSUBIT = 1.0
```

Description: The **FSUBIT** keyword determines the number of subiterations performed on coarse meshes during the coarse mesh portion of the multigrid cycle. As such, this variable is actually only used when **FMULTI** > 1.0. Additional subiterations on coarse meshes during the multigrid cycle can often improve convergence, at the expense of some additional computational work. The number of subiterations specified by **FSUBIT** is applied at each coarse mesh level during the multigrid calculations process. A value of 1.0 or 2.0 is typically best. A complete description of the multigrid calculation process is given in the Final Report [1]. A flowchart of the *ADPAC-AOACR* iteration and multigrid control algorithm is given in Figure 3.2.

Keyword: FCOAG1
Format:(Default Value = 1.0)

```
FCOAG1 = 1.0
```

Description: The **FCOAG1** keyword specifies the initial, or coarsest coarse mesh level upon which the "full" multigrid calculation is initially applied. When multiple coarse mesh levels are available for processing, it is occasionally useful to specify the initial coarse mesh level in the "full" multigrid sequence in order to avoid wasted computations on lower mesh levels. Typically, **FCOAG1** is set to **FMULTI** (start "full" multigrid on coarsest mesh level). In some cases (when **FMULTI** is larger than 3.0) it may be advisable to set **FCOAG1** to 3.0, and avoid useless processing on coarser meshes during the "full" multigrid startup process. A flowchart of the *ADPAC-AOACR* iteration and multigrid control algorithm

47

is given in Figure 3.2. An example is given in the description of
FCOAG2 below.

Keyword: FCOAG2

Format:(Default Value = 1.0)

        FCOAG2 = 1.0

Description: The **FCOAG2** keyword specifies the final, or finest coarse mesh
level upon which the "full" multigrid calculation is applied. When
multiple coarse mesh levels are available for processing, it is occa-
sionally useful to specify the final coarse mesh level in the "full"
multigrid sequence in order to examine the flowfield without ac-
tually performing any calculations on the fine mesh. For example,
the combination

        FMULTI = 3.0
        FCOAG1 = 3.0
        FCOAG2 = 3.0
        FNCMAX = 10.0
        FITFMG = 100.0

would direct a "full" multigrid startup of 100 iterations on mesh
level 3, and since **FCOAG2**=3.0, the "full" multigrid sequence
is ended at this mesh level. The solution is then interpolated
to the fine mesh, and then 10 fine mesh iterations using 3 levels
of multigrid would be performed. Typically, **FCOAG1** is set to
2.0, which indicates that the "full" multigrid startup procedure
utilizes all mesh levels from **FCOAG1** to 2 before starting any
processing on the fine mesh. A flowchart of the *ADPAC-AOACR*
iteration and multigrid control algorithm is given in Figure 3.2.

Keyword: FITFMG

Format:(Default Value = 100.0)

        FITFMG = 100.0

Description: The **FITFMG** keyword dictates the number of iterations to be performed on each of the coarse mesh levels during a "full" multigrid startup sequence. Typically, the startup sequence is used only to generate a reasonable initial guess for the fine mesh, so the value of **FITFMG** is kept relatively low (≈ 100). The function of the keyword **FITFMG** is illustrated graphically in Figure 3.2.

Keyword: FRDMUL

Format:(Default Value = 0.0)

```
FRDMUL = 0.0
```

Description: The **FRDMUL** keyword determines whether boundary condition data for the coarse mesh levels of a multigrid run are generated from the fine mesh boundary conditions specified in the *ADPAC-AOACR* boundary data file (**FRDMUL** = 0.0), or whether the coarse mesh boundary specifications are actually read in from the boundary data file (**FRDMUL** = 1.0). In most cases, **FRD-MUL** should be set to 0.0, and the program will determine the equivalent coarse mesh boundary conditions from the fine mesh specifications. For the purposes of code debugging, or to permit multigrid calculation on a mesh which does not possess perfect "multigrid" boundary segments (a boundary condition for the fine mesh does not begin or end at a mesh index which is compatible with the multigrid sequence), it is possible to "fool" the program into running multigrid by artificially specifying an equivalent coarse mesh boundary condition.

Keyword: FITCHK

Format:(Default Value = 100.0)

```
FITCHK = 100.0
```

Description: The **FITCHK** keyword controls the number of iterations between job checkpointing in the *ADPAC-AOACR* program. Job checkpointing refers to the process of periodically saving the flowfield information to minimize the loss of data in the event that the job does not terminate normally. As a safety feature, the *ADPAC-AOACR* program writes out an updated restart file every **FITCHK** iterations in case the job stops before the final restart file output procedures are completed. It is not necessary to write out intermediate restart files, but this is considered a good precaution against unexpected problems such as computer failures, or system administration quotas. A good interval for checkpointing is 100 iterations (**FITCHK** = 100.0). The intermediate restart files, as well as the final restart file, are all written to the same file name, and therefore previous checkpoints cannot be retrieved when the file is overwritten (see Section 3.5 for restart file naming conventions). Job checkpointing only applies to the iterative cycles involving the fine mesh, and does not apply to the coarse mesh iterations calculated during a "full" multigrid startup (see **FFULMG**, above).

Keyword: VISCG2

Format:(Default Value = 1/8)

```
VISCG2 = 0.125
```

Description: The **VISCG2** keyword controls the value of the second order added dissipation coefficient for coarse mesh subiterations during the multigrid time-marching solution process. Coarse mesh subiterations utilize a simpler dissipation scheme than the fine mesh time-marching scheme, and therefore, a different damping constant is required. Larger values imply increased added dissipation. The recommended value is **VISCG2** = 0.125, although values from 0.0 (no dissipation) to 1.0 are possible. Values larger

50

than 1.0 are not recommended as the solution would then likely
be dominated by the dissipation.

Keyword: FGRAFIX
Format:(Default Value = 0.0)

```
FGRAFIX = 0.0
```

Description: The **FGRAFIX** keyword sets a trigger which controls the gener-
ation of the real time interactive graphics display in the *ADPAC-
AOACR* program. A value of **FGRAFIX** = 1.0 indicates that the
interactive graphics display facility is desired, while **FGRAFIX**
= 0.0 turns this option off. When functional, the graphics screen
is updated with the latest available flow data every **FGRAFINT**
iteration. Graphics images can be automatically captured on spe-
cific computer hardware every **FIMGSAV** iterations as a means
of creating flowfield animations (see Graphics Display, Chapter
9.0). In order for the graphics display to work, the code must be
compiled with either the *graphics, pfagraphics, craygraphics*, or
aixgraphics Makefile option (see Section 3.4 for a description of
the Makefile and the *ADPAC-AOACR* code compilation process).
There are also specific machine requirements for this option to
work as well (see the section on Graphics Display, Chapter 9.0).
The generation of interactive, real time graphics images increases
the overall computational cost, and can cause network overloading
in some cases due to the transmission of graphics information.

Keyword: FGRAFINT
Format:(Default Value = 1.0)

```
FGRAFINT = 1.0
```

Description: The **FGRAFINT** keyword determines the number of iterations
between flowfield display updates for the *ADPAC-AOACR* real
time graphics display system. This option is only valid when

FGRAFIX = 1.0, and is subject to a number of other restrictions for the graphics display system (see the description of input keywords **FGRAFIX** and **FIMGSAV**, and the description of the graphics display system, Chapter 9.0). The default value for **FGRAFINT** is 1.0, which indicates that the graphics display will be updated every iteration. This can cause excessive computational and network overhead, and the user should be aware of the potential problems when using the graphics display features.

Keyword: FIMGSAVE
Format:(Default Value = 0.0)

```
FIMGSAV = 0.0
```

Description: The **FIMGSAV** keyword sets a trigger which controls the Silicon Graphics computer screen image capturing facility of the real time interactive graphics display in the *ADPAC-AOACR* program. A value of **FIMGSAV** = 1.0 indicates that the graphics image capturing facility is desired, while **FIMGSAV** = 0.0 turns this option off. When the interactive graphics display option has been enabled (see details for input keywords **FGRAFIX**, **FGRAFINT** above) the graphics screen is updated with the latest available flow data every **FGRAFINT** iteration. When the image capturing facility is enabled, these graphics images can be automatically captured on specific computer hardware every **FIMGINT** iterations as a means of creating flowfield animations (see Graphics Display, Chapter 9.0).In order for the graphics display image capturing facility to work, the code must be compiled with either the *graphics*, or *pfagraphics* Makefile option (see Section 3.4 for a description of the Makefile and the *ADPAC-AOACR* code compilation process). There are also specific machine requirements for this option to work as well (see the section on Graphics Display, Chapter 9.0). The generation of interactive, real time graphics images increases

the overall computational cost, and can cause network overloading in some cases due to the transmission of graphics information. The capturing of many screen images will also require a large amount of file storage space (see Section 3.5 for a description of the image capturing file naming convention).

Keyword: FIMGINT
Format:(Default Value = 99999.0)

```
FIMGINT = 99999.0
```

Description: The **FIMGINT** keyword determines the number of iterations between flowfield graphics display image capturing available on Silicon Graphics computers for the *ADPAC-AOACR* real time graphics display system. This option is only valid when **FGRAFIX** = 1.0, and **FIMGSAV** = 1.0, and is subject to a number of other restrictions for the graphics display system (see the description of input keywords **FGRAFIX** and **FGRAFINT**, and the description of the graphics display system, Chapter 9.0). The default value for **FIMGINT** is 99999.0, which indicates that a screen image will be saved every 99999 iterations. This large value was chosen to prohibit accidental image capturing, which can quickly fill up a large amount of disk storage. The graphics display system can cause excessive computational and network overhead, and the user should be aware of the potential problems when using this feature of the *ADPAC-AOACR* code.

Keyword: FVTSFAC
Format:(Default Value = 2.5)

```
FVTSFAC = 2.5
```

Description: The **FVTSFAC** keyword determines the value of the viscous time step evaluation factor used to stabilize the time-marching solution for viscous flows. This factor is used to magnify the importance

of the diffusion-related contributions to the time step evaluation (larger values suggest larger restrictions due to diffusion related parameters). This factor is particularly useful for meshes with rapid changes in grid spacing, and the default value of 2.5 was prescribed somewhat arbitrarily following numerical experimentation. It is unlikely that this value needs modification for most cases.

Keyword: FTOTSM
Format:(Default Value = 0.0)

        FTOTSM = 0.0

Description: The **FTOTSM** keyword is used to trigger the post multigrid smoothing algorithm. In this scheme, the residual corrections from the multigrid process are combined with the fine mesh residuals and are smoothed globally using a simple constant coefficient version of the implicit residual smoothing algorithm. The smoothing coefficient is determined by the value of the input keyword variable **EPSTOT**. The scheme is disabled when **FTOTSM** has a value of 0.0, and is employed when **FTOTSM** has a value of 1.0. This scheme is not normally recommended, but can be utilized when convergence problems are encountered.

Keyword: EPSTOT
Format:(Default Value = 0.1)

        EPSTOT = 0.1

Description: The **EPSTOT** keyword determines the value of the smoothing coefficient employed in the post multigrid smoothing algorithm described by the trigger **FTOTSM**, above. This coefficient is only used when **FTOTSM** = 1.0. The value of the coefficient may be any positive number, but for most circumstances, a value between 0.0 and 0.25 is suggested (larger values imply more smoothing).

Keyword: P3DPRT
Format:(Default Value = 1.0)

```
P3DPRT = 1.0
```

Description: The **P3DPRT** keyword assigns a trigger which determines whether *PLOT3D* format output files are written at the end of a calculation. A value of **P3DPRT** = 1.0 indicates that the output files should be written. Conversely, a value of **P3DPRT** = 0.0 indicates that the *PLOT3D* format output files should not be written. The *PLOT3D* output files (see Section 3.5 for file naming conventions for output files) are useful for graphically examining the predicted flow quantities using widely available plotting software such as *PLOT3D*, *FAST*, *SURF*, etc. Occasionally, however, due to disk space limitations or simply to speed up execution, it may be desirable to eliminate this output feature, and therefore **P3DPRT** can be used to control this output.

Keyword: RPM(*NUM*)
Format:(Default Value = 0.0)

```
RPM(1) = 0.0
RPM(2) = 0.0
RPM(3) = 0.0
RPM(4) = 0.0
RPM(5) = 0.0
```

Description: The **RPM** keyword value determines the rotational speed (in revolutions per minute) of the mesh block number specified by the value *NUM*. The value of **RPM** is, by nature, a dimensional value. Block rotational speeds are, by default, zero, unless either an **RPM** or an **ADVR** keyword are specified otherwise. The user should be aware that if the mesh has not been correctly non-dimensionalized, it is then possible that an incorrect value of rotational speed would be used in the calculation

Keyword: ADVR(*NUM*)
Format:(Default Value = 0.0)

```
ADVR(1) = 0.0
ADVR(2) = 0.0
ADVR(3) = 0.0
```

Description: The **ADVR** keyword value determines the rotational speed (in terms of an advance ratio) of the mesh block number specified by the value *NUM*. Block rotational speeds are, by default, zero, unless either an **RPM** or an **ADVR** keyword are specified otherwise. The advance ratio is inherently tied to the freestream Mach number specified in the value associated with the keyword **RMACH**. If the mesh has not been correctly non-dimensionalized, or if the value of **RMACH** is incorrect, it is possible that an incorrect value of rotational speed would be specified in the calculation.

Keyword: L2D(*NUM*)
Format:(Default Value = 0.0)

```
L2D(1) = 0.0
L2D(2) = 0.0
L2D(3) = 0.0
L2D(4) = 0.0
L2D(5) = 1.0
```

Description: The **L2D** keyword value determines whether a particular mesh block is solved as a fully three-dimensional block, or whether it should be reduced to a two-dimensional block for the purposes of the current calculation. When **L2D(NUM)** is 0.0, block **NUM** is solved in the standard 3-D manner. When **L2D(NUM)** is 1.0, block **NUM** is solved in the two-dimensional fashion described below. True two-dimensional blocks are automatically detected based on mesh size during an *ADPAC-AOACR* run, and the **L2D**

56

trigger is automatically set to the proper value. Under certain circumstances, it is possible to "collapse" a 3-D mesh circumferentially into a 2-D mesh, and solve for an essentially axisymmetric flow on the reduced mesh. This requires that the axial and radial coordinates of the grid be constant for each circumferential grid line. When the **L2D** trigger is switched on (**L2D**=1.0), the mesh block indicated by the value of **NUM** is collapsed into a 2-D block, and the calculation is performed in the usual 2-D manner. At the end of the calculation, the final data is output as if the block had been solved 3-D.

Keyword: WBF(*NUM*)

Format:(Default Value = 0.0)

```
WBF(1) = 0.0
WBF(2) = 0.0
WBF(3) = 0.0
WBF(4) = 0.0
WBF(5) = 1.0
```

Description: The **WBF** keyword value determines whether blade blockage, body force, and energy source terms are written for the block indicated by the value of **NUM**. These body forces could then be used in a later calculation involving a 2-D representation of the given blade row. The body force file created by the operation will be named according to the file naming convention described in Section 3.5.

Keyword: BFFILE(*NUM*)

Format:(Default Value = default_file_name)

```
BFFILE(1) = case.bf.b1
```

Description: The **BFFILE** keyword value determines the name of the file used to read in the data for the blade blockage and body force source

terms used to represent the effects of embedded blade rows in 2-D axisymmetric flow calculations. The file specified by **BFFILE** is used to describe the terms for the block number indicated by the value of **NUM**. Body force data files created by the *ADPAC-AOACR* program are named according to the file naming convention described in Section 3.5.

Keyword: ENDINPUT
Format:

    ENDINPUT

Description: When the *ADPAC-AOACR* program encounters the keyword **END-INPUT**, the parser which searches each line for a valid input keyword string is terminated, and no additional input file lines are parsed for input keyword values. Any lines following the **END-INPUT** statement are ignored, except when the graphics display system is in effect across a network, in which case the statements following the **ENDDATA** statement must contain two blank lines and the Internet network address of the destination display device (see Chapter 9 for a description of the Interactive Graphics Display option).

3.7 Boundary Data File Description

The *ADPAC-AOACR* boundary data file contains the user-specifiable parameters which control the application of boundary conditions on the multiple-block mesh during a time-marching solution. These boundary specifications determine the location of solid walls, input/output flow regions, and block-to-block communication paths. Prior to a detailed discussion of the actual boundary condition specifications, several boundary condition application concepts should be explained. It is important to understand how boundary conditions are applied in the *ADPAC-AOACR* finite volume solution scheme. Finite volume solution algorithms typically employ the concept of

a phantom cell to impose boundary conditions on the external faces of a mesh block. This concept is illustrated graphically for a 2-D mesh representation in Figure 3.3.

A phantom cell is a fictitious neighboring cell which is utilized in the application of boundary conditions on the outer boundaries of a mesh block. Since flow variables cannot be directly specified at a mesh surface in a finite volume solution (the flow variables are calculated and stored at cell centers, where the corners of a cell are described by the 8 surrounding mesh points), the boundary data specified in the phantom cell are utilized to control the flux condition at the cell faces of the outer boundary of the mesh block, and, in turn, satisfy a particular boundary condition. All *ADPAC-AOACR* boundary condition specifications provide data values for phantom cells to implement a particular mathematical boundary condition on the mesh.

Although boundary conditions are imposed at phantom cells in the numerical solution, the boundary specification is still most conveniently *defined* in terms of grid points, not computational cells. An illustration of the boundary specification method for *ADPAC-AOACR* is given in Figure 3.4. All boundary conditions are specified in terms of the grid points on either an i=constant, j=constant, or k=constant mesh surface. In practice, these surfaces are typically on the outer boundaries of the mesh block, but it is also possible to impose a boundary on the interior of a mesh block (see the description of the boundary specifications **KILL** and **KIL2D**, below).

The third important aspect of the application of boundary conditions in the *ADPAC-AOACR* code involves the order in which boundary conditions are applied. During the execution of the *ADPAC-AOACR* code, all boundary conditions are applied to the various mesh blocks in the order in which they are specified in the *case*.**boundata** file. As a result, it is possible to overwrite a previously specified boundary patch with a different boundary condition than was originally specified. This concept is illustrated graphically in Figure 3.5. The user must take proper precautions to prohibit accidentally overwriting a desired boundary patch as the *ADPAC-AOACR* code cannot distinguish the proper order for the user.

During code execution, the boundary data file is read one line at a time as a character string, and each string is parsed sequentially to determine the specific program action in each case. The boundary data file utilizes a keyword input format, such that any line which does not contain a recognizable keyword is treated as a comment

59

2–D Mesh Block Phantom Cell Representation

●   *Grid Point*

────────   *Grid Line*

━━━━━━━   *Mesh Block Boundary*

− − − −   *Phantom Cell Representation*

Boundary condition specifications control the flow variables for the phantom cells adjacent to the mesh block boundary

"Corner" phantom cells cannot be controlled through boundary conditions, but must be updated to accurately compute grid point averaged values
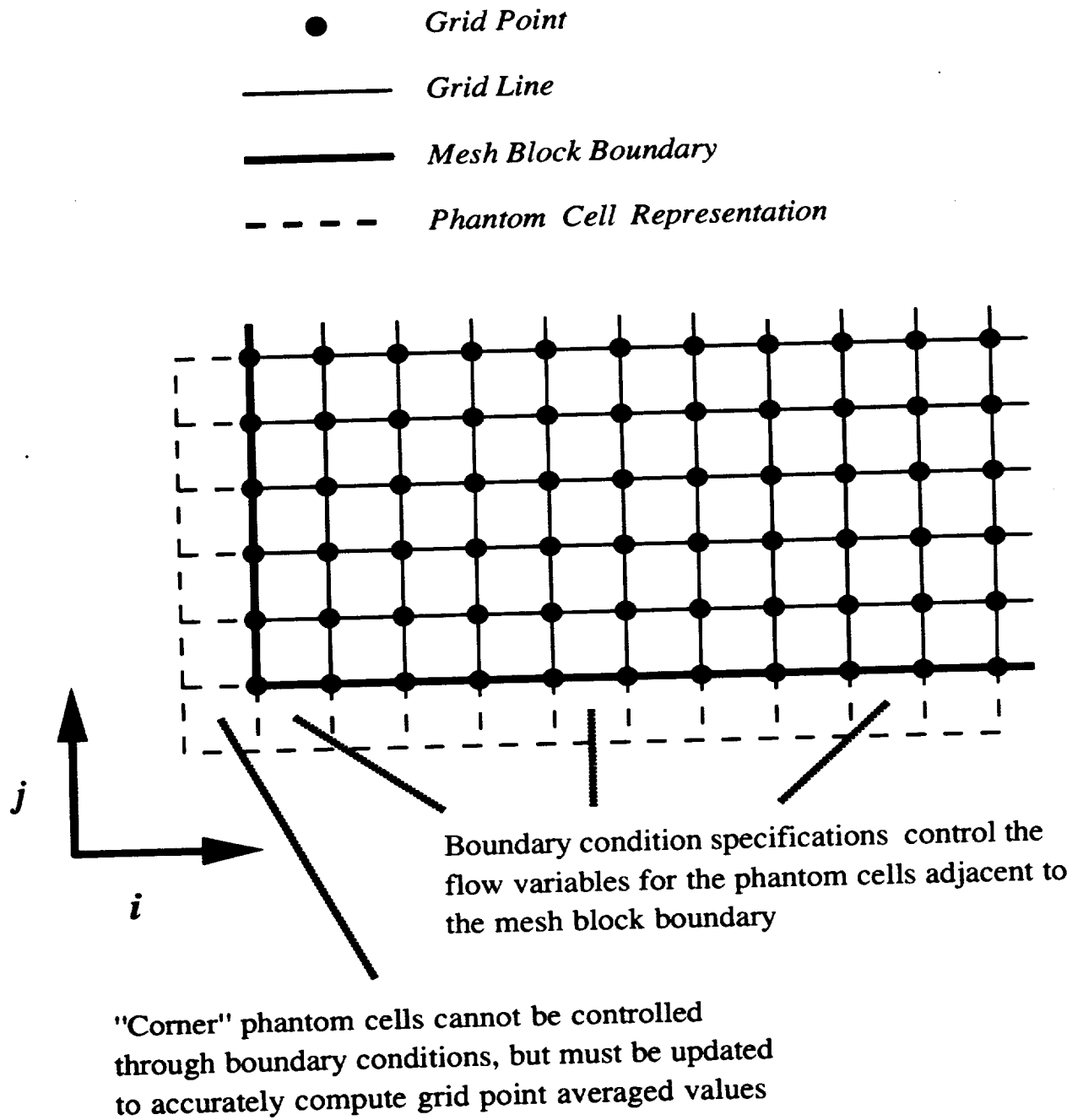
Figure 3.3:   2-D Mesh Block Phantom Cell Representation

60

ADPAC-AOACR 3-D Boundary Condition Specification

All block boundary conditions are specified as a grid–defined "patch" on an i=constant, j=constant, or k=constant mesh face
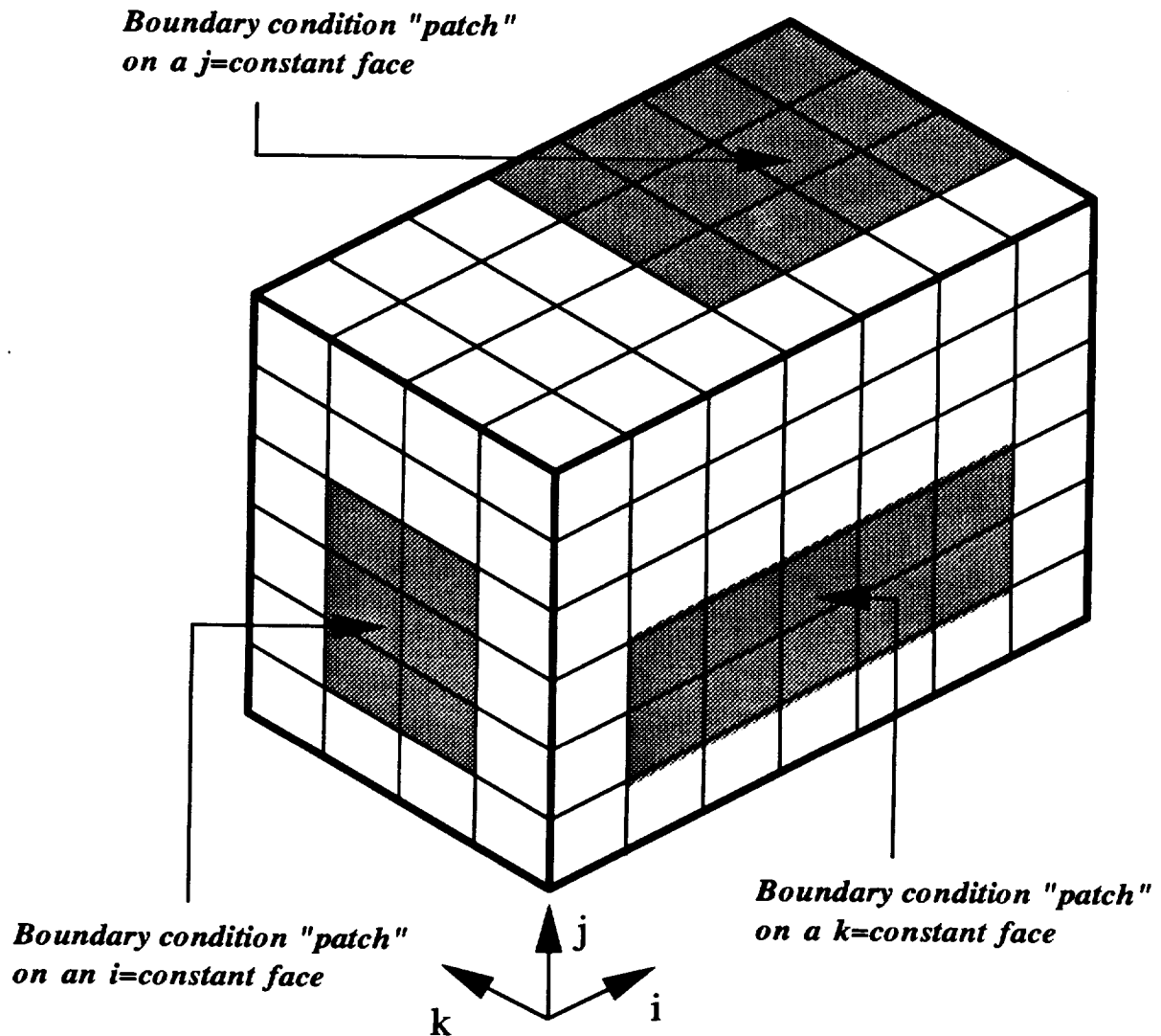
Patches may be internal to the mesh as well

Figure 3.4: *ADPAC-AOACR* 3-D Boundary Condition Specification

Effect of Ordering in Application of Boundary Conditions for *ADPAC-AOACR* Code

Computational Domain

Boundary condition "A" applied first

Boundary condition "B" applied after boundary condition "A" - part of boundary condition "A" is overwritten

Boundary condition "C" applied after boundary condition "B" and "A" - part of boundary conditions "B" and "A" are overwritten
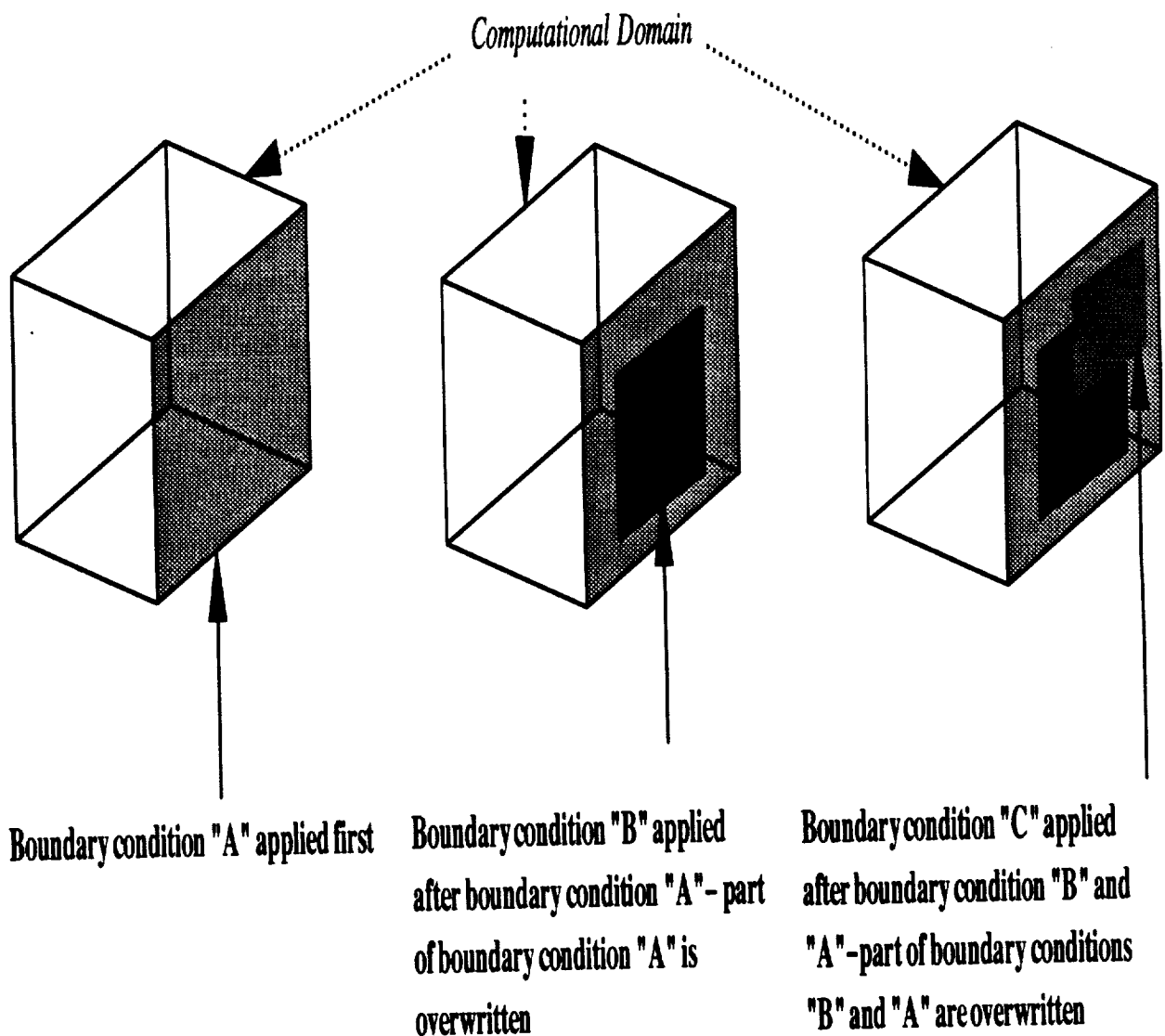
Figure 3.5: Effect of Ordering in Application of Boundary Conditions for the *ADPAC-AOACR* Code

line. Therefore, the user may place any number of comments in the file (so long as the line does not contain a keyword input string in the form described below), and code execution is unaltered. All boundary data file lines are echoed to the standard output, and the program response to each line is listed when a specific action is taken. A line in the boundary data file can also be effectively commented by inserting a # character in the first column. Therefore the lines

```
PATCH   1  1  K  K  P  M  I  J   1  17   1 129   1  17   1 129   1  17
PATCH   2  2  K  K  P  M  I  J   1  17   1 129   1  17   1 129   1  17
```

are acceptable boundary specifications; however, the lines

```
#PATCH   1  1  K  K  P  M  I  J   1  17   1 129   1  17   1 129   1  17
#PATCH   2  2  K  K  P  M  I  J   1  17   1 129   1  17   1 129   1  17
```

would be neglected.

All keyword input lines are given in the format listed in Figure 3.6. The actual specification in the boundary data file may be free format, as long as the individual parameter specifications are given in the correct order and are separated by one or more blank spaces.

All boundary specifications begin with a line containing the variables:

```
BCTYPE
LBLOCK1
LBLOCK2
LFACE1
LFACE2
LDIR1
LDIR2
LSPEC1
LSPEC2
L1LIM
L2LIM
M1LIM1
M1LIM2
```

CASE.BOUNDATA FILE

This file contains case-specific boundary data information. All boundary condition applications are represented by 1 or more lines in this file.

A sample line is given in the highlighted region below:

SSIN 1 1 J J P P L L L L 171 171 171 171 171

This is the keyword describing the type of boundary condition to be applied

This is the mesh block number to which the boundary data is applied

This is the mesh block number from which the boundary data is derived

This indicates the grid surface (i,j,k) to which the boundary data is applied in block LBLOCK1

This indicates the grid surface (i,j,k) from which the boundary data is derived in block LBLOCK2

This indicates the direction (+=P, -=M) along the LFACE1 coordinate which travels towards the interior of the flow (away from the bounding surface) in mesh block LBLOCK1

This indicates the direction (+=P, -=M) along the LFACE2 coordinate which travels towards the interior of the flow (away from the bounding surface) in mesh block LBLOCK2

These are triggers reserved for special use in some boundary conditions (usually the value LI, or K, which indicates the correspondence of the coordinates in mesh block LBLOCK2 with the remaining (non LFACE1) coordinates in mesh block LBLOCK1

These are the beginning and ending indices for the corresponding coordinate directions in block LBLOCK1 from which boundary data is applied

These are the begining and ending indices for the remaining coordinate directions in block LBLOCK1 to which boundary data is applied

This is the index in the LDIR2 direction for boundary data in block LBLOCK2

This is the index in the LDIR1 direction for boundary data in block LBLOCK1

Figure 3.6: *ADPAC-AOACR* Boundary Data File Specification Format

```
N1LIM1
N1LIM2
M2LIM1
M2LIM2
N2LIM1
N2LIM2
```

These variables are outlined in Figure 3.6. A description of the function of each of these variables in the boundary specification line is given below:

Description of Boundary Specification Line Variables

BCTYPE A 6 character string defining the type of boundary condition type to be applied. **BCTYPE** must correspond to one of the reserved keywords defined below to be a proper boundary specification. If **BCTYPE** is not one of the reserved names, then the boundary specification line is ignored.

LBLOCK1 An integer defining the grid block number to which the boundary condition implied by **BCTYPE** is applied. Naturally, this implies **LBLOCK1** ≥ 1, and **LBLOCK1** $\leq$ **NBLKS**, where **NBLKS** represents the last mesh block.

LBLOCK2 An integer defining the grid block number from which the boundary condition data is obtained. In some cases, a boundary specification may involve more than one block (patching two blocks together is an example), and the **LBLOCK2** variable is provided for this purpose. This value is only used in certain routines. If the boundary specification only involves a single block, then set **LBLOCK2** = **LBLOCK1**.

LFACE1 A single character (one of the letters I, J, or K) specifying the grid plane (i=constant, j=constant, or k=constant) to which the boundary condition is applied in block **LBLOCK1**. This specification determines the grid face to which the boundary specification is applied, based on the method by which boundary conditions are

implemented in the finite-volume solution scheme (see the discussion and figures above).

LFACE2 A single character (one of the letters I, J, or K) specifying the grid plane (i=constant, j=constant, or k=constant) from which the boundary condition data is derived in block **LBLOCK2**. This specification determines the grid face from which the neighboring block boundary data is derived, based on the method by which boundary conditions are implemented in the finite-volume solution scheme (see the discussion and figures above). Naturally, this variable is only useful for boundary specifications involving more than one block. If only one block is involved, simply set **LFACE2 = LFACE1**.

LDIR1 A single character (one of the letters P or M) specifying the direction (P=plus, M=minus) along the **LDIR1** coordinate in **LBLOCK1** which is away (towards the interior flow region) from the boundary surface patch. The specification of this variable is normally automatic when the boundary specification is applied to the external surface of a grid block - (**LDIR1** = P when **L1LIM** = 1, and **LDIR1** = M when **L1LIM** = IMX, JMX, or KMX. (IMX, JMX, KMX indicate the maximum indices of the **LBLOCK1** mesh block in the i, j, and k directions, respectively). The intent here is to provide a means of specifying which side of the boundary surface plane the interior computational cells (non-phantom cells) lie on. This specification is made by providing the coordinate direction of the interior computational cells - the phantom cells are then assumed to lie in the opposite direction.

LDIR2 A single character (one of the letters P or M) specifying the direction (P=plus, M=minus) along the **LDIR2** coordinate in **LBLOCK2** which is away (towards the interior flow region) from the boundary surface patch. This variable is only used in boundary specifications cases involving more than one mesh block. The specification of this variable is normally automatic when the boundary specifi-

cation data is obtained from the external surface of a neighboring grid block - (**LDIR2** = P when **L2LIM** = 1, and **LDIR2** = M when **L2LIM** = *IMX,JMX,* or *KMX*. (*IMX,JMX,KMX* indicate the maximum indices of the **LBLOCK2** mesh block in the $i, j,$ and k directions, respectively). The intent here is to provide a means of specifying which side of the boundary surface plane the interior computational cells (non-phantom cells) lie on. This specification is made by providing the coordinate direction of the interior computational cells - the phantom cells are then assumed to lie in the opposite direction. If the boundary specification involves only a single mesh block, then simply set **LDIR2** = **LDIR1**.

LSPEC1   A single character (usually I, J, K, L, or H) which implies some special information about the boundary condition specification. This parameter is usually boundary condition dependent. The most common application of this variable is in the boundary data file keyword **PATCH**, which provides the cell to cell connection for two grid blocks with a mating contiguous surface. For boundary conditions involving more than one mesh block (such as **PATCH**), it is possible that the connection between blocks may involve connections between different grid surfaces, and that the indices in block **LBLOCK2** correspond to a different coordinate in block **LBLOCK1**. The variables **M2LIM1, M2LIM2** control the indices in the **LSPEC1** direction in block **LBLOCK2** which correspond to the indices determined by **M1LIM1, M1LIM2** in block **LBLOCK1**. For cases when **LSPEC1** is not used, this variable can be any single character.

LSPEC2   A single character (usually I, J, K, L, or H) which implies some special information about the boundary condition specification. This parameter is usually boundary condition dependent. The most common application of this variable is in the boundary data file keyword **PATCH**, which provides the cell to cell connection for two grid blocks with a mating contiguous surface. For boundary

conditions involving more than one mesh block (such as **PATCH**), it is possible that the connection between blocks may involve connections between different grid surfaces, and that the indices in block **LBLOCK2** correspond to a different coordinate in block **LBLOCK1**. The variables **N2LIM1**, **N2LIM2** control the indices in the **LSPEC2** direction in block **LBLOCK2** which correspond to the indices determined by **N1LIM1**, **N1LIM2** in block **LBLOCK1**. For cases when **LSPEC1** is not used, this variable can be any single character.

L1LIM An integer specifying the index of the grid in the **LFACE1** direction to which the boundary condition should be applied in block **LBLOCK1**. This value determines the actual mesh index of the i=constant, j=constant, or k=constant mesh face (determined by **LFACE1**) to which the boundary condition is applied in mesh block **LBLOCK1**.

L2LIM An integer specifying the index of the grid in the **LFACE2** direction from which the boundary condition data is derived in block **LBLOCK2**. This value determines the actual mesh index of the i=constant, j=constant, or k=constant mesh face (determined by **LFACE2**) from which the boundary condition data is derived in mesh block **LBLOCK2**.

M1LIM1 An integer representing the initial index of the first remaining grid coordinate direction to which the boundary condition is applied in block **LBLOCK1**. Since the boundary specification applies to either an i=constant, j=constant, or k=constant surface, the variables **M1LIM1**, **M1LIM2**, **N1LIM1** and **N1LIM2** determine the extent of the patch in the remaining coordinate directions. The remaining coordinate directions for block **LBLOCK1** are specified in the natural order. (For example, if **LFACE1**=I, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the j direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=J, then the variables **M1LIM1**,

68

M1LIM2 refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=K, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the j direction.) The indices specified in **M1LIM1** and **M1LIM2** must be given in increasing order. The indices specified in **N1LIM1** and **N1LIM2** must also be given in increasing order.

M1LIM2   An integer representing the final index of the first remaining grid coordinate direction to which the boundary condition is applied in block **LBLOCK1**. Since the boundary specification applies to either an i=constant, j=constant, or k=constant surface, the variables **M1LIM1**, **M1LIM2**, **N1LIM1** and **N1LIM2** determine the extent of the patch in the remaining coordinate directions. The remaining coordinate directions for block **LBLOCK1** are specified in the natural order. (For example, if **LFACE1**=I, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the j direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=J, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=K, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the j direction.) The indices specified in **M1LIM1** and **M1LIM2** must be given in increasing order. The indices specified in **N1LIM1** and **N1LIM2** must also be given in increasing order.

N1LIM1   An integer representing the initial index of the second remaining grid coordinate direction to which the boundary condition is applied in block **LBLOCK1**. Since the boundary specification applies to either an i=constant, j=constant, or k=constant surface, the variables **M1LIM1**, **M1LIM2**, **N1LIM1** and **N1LIM2** de-

termine the extent of the patch in the remaining coordinate directions. The remaining coordinate directions for block **LBLOCK1** are specified in the natural order. (For example, if **LFACE1**=I, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the j direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=J, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=K, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the j direction.) The indices specified in **M1LIM1** and **M1LIM2** must be given in increasing order. The indices specified in **N1LIM1** and **N1LIM2** must also be given in increasing order. For boundaries on 2-D mesh blocks, this must always be 1.

N1LIM2   An integer representing the final index of the second remaining grid coordinate direction to which the boundary condition is applied in block **LBLOCK1**. Since the boundary specification applies to either an i=constant, j=constant, or k=constant surface, the variables **M1LIM1**, **M1LIM2**, **N1LIM1** and **N1LIM2** determine the extent of the patch in the remaining coordinate directions. The remaining coordinate directions for block **LBLOCK1** are specified in the natural order. (For example, if **LFACE1**=I, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the j direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=J, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the k direction. If **LFACE1**=K, then the variables **M1LIM1**, **M1LIM2** refer to the extent in the i direction and the variables **N1LIM1**, **N1LIM2** refer to the extent in the j direction.) The indices specified in **M1LIM1** and **M1LIM2** must be given in increasing order. The

indices specified in **N1LIM1** and **N1LIM2** must also be given in increasing order. For boundaries on 2-D mesh blocks, this must always be 2.

M2LIM1 An integer representing the initial index of the grid coordinate direction in block **LBLOCK2** corresponding to the first remaining coordinate in block **LBLOCK1**. For boundary conditions involving more than one mesh block, it is possible that the connection between blocks may involve connections between different grid surfaces, and that the indices in block **LBLOCK2** correspond to a different coordinate in block **LBLOCK1**. The variables **M2LIM1**, **M2LIM2** control the indices in the **LSPEC1** direction in block **LBLOCK2** which correspond to the indices determined by **M1LIM1**, **M1LIM2** in block **LBLOCK1**. The user should note that it is possible for **M2LIM1** > **M2LIM2** and **N2LIM1** > **N2LIM2** but it is not possible for **M1LIM1** > **M1LIM2** and **N1LIM1** > **N1LIM2**. If only a single mesh block is involved in the boundary specification, set **M2LIM1** = **M1LIM1**.

M2LIM2 An integer representing the final index of the grid coordinate direction in block **LBLOCK2** corresponding to the first remaining coordinate in block **LBLOCK1**. For boundary conditions involving more than one mesh block, it is possible that the connection between blocks may involve connections between different grid surfaces, and that the indices in block **LBLOCK2** correspond to a different coordinate in block **LBLOCK1**. The variables **M2LIM1**, **M2LIM2** control the indices in the **LSPEC1** direction in block **LBLOCK2** which correspond to the indices determined by **M1LIM1**, **M1LIM2** in block **LBLOCK1**. The user should note that it is possible for **M2LIM1** > **M2LIM2** and **N2LIM1** > **N2LIM2** but it is not possible for **M1LIM1** > **M1LIM2** and **N1LIM1** > **N1LIM2**. If only a single mesh block is involved in the boundary specification, set **M2LIM2** = **M1LIM2**.

N2LIM1 An integer representing the initial index of the grid coordinate

direction in block **LBLOCK2** corresponding to the second remaining coordinate in block **LBLOCK1**. For boundary conditions involving more than one mesh block, it is possible that the connection between blocks may involve connections between different grid surfaces, and that the indices in block **LBLOCK2** correspond to a different coordinate in block **LBLOCK1**. The variables **N2LIM1, N2LIM2** control the indices in the **LSPEC2** direction in block **LBLOCK2** which correspond to the indices determined by **N1LIM1, N1LIM2** in block **LBLOCK1**. The user should note that it is possible for **M2LIM1 > M2LIM2** and **N2LIM1 > N2LIM2** but it is not possible for **M1LIM1 > M1LIM2** and **N1LIM1 > N1LIM2**. If only a single mesh block is involved in the boundary specification, set **N2LIM1 = N1LIM1**. For boundary data on 2-D mesh blocks, this must always be 1.

N2LIM2   An integer representing the final index of the grid coordinate direction in block **LBLOCK2** corresponding to the second remaining coordinate in block **LBLOCK1**. For boundary conditions involving more than one mesh block, it is possible that the connection between blocks may involve connections between different grid surfaces, and that the indices in block **LBLOCK2** correspond to a different coordinate in block **LBLOCK1**. The variables **N2LIM1, N2LIM2** control the indices in the **LSPEC2** direction in block **LBLOCK2** which correspond to the indices determined by **N1LIM1, N1LIM2** in block **LBLOCK1**. The user should note that it is possible for **M2LIM1 > M2LIM2** and **N2LIM1 > N2LIM2** but it is not possible for **M1LIM1 > M1LIM2** and **N1LIM1 > N1LIM2**. If only a single mesh block is involved in the boundary specification, set **N2LIM2 = N1LIM2**. For boundary data on 2-D mesh blocks, this must always be 2.

Some boundary condition specifications require additional data beyond that incorporated in the boundary specification line. In these cases, described in detail for the specific boundary types later in this Section, the additional data is included

72

immediately after the boundary specification line.

A sample *ADPAC-AOACR* boundary data file containing several keywords is listed below.

Sample *ADPAC-AOACR* Boundary Data File

```
------------------------------------------------------
ADPAC Boundata File Generated by SETUP-Version 1.0
Apr 15 1992, at 17:36:38
------------------------------------------------------
  This file contains block boundary condition data information for the
  ADPAC-AOACR multiple grid block Euler-Navier-Stokes code. All boundary
  specifications begin with a line containing the variables:

The routine selected is for a two-block 3-D H-grid
about a ducted propfan as shown below:


                      I-J Plane


J=JL  -------------------------------------------------
      |Block #2              COWL|<--MIXING PLANE      |
      |          I=IGCLE    | |     I=IGCTE            |
      |     (COWL L.E.)_____V_|_____ (COWL T.E.)     |
      |------------<_____>--------------|
      |               ____  |  ____                    |
      |          J=JTIP/   |  | /  |<--- BLADE         |
      |               /   /  | /   /                   |
      |              |   /   | |  /                    |
      |Block #1      --------------------             |
      |          /  I=ILE  I=ITE         |            |
J=1   -----------/  LEADING TRAILING      ----------
      I=1   I=ISLE   EDGE    EDGE     I=ISTE     I=IL
            SPINNER                   SPINNER
         LEADING EDGE             TRAILING EDGE


                     I-K Plane
```

73

```
K=KL ------------- -----   -----------
    |            |_____|  | |_____|          |
    |              A      |    A             |
    |              |      |    |             |
    |   BLADE SURFACE     |  BLADE SURFACE   |
    |              |      |    |             |
    |  (Odd # blocks      |    |             |
    |    only)     |      |    |             |
    |              |      |<-------- MIXING  |
    |              |      |    |       PLANE |
    |              |      |    |             |
    |              V      |    V             |
K=1  -------------|   ---V--  |----|  |-----------|
    I=1            I=ILE   I=ITE           I=IL
```
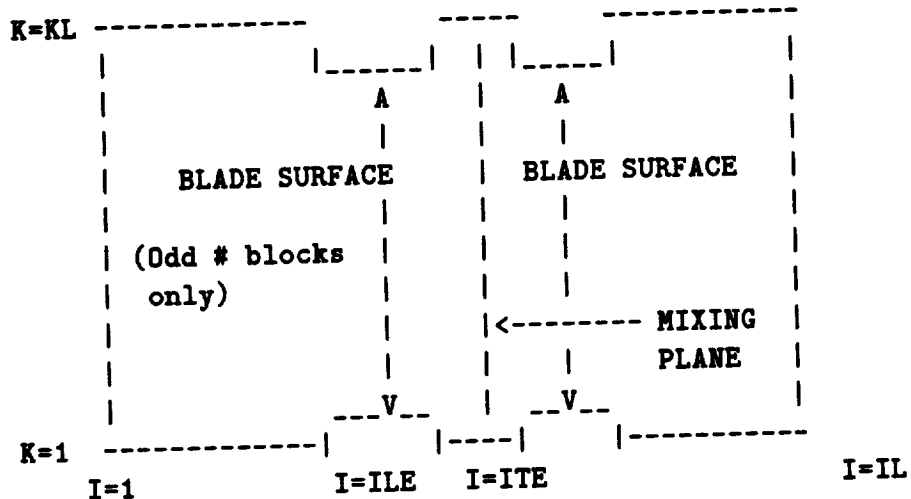
BLOCKDATA FOLLOWS: LABELS

BCTYPE	LBLOCK1	LBLOCK2	LFACE1	LFACE2	LDIR1	LDIR2	LSPEC1	LSPEC2	L1LIM	L2LIM	M1LIM1	M1LIM2	N1LIM1	N1LIM2	M2LIM1	M2LIM2	N2LIM1	N2LIM2	COMMENT
PATCH	1	1	K	K	P	M	I	J	1	17	1	129	1	17	1	129	1	17	%K=1 Per
PATCH	2	2	K	K	P	M	I	J	1	17	1	129	1	17	1	129	1	17	%K=1 Per
PATCH	3	3	K	K	P	M	I	J	1	17	1	97	1	17	1	97	1	17	%K=1 Per
PATCH	4	4	K	K	P	M	I	J	1	17	1	97	1	17	1	97	1	17	%K=KL Per
PATCH	1	1	K	K	M	P	I	J	17	1	1	129	1	17	1	129	1	17	%K=KL Per
PATCH	2	2	K	K	M	P	I	J	17	1	1	129	1	17	1	97	1	17	%K=KL Per
PATCH	3	3	K	K	M	P	I	J	17	1	1	97	1	17	1	97	1	17	%K=KL Per
PATCH	4	4	K	K	M	P	I	J	17	1	1	97	1	17	1	129	1	17	%Hub Inv
SSIN	1	1	J	J	P	P	S	S	1	1	1	129	1	17	1	97	1	17	%Hub Inv
SSIN	3	3	J	J	P	P	S	S	1	1	1	97	1	17	81	113	1	17	%K=1 Bld
SSIN	1	1	K	K	P	P	S	S	1	1	81	113	1	17	81	113	1	17	%K=KL Bld
SSIN	1	1	K	K	M	M	S	S	17	17	81	113	1	17	17	49	1	17	%K=1 Bld
SSIN	3	3	K	K	P	P	S	S	1	1	17	49	1	17	17	49	1	17	%K=KL Bld
SSIN	3	3	K	K	M	M	S	S	17	17	17	49	1	17	1	17	1	17	%Inlet
INLETA	1	1	I	I	P	P	S	S	1	1	1	17	1	17	1	17	1	17	%Inlet

74

```
PT          TT          ALPHA
1.000000   1.000000   0.000000
INLETA 2  2  I  I  P  P  S  S   1   1    1  17    1  17    1  17    1  17 %Inlet
PT          TT          ALPHA
1.000000   1.000000   0.000000
EXITP  3  3  I  I  M  M  H  H  97  97    1  17    1  17    1  17    1  17 %Inlet
NBOUN      IBOUN       JBOUN
    000004      000097      000001
EXITT  4  4  I  I  M  M  H  H  97  97    1  17    1  17    1  17    1  17 %Inlet
  PEXIT
  0.3609139
FREE   2  2  J  J  M  M  S  S  17  17    1 129    1  17    1 129    1  17 %Free flo
PT          TT          MACH(INF)  ALPHA
1.000000   1.000000   1.300000   0.000000
FREE   4  4  J  J  M  M  S  S  17  17    1  97    1  17    1  97    1  17 %Free flo
PT          TT          MACH(INF)  ALPHA
1.000000   1.000000   1.300000   0.000000
PATCH  1  2  J  J  M  P  I  J  17   1    1 129    1  17    1 129    1  17 %Ptch 1-2
PATCH  2  1  J  J  P  M  I  J   1  17    1 129    1  17    1 129    1  17 %Ptch 1-2
PATCH  3  4  J  J  M  P  I  J  17   1    1  97    1  17    1  97    1  17 %Ptch 1-2
PATCH  4  3  J  J  P  M  I  J   1  17    1  97    1  17    1  97    1  17 %Ptch 1-2
SSIN   1  1  J  J  M  M  S  S  17  17   33 129    1  17   33 129    1  17 %COWL 1
SSIN   2  2  J  J  P  P  S  S   1   1   33 129    1  17   33 129    1  17 %COWL 2
SSIN   3  3  J  J  M  M  S  S  17  17    1  81    1  17    1  81    1  17 %COWL 1
SSIN   4  4  J  J  P  P  S  S   1   1    1  81    1  17    1  81    1  17 %COWL 2
CIRAV  1  3  I  I  M  P  J  K 129   1    1  17    1  17    1  17    1  17 %Up Cirav
CIRAV  2  4  I  I  M  P  J  K 129   1    1  17    1  17    1  17    1  17 %Up Cirav
CIRAV  3  1  I  I  P  M  J  K   1 129    1  17    1  17    1  17    1  17 %Dn Cirav
CIRAV  4  2  I  I  P  M  J  K   1 129    1  17    1  17    1  17    1  17 %Dn Cirav
ENDDATA
```

A list and description of all valid boundary data keywords and any additional data required for the given boundary condition is now presented below. A quick reference to the boundary data file keywords is provided in Appendix C.

ADPAC-AOACR Boundary Data File Keyword Description

Keyword: **ENDDATA**
Format:

ENDDATA

Description: The **ENDDATA** statement indicates that all relevant boundary data statements have been specified, and therefore causes the termination of the reading of additional statements, whether they are valid boundary condition specifications or not. This is not a required statement since the boundary data input routine will essentially terminate when the end of the *case.boundata* file is reached. The **ENDDATA** statement is most useful as a visual identifier of the termination of the valid boundary data statements, and as a means of terminating the boundary data read routine before other, non-applicable boundary data statments are reached. For example, if there are boundary specifications in a file which are not desired, these statements could be effectively negated by either incorporating the comment symbol (#), or by placing them after an **ENDDATA** statement.

Keyword: **SSVI**
Format:

```
SSVI    1  1  J  J  P  P  I  K  1  1  1  21  1  11  1  21  1  11
RPM        TWALL
0.0        0.0
```

Description: The **SSVI** statement specifies that a solid surface viscous (no-slip) boundary condition is to be applied to the specified mesh surface. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variable following **RPM** is the desired solid wall dimensional rotational speed in revolutions per minute. In many cases, the rotational speed of the boundary differs from the rotational speed of the mesh block itself. A common example of this is flow through a fan rotor with tip clearance using a single blade-passage H-grid. The blade passage mesh block rotates with the blade, but the endwall

76

is non-rotating, thus requiring a specification of a zero rotational speed for the endwall boundary. The variable **TWALL** determines which type of temperature condition is applied to the surface. If **TWALL**=0.0, an adiabatic wall is assumed. For **TWALL**>0.0, a constant temperature surface with a nondimensional wall temperature of **TWALL** defined as:

$$(T_{wall})_{non-dimensional} = T_{wall}/T_{ref}$$

is imposed. A value of **TWALL**<0.0 is not permitted.

Keyword: **SS2DVI**
Format:

```
SS2DVI 1  1  J  J  P  P  I  K  1  1  1  21  1  2  1  21  1  2
·RPM       TWALL
 0.0        0.0
```

Description: The **SS2DVI** statement specifies that a solid surface viscous (no-slip) boundary condition is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **SSVI** described above. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variables **RPM** and **TWALL** are defined in the definition of **SSVI**, above.

Keyword: **SSIN**
Format:

```
SSIN    1  1  J  J  P  P  I  K  1  1  1  21  1  11  1  21  1  11
```

Description: The **SSIN** statement specifies that a solid surface inviscid (no through-flow) boundary condition is to be applied to the specified mesh surface. This boundary condition requires no additional data. The boundary rotational speed is assumed to be equal to the mesh block rotational speed.

Keyword: SS2DIN

Format:

```
SS2DIN  1  1  J  J  P  P  I  K  1  1  1  21  1  2  1  21  1  2
```

Description: The **SS2DIN** statement specifies that a solid surface inviscid (no through-flow) boundary condition is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **SSIN** described above. This boundary condition requires no additional data. The boundary rotational speed is assumed to be equal to the mesh block rotational speed.

Keyword: INLETG

Format:

```
INLETG  1  1  I  I  P  P  J  K  1  1  1  21  1  11  1  21  1  11
    PTOT       TTOT
   1.0253     1.0568
```

Description: The **INLETG** statement specifies that a subsonic generic inflow boundary condition is to be applied to the specified mesh surface. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variables following PTOT, TTOT are the desired nondimensional inlet upstream total pressure and temperature defined as:

$$(P_{total,upstream})_{non-dimensional} = \frac{P_{total,upstream}}{P_{ref}}$$

$$(T_{total,upstream})_{non-dimensional} = \frac{T_{total,upstream}}{T_{ref}}$$

The generic inflow boundary sets the inlet velocity such that the flow is locally normal to the mesh surface.

Keyword: INL2DG

Format:

```
INL2DG  1  1  I  I  P  P  J  K  1  1  1  21  1  2  1  21  1  2
      PTOT       TTOT
      1.0253     1.0568
```

Description: The **INL2DG** statement specifies that a subsonic generic inflow
boundary condition is to be applied to the specified two-dimensional
mesh surface. This is the 2-D equivalent of the 3-D boundary
specification **INLETG** described above. This boundary condition
requires the specification of additional data, as shown in the For-
mat descriptor above. The value of the variables following PTOT,
TTOT are described in the definition of **INLETG**, above.

Keyword: **INLETA**

Format:

```
INLETA  1  1  I  I  P  P  J  K  1  1  1  21  1  11  1  21  1  11
      PTOT       TTOT      ALPHA
      1.0253     1.0568     10.0
```

Description: The **INLETA** statement specifies that an angle of attack inflow
boundary condition is to be applied to the specified mesh surface.
This boundary condition requires the specification of additional
data, as shown in the Format descriptor above. The value of the
variables following **PTOT, TTOT** are the desired nondimensional
inlet upstream total pressure and temperature defined as:

$$(P_{total,upstream})_{non-dimensional} = \frac{P_{total,upstream}}{P_{ref}}$$

$$(T_{total,upstream})_{non-dimensional} = \frac{T_{total,upstream}}{T_{ref}}$$

The value of the variable **ALPHA** is the freestream angle of at-
tack, measured in degrees from the axis of rotation of the fan (see
Figure 3.7). This boundary condition is utilized for ducted and
unducted profans at angle of attack, and the flow angle is applied

ADPAC-AOACR Angle of Attack Definition
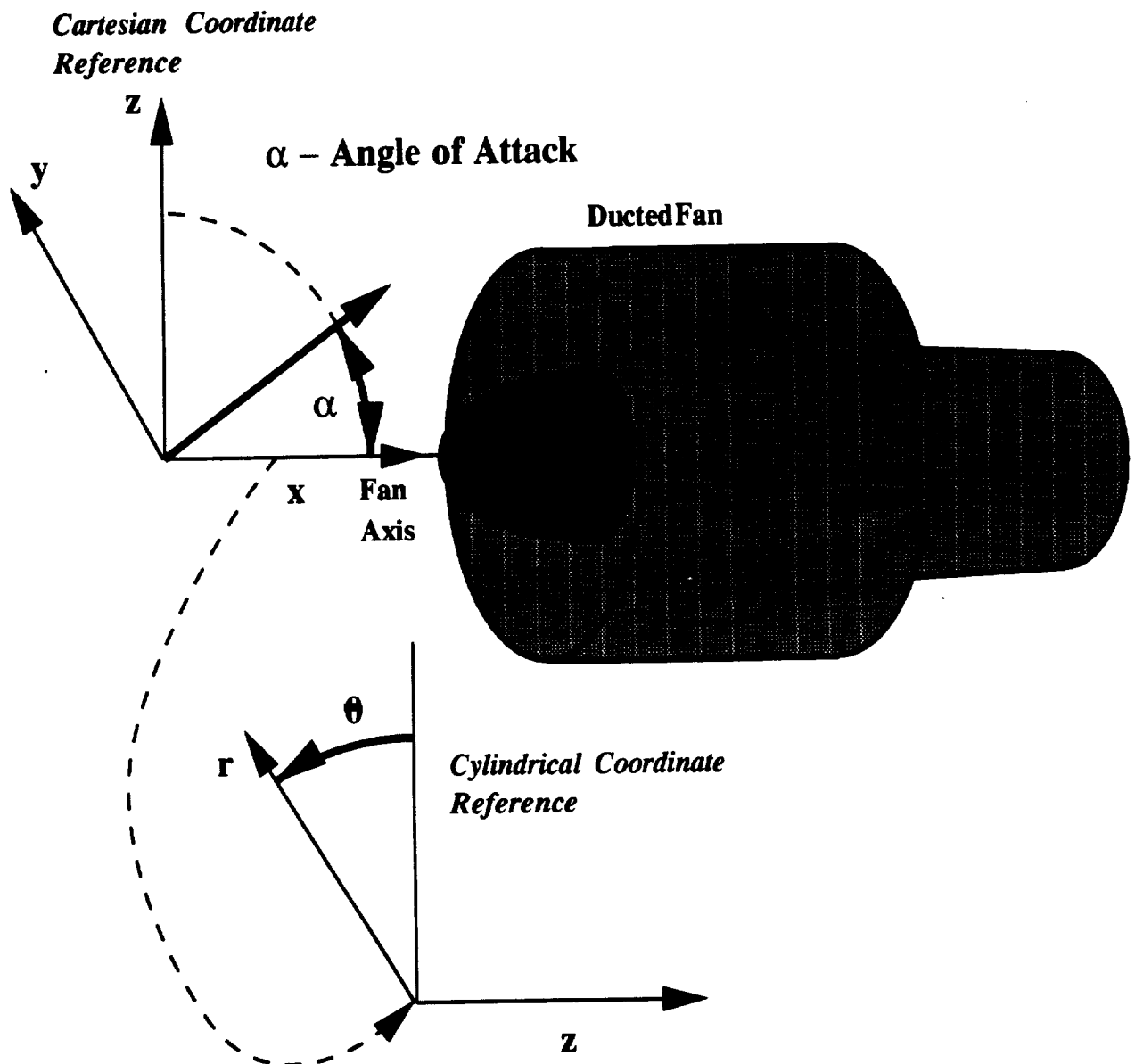
Figure 3.7: ADPAC-AOACR Angle of Attack Reference Description

in a Cartesian sense (positive angle of attack is flow pointing in the positive z direction, see Figure 3.7.

Keyword: **INL2DA**
Format:

```
INL2DA  1  1  I  I  P  P  J  K  1  1  1  21  1  2  1  21  1  2
     PTOT        TTOT     ALPHA
     1.0253      1.0568     10.0
```

Description: The **INL2DA** statement specifies that an angle of attack inflow boundary condition is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **INLETA** described above. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variables following **PTOT**, **TTOT** and **ALPHA** are defined in the description of INLETA, above.

Keyword: **INLETT**
Format:

```
INLETT  1  1  I  I  P  P  J  K  1  1  1  21  1  11  1  21  1  11
NDATA
5
    RAD  PTOT   TTOT   BETAR   BETAT
    0.05  0.91  1.00    0.0    45.0
    0.15  0.98  1.00    2.0    48.0
    0.50  0.99  1.00    3.0    49.0
    0.75  0.99  1.00   -2.0    47.0
    0.95  0.92  1.00    0.0    35.0
```

Description: The **INLETT** statement specifies that a subsonic turbomachinery-type inflow boundary condition is to be applied to the specified mesh surface. This boundary condition permits a radial specification of upstream flow variables (total pressure, total temperature,

81

radial flow angle, and circumferential flow angle). This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variable labeled **NDATA** indicates the number of points of radial flow data to be used in the specification of the upstream flow variables. The lines following this are the **NDATA** points of radial data specified as **RAD** (radial position), **PTOT** (upstream total pressure), **TTOT** (upstream total temperature) BETAR (radial flow angle, increasing radial direction being positive - see Figure 3.8), and BE-TAT (circumferential flow angle, increasing θ being positive - see Figure 3.8). The **NDATA** radial data points will be used to interpolate the data to the actual inlet plane grid coordinates. The value of the variables following **RAD** is the nondimensional radius defined as:

$$r_{non-dimensional} = \frac{r}{d_{ref}}$$

the values of the variables following PTOT, TTOT are the desired nondimensional inlet upstream total pressure and temperature defined as:

$$\left(P_{total,upstream}\right)_{non-dimensional} = \frac{P_{total,upstream}}{P_{ref}}$$

$$\left(T_{total,upstream}\right)_{non-dimensional} = \frac{T_{total,upstream}}{T_{ref}}$$

Note that the radial data point RAD has been nondimensionalized by the same diameter as the grid (see Section 3.8 for a description of the mesh nondimensionalization). The total pressure and total temperature radial data **PTOT,TTOT** have been nondimensionalized by the input file values **PREF,TREF**. Normally, these values are chosen as an average overall inlet value and the radial distribution is nondimensionalized accordingly. This boundary condition is utilized for many turbomachinery flow calculations

where the effects of upstream blade rows are modeled by this radial variation in inlet properties. It should be mentioned that the data specified by the **NDATA** points are normally expected to encompass the entire radial span of the mesh section to which the boundary condition is applied. If this condition is not satisfied, the *ADPAC-AOACR* program will extrapolate the data beyond the inner and outer radial limits of the imposed data, which could cause in erroneous results.

Keyword: **INL2DT**
Format:

```
INL2DT 1  1  I  I  P  P  J  K  1  1  1 21  1  2  1 21  1  2
NDATA
5
     RAD  PTOT  TTOT  BETAR   BETAT
    0.05  0.91  1.00   0.0    45.0
    0.15  0.98  1.00   2.0    48.0
    0.50  0.99  1.00   3.0    49.0
    0.75  0.99  1.00  -2.0    47.0
    0.95  0.92  1.00   0.0    35.0
```

Description: The **INL2DT** statement specifies that a subsonic turbomachinery-type inflow boundary condition is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **INLETT** described above. This boundary condition permits a radial specification of upstream flow variables (total pressure, total temperature, radial flow angle, and circumferential flow angle). This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variables labeled **NDATA, RAD, PTOT, TTOT, BETAR**, and **BETAT** are described in the definition of **INLETT**, above.

Keyword: **EXITG**
Format:

ADPAC–AOACR Turbomachinery Flow Angle Description

Radial Flow Angle

Circumferential Flow Angle
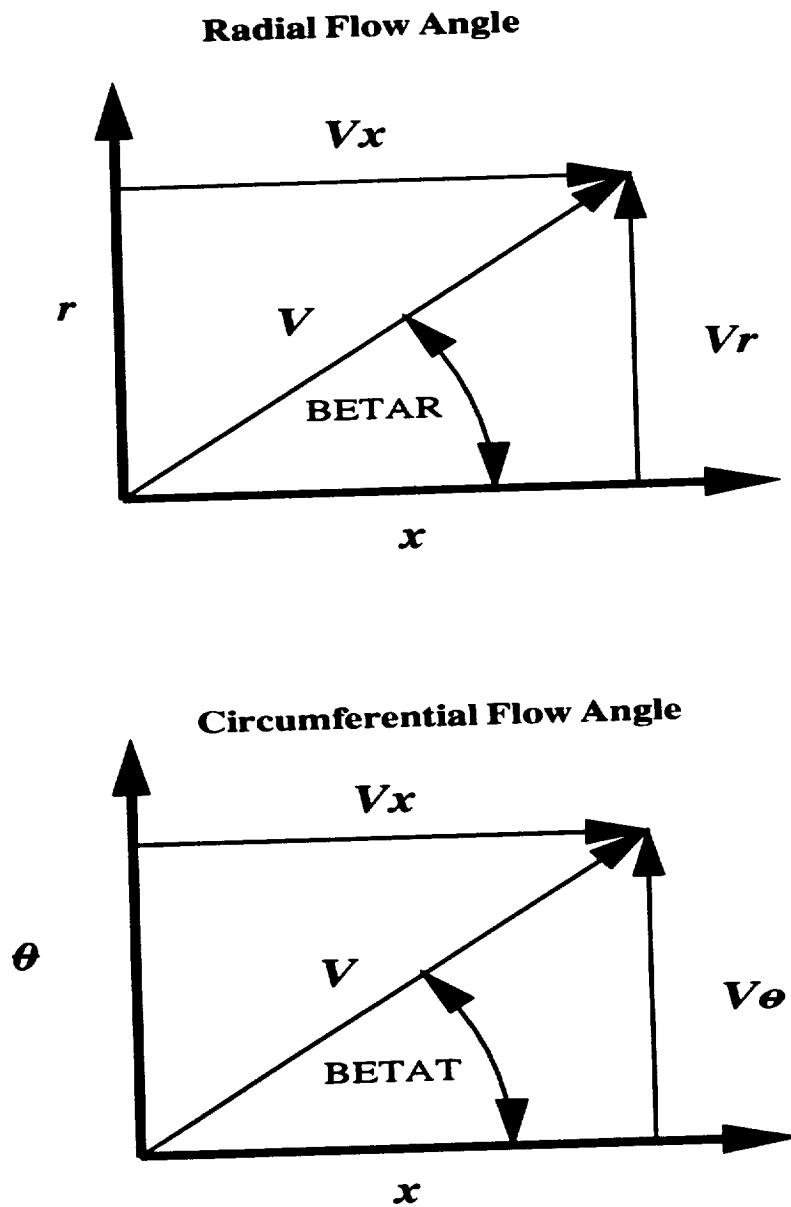
Figure 3.8:   *ADPAC-AOACR* Turbomachinery Flow Angle Reference Description

```
EXITG   1  1  I  I  M  M  L  L 125 125   1  25   1  17   1  25   1  17
  PEXIT
  0.8924
```

Description: The **EXITG** statement specifies that a subsonic, constant pres-
sure, generic exit flow boundary condition is to be applied to the
specified mesh surface. This boundary condition requires the spec-
ification of additional data, as shown in the Format descriptor
above. The value of the variable following **PEXIT** is the value
of the desired nondimensional exit static pressure applied at the
exit boundary surface defined as:

$$P_{exit,non-dimensional} = \frac{p_{exit}}{p_{ref}}$$

All remaining exit boundary flow values are extrapolated from the
interior of the solution domain.

Keyword: **EXT2DG**
Format:

```
EXT2DG  1  1  I  I  M  M  L  L 125 125   1  25   1   2   1  25   1   2
  PEXIT
  0.8924
```

Description: The **EXT2DG** statement specifies that a subsonic, constant pres-
sure, generic exit flow boundary condition is to be applied to the
specified two-dimensional mesh surface. This is the 2-D equivalent
of the 3-D boundary specification **EXITG** described above. This
boundary condition requires the specification of additional data, as
shown in the Format descriptor above. The value of the variable
following **PEXIT** is the value of the desired nondimensional exit
static pressure applied at the exit boundary surface defined as:

$$P_{exit,non-dimensional} = \frac{p_{exit}}{p_{ref}}$$

All remaining exit boundary flow values are extrapolated from the interior of the solution domain.

Keyword: **EXITT**

Format:

```
EXITT   1  1  I  I  M  M  L  L 125 125   1  25   1  17   1  25   1  17
   PEXIT
   0.8924
```

Description: The **EXITT** statement specifies that a subsonic, constant pressure, turbomachinery-type exit flow boundary condition utilizing radial equilibrium is to be applied to the specified mesh surface. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variable following **PEXIT** is the value of the desired nondimensional exit static pressure applied at the exit boundary surface defined as:

$$P_{exit, non-dimensional} = \frac{p_{exit}}{p_{ref}}$$

This exit static pressure is applied at either the upper or lower (radially) limit of the exit boundary surface. The remaining exit static pressures are integrated across the boundary by utilizing the radial equilibrium equation. The direction of integration, and location of application of the specified exit static pressure are determined by the **LSPEC1** variable in the calling sequence (see top of this section for a description of **LSPEC1**). If **LSPEC1 = L**, for LOW, then **PEXIT** is applied to the lower (smallest value) of the j index, and the radial equilibrium equation is integrated outward (increasing j direction). If **LSPEC1 = H**, for HIGH, then **PEXIT** is applied to the upper (largest value) of the j index, and the radial equilibrium equation is integrated inward (decreasing j direction). This boundary condition is typically used for turbomachinery flow calculations. All remaining exit boundary flow

86

values are extrapolated from the interior of the solution domain. Because of the radial integration, this boundary condition is only valid for i=constant or k=constant boundary surfaces, as the j is the assumed radial direction.

Keyword: **EXT2DT**
Format:

```
EXT2DT  1  1  I  I  M  M  L  L 125 125   1  25   1   2   1  25   1   2
  PEXIT
  0.8924
```

Description: The **EXT2DT** statement specifies that a subsonic, constant pressure, turbomachinery-type exit flow boundary condition utilizing radial equilibrium is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **EXITT** described above. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variable following **PEXIT** is the value of the desired nondimensional exit static pressure applied at the exit boundary surface defined as:

$$p_{exit,non-dimensional} = \frac{p_{exit}}{p_{ref}}$$

This exit static pressure is applied at either the upper or lower (radially) limit of the exit boundary surface. The remaining exit static pressures are integrated across the boundary by utilizing the radial equilibrium equation. The direction of integration, and location of application of the specified exit static pressure are determined by the **LSPEC1** variable in the calling sequence (see top of this section for a description of **LSPEC1**). If **LSPEC1** = L, for LOW, then **PEXIT** is applied to the lower (smallest value) of the j index, and the radial equilibrium equation is integrated outward (increasing j direction). If **LSPEC1** = H, for HIGH, then

87

PEXIT is applied to the upper (largest value) of the j index, and the radial equilibrium equation is integrated inward (decreasing j direction). This boundary condition is typically used for turbomachinery flow calculations. All remaining exit boundary flow values are extrapolated from the interior of the solution domain. Because of the radial integration, this boundary condition is only valid for i=constant or k=constant boundary surfaces, as the j is the assumed radial direction.

Keyword: **EXITP**

Format:

```
EXITP   1  1  I   I  M  M  L  L 125 125   1  25   1  17   1  25   1  17
NBOUN      IBOUN     JBOUN
   4        97         1
```

Description: The **EXITP** statement specifies that a patched subsonic, constant pressure, turbomachinery-type exit flow boundary condition utilizing radial equilibrium is to be applied to the specified mesh surface. This boundary condition is very similar to **EXITT**, above, except that the initial static pressure specification for the radial equilibrium equation is obtained from a neighboring grid. This specification is useful for turbomachinery-based calculations involving more than one mesh block across an exit boundary for which the radial equilibrium equation must be integrated. The *ADPAC-AOACR* code performs this function by using the **EXITT** specification to set the exit static pressure at some point, and then begin integrating the radial equilibrium equation (in the coordinate direction determined by LSPEC1 - see **EXITT**) until the end of the boundary is reached. The specification of **EXITP** then requires the code to extract the static pressure from the neighboring mesh block and continue the integration across the new mesh block. This procedure may be repeated as many times as is necessary to complete the

integration across the entire exit boundary. The **EXITP** boundary condition must be used in conjunction with boundary specification **EXITT** to impose an accurate integration of the radial equilibrium boundary condition across mutliple grid blocks. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variable following **NBOUN** represents the mesh block number from which the required static pressure data is extracted in order to complete the integration of the radial equilibrium equation. The variable following **IBOUN** and **JBOUN** represent the i and j indices in block **NBOUN** which correspond to the **L1LIM** exit plane in block **LBLOCK1**. The direction of integration, and location of application of the specified exit static pressure are determined by the **LSPEC1** variable in the calling sequence (see top of this section for a description of **LSPEC1**). If **LSPEC1** = L, for LOW, then the extracted exit static pressure is imposed at the lower (smallest value) of the j index, and the radial equilibrium equation is integrated outward (increasing j direction). If **LSPEC1** = H, for HIGH, then the extracted exit static pressure is imposed at the upper (largest value) of the j index, and the radial equilibrium equation is integrated inward (decreasing j direction).

Keyword: **EXT2DP**
Format:

```
EXT2DP   1  1  I  I  M  M  L  L 125 125   1  25   1   2   1  25   1   2
NBOUN      IBOUN      JBOUN
   4          97          1
```

Description: The **EXT2DP** statement specifies that a patched subsonic, constant pressure, turbomachinery-type exit flow boundary condition utilizing radial equilibrium is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **EXITT** described above. This boundary

condition is very similar to **EXT2DT**, above, except that the initial static pressure specification for the radial equilibrium equation is obtained from a neighboring grid. This specification is useful for turbomachinery-based calculations involving more than one mesh block across an exit boundary for which the radial equilibrium equation must be integrated. The *ADPAC-AOACR* code performs this function by using the **EXT2DT** specification to set the exit static pressure at some point, and then begin integrating the radial equilibrium equation (in the coordinate direction determined by **LSPEC1** - see **EXT2DT**) until the end of the boundary is reached. The specification of **EXT2DP** then requires the code to extract the static pressure from the neighboring mesh block and continue the integration across the new mesh block. This procedure may be repeated as many times as is necessary to complete the integration across the entire exit boundary. The **EXT2DP** boundary condition must be used in conjunction with boundary specification **EXT2DT** to impose an accurate integration of the radial equilibrium boundary condition across mutliple grid blocks. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variable following **NBOUN** represents the mesh block number from which the required static pressure data is extracted in order to complete the integration of the radial equilibrium equation. The variable following **IBOUN** and **JBOUN** represent the i and j indices in block **NBOUN** which correspond to the **L1LIM** exit plane in block **LBLOCK1**. The direction of integration, and location of application of the specified exit static pressure are determined by the **LSPEC1** variable in the calling sequence (see top of this section for a description of **LSPEC1**). If **LSPEC1** = L, for LOW, then the extracted exit static pressure is imposed at the lower (smallest value) of the j index, and the radial equilibrium equation is integrated outward (increasing j direction). If **LSPEC1** =

90

H, for HIGH, then the extracted exit static pressure is imposed at the upper (largest value) of the j index, and the radial equilibrium equation is integrated inward (decreasing j direction).

Keyword: **FREE**
Format:

```
FREE    1  1  I   I  M  M  L  L 125 125   1  25   1  17   1  25   1  17
  PTOT     TTOT     EMINF     ALPHA
  1.0      1.0       0.9       0.0
```

Description: The **FREE** statement specifies that a free flow external flow boundary condition is to be applied to the specified mesh surface. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variables following PTOT, TTOT are the desired nondimensional freestream upstream total pressure and temperature defined as:

$$(p_{total,upstream})_{non-dimensional} = \frac{p_{total,upstream}}{p_{ref}}$$

$$(T_{total,upstream})_{non-dimensional} = \frac{T_{total,upstream}}{T_{ref}}$$

The value of the variable EMINF is the freestream Mach number. The value of the variable ALPHA is the freestream angle of attack, measured in degrees from the reference x axis (see Figure 3.7). The angle of attack is applied to the flow in the x-z plane (Cartesian sense - positive angle of attack is flow pointing in the positive z direction, see Figure 3.7).

Keyword: **FRE2D**
Format:

```
FRE2D   1  1  I   I  M  M  L  L 125 125   1  25   1   2   1  25   1   2
  PTOT     TTOT     EMINF     ALPHA
  1.0      1.0       0.9       0.0
```

Description: The **FRE2D** statement specifies that a free flow external flow boundary condition is to be applied to the specified two-dimensional mesh surface. This is the 2-D equivalent of the 3-D boundary specification **FREE** described above. This boundary condition requires the specification of additional data, as shown in the Format descriptor above. The value of the variables following PTOT, TTOT are the desired nondimensional freestream upstream total pressure and temperature defined as:

$$(p_{total,upstream})non-dimensional = \frac{p_{total,upstream}}{p_{ref}}$$

$$(T_{total,upstream})non-dimensional = \frac{T_{total,upstream}}{T_{ref}}$$

The value of the variable EMINF is the freestream Mach number. The value of the variable ALPHA is the freestream angle of attack, measured in degrees from the reference x axis (see Figure 3.7). The angle of attack is applied to the flow in the x-z plane (Cartesian sense - positive angle of attack is flow pointing in the positive z direction, see Figure 3.7).

Keyword: **KILL**
Format:

```
KILL    1   1   I   I   M   M   L   L  125  125   1   25   1   17   1   25   1   17
   LSTART    LEND
       41       61
```

Description: The **KILL** statement is a tool to effectively neutralize or "kill" the time-marching solution over a three-dimensional segment of the computational domain. In cases where a portion of the mesh does not represent a valid flow region, the KILL specification can be used, in conjunction with boundary conditions specified about the region to be "killed", to effectively remove a portion of a given mesh block from the computational domain. An example of this

technique is illustrated in Figure 3.9. The figure depicts a single block mesh for the flow through a simple nozzle. Suppose that for whatever reason, the user wished to remove an internal rectangular portion of the mesh (as if there were an obstruction placed in the flowpath). This could be accomplished by subdividing the original mesh into several smaller pieces, and applying the appropriate boundary conditions along the outer boundaries of each block. This same configuration could also be modeled using the original mesh by invoking the **KILL** specification for the points inside the obstruction, and applying the proper boundary specifications along the obstruction internally on the single-block mesh. This boundary condition (although really, this is more than a boundary condition) requires the specification of additional data, as shown in the format descriptor above. The variable following the label **LSTART** indicates the starting index in the I coordinate direction for the region to be "killed". The variable following the label **LEND** indicates the final index in the I direction for the region to be "killed". The remaining coordinate indices for the region to be "killed" are determined by the variables **M1LIM1**, **M1LIM2** for the J coordinate direction and **N1LIM1**, and **N1LIM2** for the K coordinate direction. The additional specification of the **LSTART**, **LEND** variables imply that the variables **L1LIM**, **L2LIM** are not used in this specification.

Keyword: **KIL2D**

Format:

```
KIL2D   1   1   I   I   M   M   L   L 125 125    1  25    1   2   1  25    1   2
     LSTART     LEND
        41        61
```

Description: The **KIL2D** statement is a tool to effectively neutralize or "kill" the time-marching solution over a segment of the computational domain for a two-dimensional mesh. This is the 2-D equivalent of

93

ADPAC–AOACR Solution Killing Illustration

Physical Domain
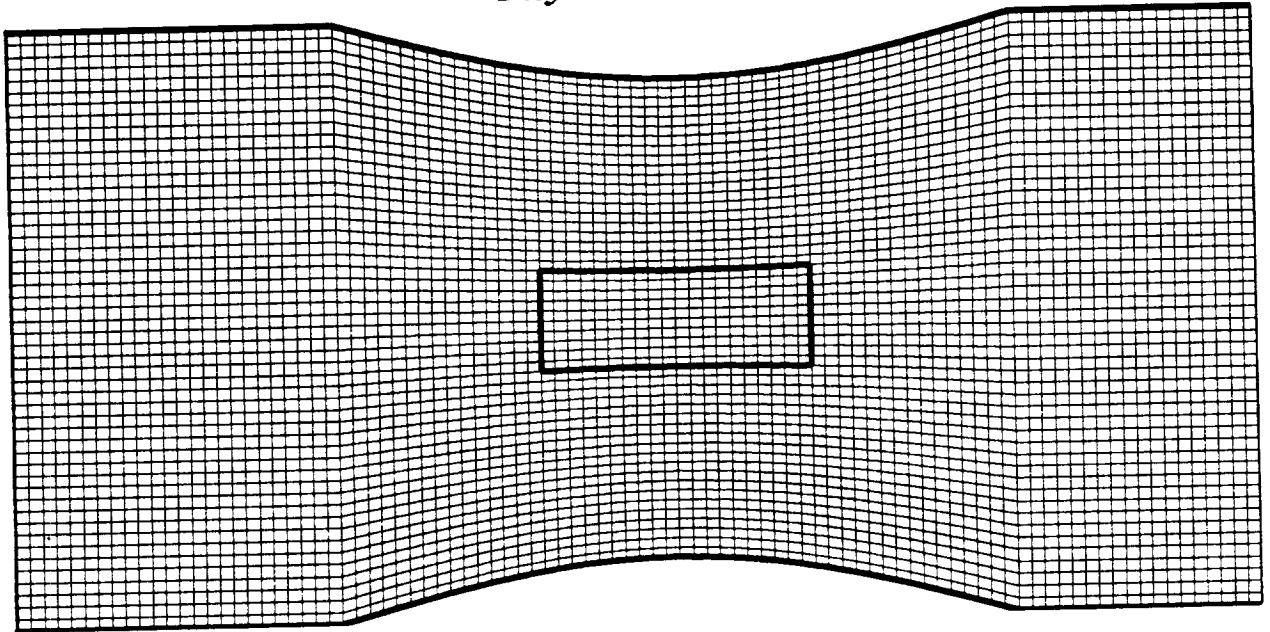
Predicted Mach Number Contours
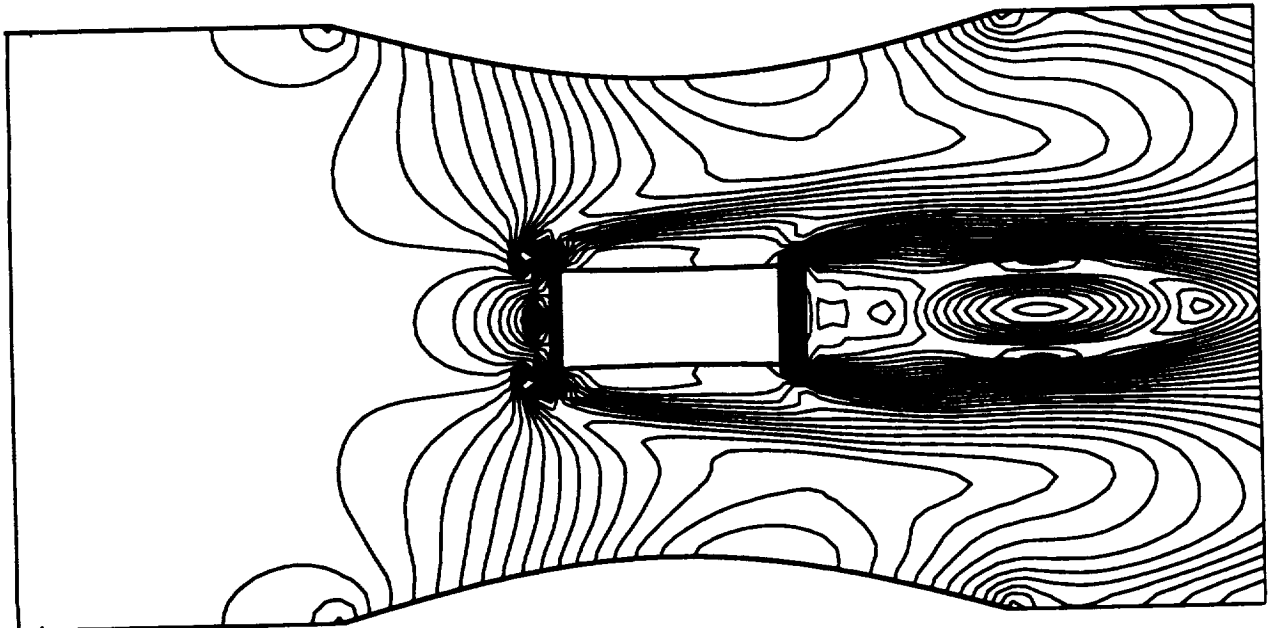
Figure 3.9: *ADPAC-AOACR* Solution Killing Reference Figure

the 3-D boundary specification **KILL** described above. In cases where a portion of the mesh does not represent a valid flow region, the **KILL** specification can be used, in conjunction with boundary conditions specified about the region to be "killed", to effectively remove a portion of a given mesh block from the computational domain. An example of this technique is illustrated in Figure 3.9. The figure depicts a single block mesh for the flow through a simple nozzle. Suppose that for whatever reason, the user wished to remove an internal rectangular portion of the mesh (as if there were an obstruction placed in the flowpath). This could be accomplished by subdividing the original mesh into several smaller pieces, and applying the appropriate boundary conditions along the outer boundaries of each block. This same configuration could also be modeled using the original mesh by invoking the **KILL** specification for the points inside the obstruction, and applying the proper boundary specifications along the obstruction internally on the single-block mesh. This boundary condition (although really, this is more than a boundary condition) requires the specification of additional data, as shown in the format descriptor above. The variable following the label **LSTART** indicates the starting index in the I coordinate direction for the region to be "killed". The variable following the label **LEND** indicates the final index in the I direction for the region to be "killed". The remaining coordinate indices for the region to be "killed" are determined by the variables **M1LIM1, M1LIM2** for the J coordinate direction and **N1LIM1,** and **N1LIM2** for the K coordinate direction. The additional specification of the **LSTART, LEND** variables imply that the variables **L1LIM, L2LIM** are not used in this specification.

Keyword: **PATCH**
Format:

```
PATCH   1   1   K   K   P   M   I   J   1  17   1 129   1  17   1 129   1  17 %K=1 Per
```

95

Description:　The **PATCH** statement is utilized to provide direct block to block communication between mesh blocks with contiguous grid points. This is perhaps the most common, and most useful of the boundary condition specifications, and therefore, a lengthy discussion is given to complete this description. For many complicated geometries requiring a multiple block mesh system, a common approach is to generate mesh systems with coincident mesh points along all, or at least part of the mesh block interfaces. This property is henceforth referred to as a contiguous mesh block interface (coincident mesh points). By default, then, the boundary condition specification must have a one-to-one correspondence between mesh points in block **LBLOCK1** and mesh points in block **LBLOCK2**. This type of boundary is particularly effective in the finite-volume flow solver due to the fact that local and global conservation of the flow variables can be accomplished without special treatment, by direct substitution of the neighboring block flow variables into the phantom cells of the block of interest. The **PATCH** boundary condition performs this direct substitution between blocks to provide an aerodynamic communication between neighboring blocks with a contiguous interface. A **PATCH** specification can also be imposed connecting a block to itself. In fact, this is the manner by which spatial periodicity is enforced in many cases, including the Standard Configurations described in Chapter 5. The **PATCH** boundary condition requires no additional data beyond the initial specification line, but does require the proper specification of the variables **LSPEC1** and **LSPEC2**. For boundary conditions involving more than one mesh block (such as **PATCH**), it is possible that the connection between blocks may involve communication between different grid surfaces (for example, an i=constant mesh face in **LBLOCK1** connects to a j=constant mesh face in **LBLOCK2**) and that the remaining indices in block **LBLOCK2** correspond to different coordinates in block **LBLOCK1**. The

specification of the variables **LSPEC1**, **LSPEC2** serve to eliminate any confusion between contiguous boundary patches involving dissimilar mesh coordinates. In every case, when a particular coordinate direction is specified by the variable **LFACE1**, the remaining coordinate indices defining the extent of the patch on **LFACE1** are specified in their "natural" (i, j, k) order. For example, if **LFACE1** is an i=constant mesh surface, then the variables **M1LIM1**, **M1LIM2** control the indices in the j coordinate direction and the variables **N1LIM1**, **N1LIM2** control the indices in the k coordinate direction. Similarly, if **LFACE1** is a k=constant mesh surface, then the variables **M1LIM1**, **M1LIM2** control the indices in the i coordinate direction and the variables **N1LIM1**, **N1LIM2** control the indices in the j coordinate direction, and so on. Now, in order to relate the coordinate indices in block **LBLOCK2** with the indices specified in block **LBLOCK1**, the special terms **LSPEC1** and **LSPEC2** are utilized. The variables **LSPEC1** and **LSPEC2** should be defined as either **I**, **J**, or **K**, based on the connection scheme between the two blocks. The **LSPEC1** variable should define the coordinate direction in block **LBLOCK2** which corresponds to the first remaining coordinate in block **LBLOCK1** (whose range is defined by **M1LIM1**, **M1LIM2**, and the **LSPEC2** variable should define the coordinate direction in block **LBLOCK2** which corresponds to the second remaining coordinate in block **LBLOCK1** (whose range is defined by **N1LIM1**, **N1LIM2**. The **PATCH** specification may also be used for two-dimensional meshes as long as the third coordinate direction (k) limits **N1LIM1**, **N1LIM2**, and **N2LIM1**, **N2LIM2** are "1" and "2", respectively (2-D patches are specified as if the mesh were actually 2 cells deep in the k direction).

Keyword: **PINT**

Format:

```
PINT    1  1  I  I  M  M  L  L 125 125   1  25   1   2   1  25   1   2
```

Description: The **PINT** boundary statement provides a means for block to block communication for cases involving neighboring meshes which share a common surface, but not necessarily common grid points along a block boundary. The **PINT** specification instructs the *ADPAC-AOACR* code to perform a weighted interpolation to determine the appropriate flow variables for the phantom cells, based on the noncontiguous data structure of the neighboring mesh. An example of this type of boundary is given in Figure 2.2. The bounding surfaces of each block should lie on a common surface (no significant overlap). The interpolation scheme used in the **PINT** specification is not conservative, and therefore the solution accuracy can be degraded by this procedure. The remaining terms in the **PINT** boundary specification are given in their "natural" order (i, j, k). During code execution, the first time the **PINT** specification is encountered, the code initiates a search to determine the interpolation stencil for the given array of points in block **LBLOCK1** based on the data in block **LBLOCK2**. This stencil is then saved to eliminate the search routine at every application of **PINT**. In order to provide storage for the interpolation stencil information, a separate array system based on the dimensioning parameter **NPINT** (see Section 3.3) is utilized.

Keyword: **BCPRR**

Format:

```
BCPRR   1  2  I  I  M  P  J  K  21   1   1   5   1   7   1   5   1   7
THPER
0.523598775
NBCPRR
4
NRRDAT
4
5
```

Description: The **BCPRR** statement specifies that a time-space interpolation utilizing data from several neighboring mesh blocks is to be performed to determine the boundary data for block **LBLOCK1**. This time-space interpolation provides the computational means of performing time-dependent predictions of the flow through multiple blade row turbomachines (see the discussion in Section 2.2). In order to perform this type of calculation, several conditions must be satisfied. For calculations involving blade rows with dissimilar blade counts, it is necessary to model several blade passages per blade row. The number of blade passages modeled should be chosen such that the overall circumferential span of each blade row is identical. This implies that the blade counts should be reducible to simple integer ratios (1:2, 3:4, etc.) to avoid the need for modeling an excessive number of blade passages. For example, if we seek a solution for a single stage turbomachine involving two blade rows with blade counts of 36 and 48, respectively (reduced blade ratio of 3:4), then the simulation would require 3 blade passages for the first blade row and 4 passages from the second blade row, such that the overall circumferential pitch for either blade row is $\frac{2\pi}{12}$ (the number 12 chosen as the largest common factor in the blade counts 36 and 48). The second restriction is that interface separating two adjacent blade rows be a surface of revolution, and that meshes along this interface have common axial and radial grid distributions. This restriction simplifies the time-space interpolation provided by the **BCPRR** specification. This boundary condition requires the specification of additional data, as shown in the format descriptor above. The variable following the label **THPER** defines the total circumferential span of the neighboring blade row's mesh representation in radians. For example, using the blade counts given in the previous example, the circumferen-

tial span represented in each blade row is determined by $\frac{2\pi}{12}$, and therefore **THPER** should be 0.523598775. The variable following the next label, **NBCPRR**, indicates the number of mesh blocks through which the time-space interpolation is to be performed. In the example above, if we are applying the **BCPRR** specification to the first blade row, then **NBCPRR** should be 4, since there are 4 mesh blocks in the neighboring blade row. The numbers immediately following the label NRRDAT are the **NBCPRR** block numbers across which the interpolation schould be performed. In the example described above, if block numbers 1, 2, and 3 are the block numbers for the first blade row, and block numbers 4, 5, 6, and 7 are the block numbers for the second blade row, then the **BCPRR** specification for each of the first blade row blocks would set **THPER** = 0.523598775, **NBCPRR** = 4, and **NRRDAT** = 4, 5, 6, 7. In a similar manner, the specification for each of the blocks in the second blade row would set **THPER** = 0.523598775, **NBCPRR** = 3, and **NRRDAT** = 1, 2, 3. It should be mentioned that this specification is somewhat unique in that more than one block is involved in the boundary specification, therefore the variable **LBLOCK2** is essentially ignored; however, since the blocks specified by **NRRDAT** are assumed to be essentially duplicate representations of neighboring blade passages, the variables **L2LIM, M2LIM1, M2LIM2, N2LIM1,** and **N2LIM2** are assumed to apply to each of the **NRRDAT** blocks. The time-space interpolation is constructed to permit the relative rotation of blocks representing neighboring blade rows. The simulation is initiated from the relative position of the blocks at the start of the calculation $t=0$. The interpolation scheme is area weighted to maintain a conservative property across the interface between the relatively rotating mesh blocks (see the Final Report for additional details on the implementation of this boundary procedure).

Keyword: **CIRAV**
Format:

```
CIRAV  1  3  I  I  M  P  J  K 129  1  1  17  1  17  1  17  1  17
```

Description: The **CIRAV** boundary statement specifies that a circumferentially averaged boundary specification is to be performed on block **LBLOCK1** using data from block **LBLOCK2**. This boundary procedure provides the computational means of imposing the mixing plane concept for multiple blade row calculations discussed in Section 2.2. The mixing plane is a surface of revolution which defines the interface between two adjacent mesh blocks (also, normally, two blade rows). At this interface, the meshes must have a common axial and radial distribution of points. The circumferential averaging is performed using a simple area average of the flow variables at each radial mesh index in block **LBLOCK2**, and imposing the averaged flow variables uniformly in the corresponding radial array of phantom cells in block **LBLOCK1**. In practice, it is possible that the location of the mixing plane may influence the final solution. In cases where the spacing between adjacent blade rows is very small, the mixing plane solution technique may yield questionable results, as the true flow is likely to be highly unsteady. The **CIRAV** specification is currently only valid for **LFACE1**, **LFACE2** values of *I* or *J*.

Keyword: **MBCAVG**
Format:

```
MBCAVG  1  2  J  J  M  P  I  K  25  1  9  65  1  17  1  57  1  9
NBL
2
```

BLKNO	LLIM	M2LIM1	M2LIM2	N2LIM1	N2LIM2
2	1	1	57	1	9
3	1	1	57	1	9

101

Description: The **MBCAVG** boundary statement specifies that a multiple block circumferentially averaged boundary specification is to be performed on block **LBLOCK1**. This boundary condition is functionally equivalent to the **CIRAV** routine described above, except that the circumferential average may be performed across more than one mesh block. This boundary condition requires the specification of additional data, as shown in the format descriptor above. The variable following the label **NBL** defines the number of blocks across which the circumferential average is performed. The remaining lines following the descriptors describe the neighboring block interfaces across which the circumferential average is performed. **BLOCKNO** indicates the neighboring mesh block number for each of the **NBL** blocks. **LLIM** is the mesh index in block **BLKNO** in the **LFACE2** direction across which the circumferential average is performed. The variables **M2LIM1, M2LIM2** and **N2LIM1, N2LIM2** represent the extent of the averaging procedure in the remaining (non-**LFACE2**) co-ordinate directions (e.g., if **LFACE2**=I, then the additional **M2LIM1, M2LIM2** variables represent the extent in the J direction, and **N2LIM1, N2LIM2** variables represent the extent in the K direction for the block indicated by **BLKNO**. This boundary procedure can be used to provide the computational means of imposing the mixing plane concept for multiple blade row calculations discussed in Section 2.2. At this interface, the meshes must have a common axial and radial distribution of points. The circumferential averaging is performed using a simple area average of the flow variables at each radial mesh index in each of the **NBL** blocks, and imposing the averaged flow variables uniformly in the corresponding array of phantom cells in block **LBLOCK1**. In practice, it is possible that the location of the mixing plane may influence the final solution. The **MBCAVG** specification is currently only valid for **LFACE1, LFACE2** values of I or J.

3.8   Mesh File Description

The *ADPAC-AOACR case*.mesh file is a data file containing the x, y, z grid co-ordinates of the multiple mesh blocks which are read in to define the physical grid points used in the time-marching solution (see Section 3.5 for a description of the *case* name and the mesh file naming convention). The mesh coordinates are specified in a

Cartesian frame of reference, as shown in Figure 3.10, although the *ADPAC-AOACR* program ultimately converts these coordinates to a cylindrical coordinate system during execution. The mesh coordinates are stored in what is known as *PLOT3D* multiple grid format, and are formatted using the Scientific Database Library (SDBLIB). (The SDBLIB system allows machine-independent binary file storage.) The *case*.mesh file **must** be available for every *ADPAC-AOACR* run. At the beginning of program execution, the *ADPAC-AOACR* program attempts to open the mesh file and read in the mesh size to make sure that enough memory has been allocated for the given problem. If the mesh file is not found, or if the mesh is too large, the appropriate error message is issued, and the program will terminate.

Mesh coordinates **must** be specified as nondimensional numbers. Typically, the length scale used to nondimensionalize the coordinates is the maximum diameter of all the blade rows involved in a given solution. Proper nondimensionalization is required in order to accurately achieve the desired flow Reynolds number and rotational speed (see the discussion of input variable **ADVR** is Section 3.6). It is also required that the ordering of the mesh points form a "left handed" mesh. This implies that at every point in the mesh, the vectors representing the positive i, j, and k coordinate directions form a left-handed coordinate system (see Figure 3.11). Consider the case of a sheared H-grid discretizing a single blade passage of a compressor (this type of mesh is used extensively in the Standard Configurations described in Chapter 5). If we assume that looking downstream through the blade passage is essentially the positive i direction, and that the radial direction from hub to tip is essentially the positive j direction, a left-handed mesh would require that the positive k direction be from right to left in this orientation.

In order to understand the *PLOT3D* multiple-grid mesh file format, and the utilization of the *SDBLIB* routines, a comparison of the FORTRAN coding for each method is given below for comparison.

The FORTRAN coding to read a *PLOT3D* unformatted multiple-block mesh file might be given as:

PLOT3D Mesh File Format FORTRAN Coding Example

```
OPEN(UNIT=IGRID,FILE=FNAME,FORM='UNFORMATTED',STATUS='OLD')
```

ADPAC-AOACR Coordinate System Reference
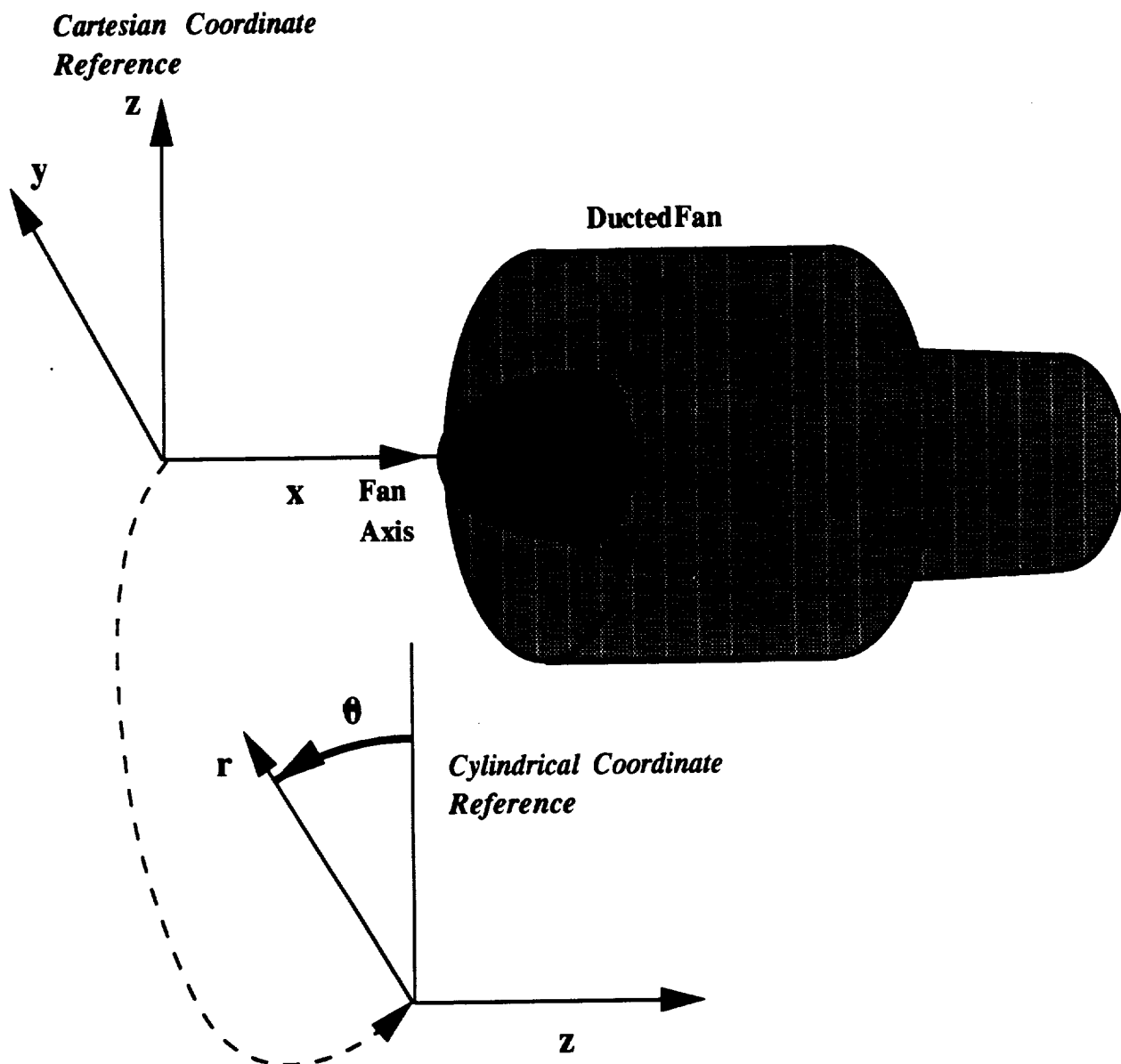
Figure 3.10: *ADPAC-AOACR* Mesh Coordinate Reference Description

ADPAC-AOACR Left-Handed Coordinate Description

All ADPAC-AOACR Mesh Blocks Must Be Based on a Left Handed Indexing System

Left-Handed Mesh System
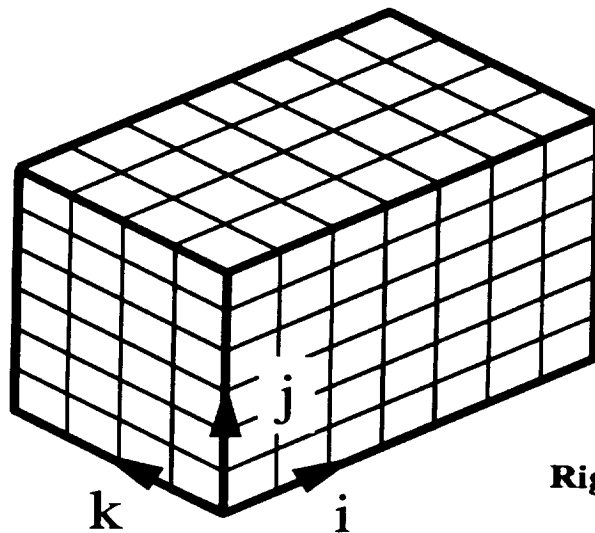
Right-Handed Mesh System
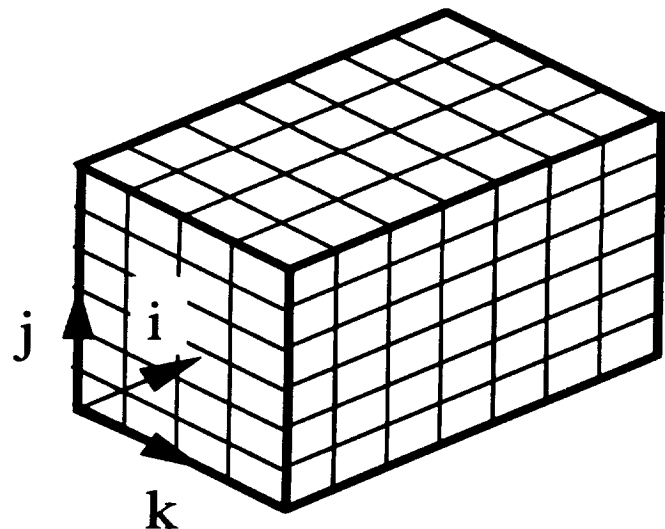
Figure 3.11: *ADPAC-AOACR* Left-Handed Coordinate System Description

```
      READ(IGRID) MG
      READ(IGRID) (IL(L), JL(L), KL(L),L=1,MG)
      DO 10 L = 1, MG
      READ(IGRID) (((X(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L)),
                  (((Y(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L)),
                  (((Z(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L))
10    CONTINUE
```

Each of the terms used in the FORTRAN code given above are defined below:

$$
\begin{array}{ll}
\text{IGRID} & \text{FORTRAN unit number for read statement} \\
\text{FNAME} & \text{File name for mesh file} \\
\text{MG} & \text{number of grid blocks} \\
\text{IL(L)} & \text{maximum } i \text{ grid index for block L} \\
\text{JL(L)} & \text{maximum } j \text{ grid index for block L} \\
\text{KL(L)} & \text{maximum } k \text{ grid index for block L} \\
\text{X(I,J,K,L)} & \text{Cartesian coordinate value of x for point (I,J,K) in block L} \\
\text{Y(I,J,K,L)} & \text{Cartesian coordinate value of y for point (I,J,K) in block L} \\
\text{Z(I,J,K,L)} & \text{Cartesian coordinate value of z for point (I,J,K) in block L}
\end{array}
$$

An example of the corresponding FORTRAN coding to read an *ADPAC-AOACR* binary mesh file using the Scientific Database Library (SDBLIB) routines is given below:

PLOT3D Mesh File Format FORTRAN Coding Example Using SDBLIB

```
      CALL QDOPEN( IGRID, FNAME, JE )
      CALL QDGETI( IGRID, MG    , JE )
      ILENGTH = 3 * MG
      CALL QDGEIA( IGRID, IB, ILENGTH, JE )
      DO 10 L = 1, MG
      IL(L) = IB((L-1)*3+1)
      JL(L) = IB((L-1)*3+2)
      KL(L) = IB((L-1)*3+3)
      ILENGTH = IL(L) * JL(L) * KL(L)
      CALL QDGEEA( IGRID, X(IPOINT(L)), ILENGTH, JE )
      CALL QDGEEA( IGRID, Y(IPOINT(L)), ILENGTH, JE )
      CALL QDGEEA( IGRID, Z(IPOINT(L)), ILENGTH, JE )
```

```
10    CONTINUE
      CALL QDCLOS( IGRID, JE )
```

A listing of the additional terms used in the coding above is given below:

QDOPEN	SDBLIB routine to open a file for input or output
QDGETI	SDBLIB routine to get an integer
QDGEIA	SDBLIB routine to get an integer array of length ILENGTH
QDGETE	SDBLIB routine to get a real number
QDGEEA	SDBLIB routine to get a real array of length ILENGTH
QDCLOS	SDBLIB routine to close a file
IGRID	FORTRAN logical unit number for grid input
JE	An error trigger; 0 for no error, 1 if an error occurs
IB	Integer array containing the IL, JL, and KL grid block sizes
ILENGTH	Integer length of an array of data
IPOINT(L)	Integer pointer for block L to locate the initial memory location for a block of data

The x, y, z coordinates are read in as a single-dimensioned array in the *SDBLIB* format, and the *ADPAC-AOACR* program includes a conversion routine (source file *convas.f*) which converts the single dimension array data to a three-dimensional data array.

The mesh file may be utilized directly with the *PLOT3D* program when the default real number size of the compiled *PLOT3D* code is defined as 32 bits (as it is on many workstations). The corresponding *PLOT3D* read command for an *ADPAC-AOACR* mesh file are:

```
PLOT3D PROMPT> read/mg/x=case.mesh
```

Obviously the user should substitute their own case name in the *PLOT3D* input line.

Unformatted mesh files may be converted to *ADPAC-AOACR* format using the *MAKEADGRID* program described in Chapter 7.

3.9 Body Force File Description

The *ADPAC-AOACR* body force file is a data file containing the blade blockage, body force, and energy source terms used in a 2-D axisymmetric representation of an embedded blade row (see 2-D/3-D Solution Concepts, Section 2.3). Individual body force files contain the cell-centered blade blockage, body forces, and energy source terms for a specific mesh block. As a result, the file naming procedure for the body force file is somewhat different than the mesh, plot and restart files, where a single file contains all the data for a multiple-block solution (a complete description of the *ADPAC-AOACR* file naming procedure is given in Section 3.5).

The terms in the body force file are stored in binary format, based on the Scientific Database Library routines. (The SDBLIB system permits machine-independent binary file storage.) The blockage, body forces, and energy sources are stored as nondimensional numbers using the nondimensionalization strategy listed in Section 1.2 of the Final Report [1].

In order to understand the body force file format, and the utilization of the *SDBLIB* routines, a representative FORTRAN coding example to read in a body force file is given below for comparison.

Body Force File Format FORTRAN Coding Example Using SDBLIB

```
CALL QDOPEN( IBODY, FNAME, JE )
ILENGTH = 3
CALL QDGEIA( IBODY, IB, ILENGTH, JE )
IMX = IB(1)
JMX = IB(2)
KMX = IB(3)
ILENGTH = IMX * JMX * KMX
CALL QDGETE( IBODY, DUMMY, JE )
CALL QDGETE( IBODY, DUMMY, JE )
CALL QDGETE( IBODY, DUMMY, JE )
CALL QDGETE( IBODY, DUMMY, JE )
CALL QDGEEA( IBODY, BFR  (IPOINT(L)), ILENGTH, JE )
CALL QDGEEA( IBODY, BFRU (IPOINT(L)), ILENGTH, JE )
CALL QDGEEA( IBODY, BFRV (IPOINT(L)), ILENGTH, JE )
CALL QDGEEA( IBODY, BFRW (IPOINT(L)), ILENGTH, JE )
```

```
CALL QDGEEA( IBODY, BFRE (IPOINT(L)), ILENGTH, JE )
CALL QDGEEA( IBODY, BL   (IPOINT(L)), ILENGTH, JE )
CALL QDCLOS( IBODY, JE )
```

A listing of the FORTRAN variables and their meanings is given below:

QDOPEN	SDBLIB routine to open a file for input or output
QDGETI	SDBLIB routine to get an integer
QDGETE	SDBLIB routine to get a real number
QDGEIA	SDBLIB routine to get an integer array of length ILENGTH
QDGEEA	SDBLIB routine to get a real array of length ILENGTH
QDCLOS	SDBLIB routine to close a file
IBODY	FORTRAN logical unit number for body force file input
JE	An error trigger; 0 for no error, 1 if an error occurs
IB	Integer array containing the IL, JL, and KL grid block sizes
IMX	Mesh size+1 in the i coordinate direction
JMX	Mesh size+1 in the j coordinate direction
KMX	Mesh size+1 in the k coordinate direction
ILENGTH	Integer length of an array of data
IPOINT(L)	Integer pointer for block L to locate the initial memory location for a block of data
BFR	Body force for density (continuity equation)
BFRU	Body force for axial momentum
BFRV	Body force for radial momentum
BFRW	Body force for circumferential momentum
BFRE	Body force for internal energy
BL	Blockage term

The body force data are read and written as a single-dimensioned array in the *SDBLIB* format, and the *ADPAC-AOACR* program includes a conversion routine (source file *convas.f*) which converts the three-dimensional array data to the single dimension array data.

109

3.10 Standard Output File Description

The *ADPAC-AOACR* standard output file *case*.output provides information regarding the status of a particular calculation during code execution. The status information includes startup, code response to input files (mesh, restart, standard input, and boundary data), convergence history, error messages, and output file generation (plot files, restart files, body force files). The information given in the standard output file is essentially self explanatory, so no further description is given here. A sample output file is included in the standard distribution of the *ADPAC-AOACR* code for the test case described in Appendix A. Additional details may be found in this Appendix.

3.11 Plot File Description

The *ADPAC-AOACR case*.p3dabs and *case*.p3drel plot files contain predicted absolute and relative flow data values, respectively, for each of the mesh points in a multiple-block mesh *ADPAC-AOACR* solution. The grid-centered aerodynamic data is obtained by algebraically averaging the cell-centered data generated by the finite-volume solver during the time-marching process. As a result of the averaging procedure, this data can occasionally appear inconsistent at the corners of a mesh block, and should therefore only be used for graphical viewing, and not for post processing to obtain performance data, mass flow rates, pressure rise, etc. The flow plot data are specified in a Cartesian coordinate system (velocities are u_x, u_y, u_z) to be consistent with the representation of the mesh file (see Section 3.8). The plot files are written in what is known as *PLOT3D* multiple grid binary format. The plot data are formatted using the Scientific Database Library (SDBLIB). (The SDBLIB system permits machine-independent binary file storage.) The flow data are listed as nondimensional numbers using the nondimensionalization strategy listed in Section 1.2 of the Final Report [1].

In order to understand the *PLOT3D* multiple-grid flow file format, and the utilization of the *SDBLIB* routines, a comparison of the FORTRAN coding for each method is given below for comparison.

The equivalent FORTRAN coding for an unformatted *PLOT3D* flow file could be given as:

PLOT3D Flow File Format FORTRAN Coding Example

```
      WRITE( ) MG
      WRITE( ) (IL(L), JL(L), KL(L),L=1,MG)
      DO 20 L = 1, MG
      WRITE( ) EM(L), REY(L), ALF(L), TIME(L)
      WRITE( ) (((R (I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L)),
     .           (((RU(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L)),
     .           (((RV(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L)),
     .           (((RW(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L)),
     .           (((RE(I,J,K,L),I=1,IL(L)),J=1,JL(L)),K=1,KL(L))
 20   CONTINUE
```

Each of the terms used in the FORTRAN code given above are defined below:

MG	number of grid blocks
IL(L)	maximum i grid index for block L
JL(L)	maximum j grid index for block L
KL(L)	maximum k grid index for block L
X(I,J,K,L)	Cartesian coordinate value of x for point (I,J,K) in block L
Y(I,J,K,L)	Cartesian coordinate value of y for point (I,J,K) in block L
Z(I,J,K,L)	Cartesian coordinate value of z for point (I,J,K) in block L
EM(L)	*PLOT3D* Reference Mach number for block L
REY(L)	*PLOT3D* Reference Reynolds number for block L
ALF(L)	*PLOT3D* Reference angle for block L
TIME(L)	*PLOT3D* Reference time for block L
R (I,J,K,L)	ρ at point (I,J,K) in block L
RU(I,J,K,L)	ρu_x at point (I,J,K) in block L
RV(I,J,K,L)	ρu_y at point (I,J,K) in block L
RW(I,J,K,L)	ρu_z at point (I,J,K) in block L
RE(I,J,K,L)	ρe at point (I,J,K) in block L

PLOT3D Flow File Format FORTRAN Coding Example Using SDBLIB

```
        CALL QDOPEN( IFLOW, FNAME, JE )
        CALL QDGETI( IFLOW, MG    , JE )
        ILENGTH = 3 * MG
        CALL QDGEIA( IFLOW, IB, ILENGTH, JE )
        DO 20 L = 1, MG
        IL(L) = IB((L-1)*3+1)
        JL(L) = IB((L-1)*3+2)
        KL(L) = IB((L-1)*3+3)
        ILENGTH = IL(L) * JL(L) * KL(L)
        CALL QDGETE( IFLOW, EM(L)  , JE )
        CALL QDGETE( IFLOW, REY(L) , JE )
        CALL QDGETE( IFLOW, ALF(L) , JE )
        CALL QDGETE( IFLOW, TIME(L), JE )
        CALL QDGEEA( IFLOW, R (IPOINT(L)), ILENGTH, JE )
        CALL QDGEEA( IFLOW, RU(IPOINT(L)), ILENGTH, JE )
        CALL QDGEEA( IFLOW, RV(IPOINT(L)), ILENGTH, JE )
        CALL QDGEEA( IFLOW, RW(IPOINT(L)), ILENGTH, JE )
        CALL QDGEEA( IFLOW, RE(IPOINT(L)), ILENGTH, JE )
20      CONTINUE
        CALL QDCLOS( IFLOW, JE )
```

A listing of the additional terms used in the coding above is given below:

QDOPEN	SDBLIB routine to open a file for input or output
QDGETI	SDBLIB routine to get an integer
QDGEIA	SDBLIB routine to get an integer array of length ILENGTH
QDGETE	SDBLIB routine to get a real number
QDGEEA	SDBLIB routine to get a real array of length ILENGTH
QDCLOS	SDBLIB routine to close a file
IFLOW	FORTRAN logical unit number for flow input
JE	An error trigger; 0 for no error, 1 if an error occurs
IB	Integer array containing the IL, JL, and KL grid block sizes
ILENGTH	Integer length of an array of data
IPOINT(L)	Integer pointer for block L to locate the initial memory location for a block of data

The flow data are written as a single-dimensioned array in the *SDBLIB* format,

and the *ADPAC-AOACR* program includes a conversion routine (source file *convas.f*) which converts the three-dimensional array data to the single dimension array data.

The plot files may be utilized directly with the *PLOT3D* program when the default real number size of the compiled *PLOT3D* code is defined as 32 bits (as it is on many workstations). The corresponding *PLOT3D* read commands for an *ADPAC-AOACR* mesh and flow file are:


```
PLOT3D PROMPT> read/mg/x=case.mesh/q=case.p3dabs
```


Obviously the user should substitute their own case name in the *PLOT3D* input line.

3.12   Restart File Description

The *ADPAC-AOACR* restart file is a data file containing the cell-centered flow variables generated during an *ADPAC-AOACR* solution. This file is intended to permit continued execution of the code from the point at which a previous calculation was terminated. This feature permits breaking large jobs into smaller computational pieces. This process of job restarting is considered a good practice to avoid loss of data due to computer malfunctions and job quota limitations. At the end of a given job, whether the calculation is a restart run or not, the *ADPAC-AOACR* program will attempt to write out the current cell centered data to the file *case.restart.new* (see Section 3.2 for a description of the file naming convention). The restart file may then be used to continue the calculation at this same point by simply renaming the file *case.restart.new* to *case.restart.old*, setting the input trigger appropriately (see the description of **FREST** in Section 3.6), and rerunning the code. The restart data are written in the cylindrical coordinate system variable format used during execution of the *ADPAC-AOACR* code. Velocities are specified as u_z, u_r, u_θ, and all flow variables utilize the nondimensionalization strategy described in Section 1.2 of the companion Final Report [1].

In order to demonstrate the format of the restart file, a sample of the FORTRAN coding utilizing the *SDBLIB* library required to read a restart file is given below.

113

```
      call qdopen( irest, fname, je )
      call qdgeti( irest, mg    , je )
      nlength = 3 * mg
      call qdgeia( irest, ib, nlength, je )
        do 10 n = 1, mg
        imx(n) = ib((n-1)*3+1)
        jmx(n) = ib((n-1)*3+2)
        kmx(n) = ib((n-1)*3+3)
        length = imx(n) * jmx(n) * kmx(n)
        call qdgeea( irest, r (ijk(n)), length, je )
        call qdgeea( irest, ru(ijk(n)), length, je )
        call qdgeea( irest, rv(ijk(n)), length, je )
        call qdgeea( irest, rw(ijk(n)), length, je )
        call qdgeea( irest, re(ijk(n)), length, je )
        call qdgeea( irest, p (ijk(n)), length, je )
49      continue
      nlength = mg
      call qdgeia( irest, ncyc    , nlength  , je )
      call qdgeea( irest, dtheta  , nlength  , je )
      call qdgeea( irest, omegal  , nlength  , je )
      call qdclos( irest, je )
```

Each of the terms used in the FORTRAN code given above are defined below:

MG	number of grid blocks
IMX(L)	maximum i grid index for block L
JMX(L)	maximum j grid index for block L
KMX(L)	maximum k grid index for block L
R (IJK(L))	ρ at point IJK(L) in block L
RU(IJK(L))	ρu_x at point IJK(L) in block L
RV(IJK(L))	ρu_y at point IJK(L) in block L
RW(IJK(L))	ρu_z at point IJK(L) in block L
RE(IJK(L))	ρe at point IJK(L) in block L
P (IJK(L))	pressure at point IJK(L) in block L
QDOPEN	SDBLIB routine to open a file for input or output
QDGETI	SDBLIB routine to get an integer

QDGEIA	SDBLIB routine to get an integer array of length ILENGTH
QDGEEA	SDBLIB routine to get a real array of length ILENGTH
QDCLOS	SDBLIB routine to close a file
IREST	FORTRAN logical unit number for restart input
JE	An error trigger; 0 for no error, 1 if an error occurs
IB	Integer array containing the IMAX, JMAX, and KMAX grid block sizes
ILENGTH	Integer length of an array of data
IJK(L)	Integer pointer for block L to locate the initial memory location for a block of data

The restart data are written as a single-dimensioned array in the *SDBLIB* format, and the *ADPAC-AOACR* program includes a conversion routine (source file *convas.f*) which converts the three-dimensional array data to single dimension array data.

3.13 Convergence File Description

The *ADPAC-AOACR* convergence history file *case*.converge (see Section 3.5 for a description of the *ADPAC-AOACR* file naming convention) is an ASCII data file which contains the residual convergence history of the time-marching solution. The residual history is useful for determining whether the numerical solution has converged sufficiently to permit interrogation of the numerical results, or whether additional restarted calculations are required to obtain an accurate solution. Typically, a solution is deemed converged when the residuals have been reduced by three orders of magnitude or more. The data in the *case*.converge file are organized in the following format:

```
iteration        log10(maximum )      log10(root-mean square)
number               (residual)       (residual          )
    .                    .                    .
    .                    .                    .
    .                    .                    .
    .                    .                    .
```

The residual R at any cell in the finite volume solution is calculated as the sum of the changes in the 5 conservation variables ρ, ρu, ρv, ρw, and ρe. The maximum

residual is then defined as the maximum of all the residuals over all the cells of all mesh blocks. The root-mean square residual is the square root of the sum of the squares of all the cells for all mesh blocks. The *case*.**converge** file residual data are reported as the base 10 logarithm of the actual residuals in order to quickly evaluate the convergence of the solution (if the reported log10 maximum residual starts at -2.5 and ends up at -5.5, the solution has converged three orders of magnitude). The *case*.**converge** file is formatted in columns to permit convenient plotting using any of a number of x-y plotting programs (the *FULLPLOT* program described in Reference [3] is one example).

3.14   Image File Description

The *ADPAC-AOACR* graphics display system (see Chapter 9.0) has the capability of saving a raster image of the local graphics screen to a file at specific iteration intervals using the Silicon Graphics image file format. This feature is included as a simple means of constructing flowfield animations. The input variables dealing with this facility **FGRAFIX, FGRAFINT, FIMGSAV, FIMGINT** are described in Section 3.6, and the image file naming convention is described in Section 3.5. In short, image files can be saved when the graphics display system is running on a single Silicon Graphics workstation or across a network between two Silicon Graphics workstations supporting the IRIX Operating System Version 4.0.1 or above, and also supporting the IRIX *scrsave* command. Image files can be viewed after they have been saved by issuing the command

<div align="center">

ipaste *case.img.#*

</div>

Other IRIS-specific commands such as *imgview, movie* and others may also be suitable for viewing image files. Additional information on the IRIS image format and the image manipulation commands are available in the Silicon Graphics system documentation.

4. *ADPOST* POST PROCESSOR OPERATING INSTRUCTIONS

4.1 Introduction to *ADPOST*

In this chapter, a brief description of the *ADPOST* post-processing routine used to calculate a number of integrated quantities presented in the companion Final Report [1] is given. Early in the development of the *ADPAC-AOACR* code, it was determined that the most effective way to handle output data was to utilize a separate, efficient data processing routine, to free the *ADPAC-AOACR* code from the expense and complexity of calculating specific case-dependent output data. The wide variety of applications possible with the multiple-block code simply prohibited embedding a number of output data summaries in the aerodynamic analysis, and this prompted the development of the *ADPOST* program. *ADPOST* is designed to handle a number of post processing chores including mass-averaging and calculating radial distributions of flow data. Many of the options in *ADPOST* are slanted towards turbomachinery flows, but the program may be used with equal effectiveness on non-turbomachinery flows.

4.2 Configuring Maximum Array Dimensions

The maximum array dimensions of the *ADPOST* program are set in exactly the same manner as the *ADPAC-AOACR* program itself. All array size PARAMETER statements are in the source file **parameter.inc** included with the *ADPOST* program. A sample **parameter.inc** file would appear as:

```
parameter   ( nbmax  =       6    )
parameter   ( nra3d  = 350000     )
```

Here again, as in *ADPAC-AOACR*, NBMAX is the maximum number of mesh blocks permitted (6, in this case), and NRA3D is the maximum sum of all cell centered elements required for the mesh (a formula for estimating this value is given in Section 3.3). An error message is displayed whenever the user attempts to read in a mesh which is larger than that permitted by either NBMAX or NRA3D.

4.3 Compiling the *ADPOST* Program

In the directory containing the FORTRAN source of the *ADPOST* code, compilation is performed by executing the command:

make *option*

· The **make** command is standard on UNIX systems and automatically interrogates the file *Makefile* for instructions on how to perform the compilation. The *option* argument may be any of the variables listed below:

No argument - same as *link* below.

link This is the standard UNIX system compilation. This option will deliver a working executable on most UNIX systems which support standard naming conventions (*f77* as the standard compiler, etc.). The compilation includes basic compiler optimization (f77 -O).

cray This option is utilized when compiling the *ADPOST* code on a Cray computer.

aix This option is used when compiling the *ADPOST* code on an IBM RS-6000 workstation running the AIX operating system.

Once the code has been compiled, change directories to the location where the case of interest is stored. The *ADPOST* program is invoked by issuing the command:

*path/***adpost**

where *path* is the relative or absolute pathname of the directory containing the *ADPOST* executable file from the current local directory. For example, if the mesh and flow files are in the directory

/usr/people/me/testcase

and the *ADPOST* executable is in the directory

/usr/people/me/adpac-aoacr/src/adpost

then the commands

cd /usr/people/me/testcase
/usr/people/me/adpac-aoacr/src/adpost/adpost

would begin the *ADPOST* program process.

Once the *ADPOST* program is invoked, the following message appears:

```
******************************************************


    aaaa     dddd   ppppp   ooooo   sssss   ttttt
   a   a    d   d   p   p   o   o   s          t
   aaaaa    d   d   ppppp   o   o   ssssss     t
  a     a   d   d   p       o   o        s     t
  a     a   dddd    p       ooooo   sssss      t



******************************************************
```

Welcome to ADPOST, a post-processor for the
ADPAC code. All that is required to run this
routine is the original input file and the
corresponding mesh, and restart.old files.

You will first be prompted for the input
filename. It must contain the casename that
precedes the other files.

119

You will then be presented with several post
processing options from which you may choose.

Enter the Name of the Input File to be Post Processed.
 (be sure to enter both the filename and its suffix)

The input filename must now be entered and must contain the *case* name to
be post-processed. The original input file, used with the *ADPAC-AOACR* code is
recommended since it is in the expected format and contains the information necessary
for dimensionalizing flow variables. Next the restart.old and mesh data are read, after
which the following menu appears:

****

Choose one of the following options to continue:

MAIN OPTION MENU
----

1. Specify a section to be averaged.

2. Dimensionalize flow values.

3. Print relative quantities instead of absolute.

4. Print spanwise information also.

5. Maintain thermodynamic consistency.

6. Plot flow variables.

7. Exiting Adpost.

****

Enter option number now.

Currently, only six options are available since the plot routines for option #6 have not yet been incorporated. A brief description of each of the other six options follows:

1. Specify a section to be averaged.

Option #1 allows the user to specify the portion of the flow field for which the mass averaged quantities are to be calculated. This value can only be specified for constant i, j or k planes but need not include the entire flow field on these planes. If no other options are picked in conjunction with #1 then, by default, the mass averaged values calculated for the specified section are in nondimensional form; the velocities and stagnation properties are absolute values, and thermodynamic consistency is maintained in each grid cell but not necessarily in the resulting mass averaged quantities. Other formations are possible by choosing additional options in combination with #1. Only one section of the flow field can be mass averaged at a time, so if values are needed in separate sections/blocks this option must be chosen repeatedly. Mass averaged quantities calculated with *ADPOST* are; stagnation temperature and pressure, static temperature and pressure, and velocity. A sample output is shown below:

FLOW QUANTITIES FOR BLOCK #1

121

MASS AVERAGING OVER i = CONSTANT SLICES

GRID SLICE	AREA	MASSFLOW RATE	TOTAL PRESSURE	TOTAL TEMPERATURE	STATIC PRESSURE	STATIC TEMPERATURE
1	0.1282E-01	0.2135E-02	0.1033E+01	0.1000E+01	0.9949E+00	0.9894E+00
2	0.1282E-01	0.2135E-02	0.1033E+01	0.1000E+01	0.9949E+00	0.9894E+00
3	0.1282E-01	0.2135E-02	0.1033E+01	0.1000E+01	0.9949E+00	0.9894E+00
4	0.1282E-01	0.2135E-02	0.1033E+01	0.1000E+01	0.9949E+00	0.9894E+00
etc.						

If no section if specified with option #1, then calculations for all other options, except #7, are performed on the entire flow field.

2. Dimensionalize flow values.

Option #2 dimensionalizes mass averaged flow values with the reference pressure, temperature, and diameter found in the input file. For a description of how variables are dimensionalized, see Section 1.2 of the Final Report [1].

3. Print relative quantities instead of absolute.

Option #3 calculates the mass averaged velocity, total pressure and total temperature relative to a rotating coordinate system (e.g., rotor).

4. Print spanwise information also.

Option #4 provides additional mass averaged values for each radial (spanwise, j index) location in the specified planes. Unless radial planes are specified, then values at each axial location are given instead. The spanwise/axial averages are printed first, followed by the values for the entire plane. A sample output is shown below:

FLOW QUANTITIES FOR BLOCK #1

MASS AVERAGING OVER i = CONSTANT SLICES

spanwise integration for grid surface i =   1

j slice #	percent span	mass flow	ptotal	ttotal	swirl angle
2	0.000 - 0.001	0.265201E-06	1.00322	1.00140	-71.9231
3	0.001 - 0.001	0.733345E-06	1.00299	1.00024	-60.4713
4	0.001 - 0.003	0.112133E-05	1.00344	0.999918	-57.8162
.					
.					
.					
.					

Mass Average Totals for i= 1 Plane

GRID SLICE	AREA	MASSFLOW RATE	TOTAL PRESSURE	TOTAL TEMPERATURE	STATIC PRESSURE	STATIC TEMPERATURE
1	0.1282E-01	0.2135E-02	0.1033E+01	0.1000E+01	0.9949E+00	0.9894E+00

5. Maintain thermodynamic consistency.

Option #5 maintains thermodynamic consistency in the averaged flow quantities. This is accomplished by mass averaging only the cell centered conservation variables. The resulting mass averaged velocity, density and energy are then used with perfect gas relationships to calculate pressure and temperature.

6. Plot flow variables.

This option is not currently available.

7. Exiting Adpost.

After each option has been chosen, the user is asked if they would like to specify any additional options. If the response is "yes", then you are returned to the main menu. If the response is "no" then calculations are performed on the specified section with the options chosen, after which you are again returned to the main menu. The user may then either specify another section or other options to be performed.

The entire main menu sequence is repeated until option #7 is picked and the process is terminated. Calculated quantities are stored in a file called *adpost.out.*

4.4   Running the *ADPOST Program*

4.5   Sample Session Using the *ADPOST* Program

A sample interactive session from the *ADPOST* program is printed below for the case of a ducted propfan H-grid with multiple blade rows (Standard Configuration #7). The mesh configuration and mesh index reference are described in the Appendix for the sample test case. All the user responses are listed in boldface type during the interactive session listing.

Sample Interactive *ADPOST* Session Listing

```
*************************************************

    aaaa      dddd   ppppp   ooooo   sssss   ttttt
   a    a    d   d   p   p   o   o   s          t
   aaaaa     d   d   ppppp   o   o   ssssss     t
   a    a    d   d   p       o   o        s      t
   a    a    dddd    p       ooooo   ssssss      t
```

```
*************************************************
```

Welcome to ADPOST, a post-processor for the
ADPAC code. All that is required to run this
routine is the original input file and the
corresponding mesh, restart.old, p3dabs, and
p3drel files.

You will first be prompted for the input
filename. It must contain the casename that
precedes the other files.

Next, mass-average flow values are calculated
at each axial station.

Finally, the user is presented with several
other post processing options from which they
may choose.

Enter the Name of the Input File to be Post Processed.
 (be sure to enter both the filename and its suffix)

    nasa.input

Mass averaged values are now being
calculated for every 'i' plane.


```
*************************************************
```

Calculations have been written to 'adpost.out'.

_____

Choose one of the following options to continue:

MAIN OPTION MENU
\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-\-

1. Specify another section to be averaged.

2. Dimensionalize flow values.

3. Print relative quantities instead of absolute.

4. Print spanwise information also.

5. Maintain thermodynamic consistency.

6. Plot flow variables.

7. Exit Adpost.

Enter option number now.

   1

****

   Option #1 allows you to specify the portion of
   flow field to be averaged. This can be done on
   any i, j or k plane and need not include the
   the entire slice.

Enter the block number to be mass-averaged

126

or if you wish to return to the Main Menu,
enter a number greater than 4

    1

In what direction are slices to be averaged?
 (Enter i, j, or k)

    i

Specify the section to be averaged

Enter the minimum i,j,k, indices of the first surface.

    129 1 1

Enter the maximum i,j,k, indices of the last surface.
Maximum grid indices: 129, 17, 17

    129 17 17

Enter the increment in the i direction.

    1

 Would you like to specify any additional
 options? (yes/no)

yes

Choose one of the following options to continue:

                MAIN OPTION MENU
                ----

    1. Specify another section to be averaged.

    2. Dimensionalize flow values.

    3. Print relative quantities instead of absolute.

    4. Print spanwise information also.

    5. Maintain thermodynamic consistency.

    6. Plot flow variables.

    7. Exit Adpost.

Enter option number now.

        5

****

Thermodynamic consistency will be preserved.

****

Would you like to specify any additional
options? (yes/no)

yes

Choose one of the following options to continue:

MAIN OPTION MENU
----

1. Specify another section to be averaged.

2. Dimensionalize flow values.

3. Print relative quantities instead of absolute.

4. Print spanwise information also.

5. Maintain thermodynamic consistency.

6. Plot flow variables.

7. Exit Adpost.

Enter option number now.

4

****

Spanwise print flag has been turned on.

****

Would you like to specify any additional
options? (yes/no)

no

Calculations for selected options are now being processed.

Calculations are complete and can be found in 'adpost.out'.

Choose one of the following options to continue:

         MAIN OPTION MENU
         ----

1. Specify another section to be averaged.

2. Dimensionalize flow values.

3. Print relative quantities instead of absolute.

4. Print spanwise information also.

5. Maintain thermodynamic consistency.

6. Plot flow variables.

7. Exit Adpost.

Enter option number now.

   7

4.6 Sample Output File from the *ADPOST* Program

The output file **adpost.out** from the interactive session given in Section 4.4 is listed below. Note that the output is similar to the *ADPAC-AOACR* program initially, as *ADPOST* reads in the input, mesh and restart files. At the end of the file, the various summations from the mass-averages requested in the interactive session are displayed.

```
*****************************************************


     aaaa      dddd   ppppp   ooooo   sssss   ttttt
    a    a    d    d   p    p  o     o  s          t
    aaaaa    d    d   ppppp   o     o  ssssss     t
   a    a    d    d   p       o   o       s       t
   a    a    dddd     p       ooooo   sssss       t



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




   case name defined as: nasa
   *********************



   *********************************************************
             begin grid input
   *********************************************************


   mesh file name: nasa.mesh
   ***************

   number of grids: (nblks) =              4
```

```
****************

grid block dimensions :
*********************

    block      imx      jmx      kmx
    -----      ---      ---      ---
        1      129       17       17
        2      129       17       17
        3       97       17       17
        4       97       17       17


initialize storage array and pointer locations
******************************************************

    block    3-d ijk     length    total length
        1          1      42120          42120
        2      42121      42120          84240
        3      84241      31752         115992
        4     115993      31752         147744


program array dimensions:
*****************************

nra3d (3-d array size) =           350000



begin reading grid block coordinates
*************************************************

grid #              1 size          130          18          18
 grid block              1 coordinates successfully read
 conatan routine used for cylindrical conversion
grid #              2 size          130          18          18
 grid block              2 coordinates successfully read
 conatan routine used for cylindrical conversion
grid #              3 size           98          18          18
 grid block              3 coordinates successfully read
 conatan routine used for cylindrical conversion
```

132

```
grid #           4 size           98          18          18
 grid block              4 coordinates successfully read
 conatan routine used for cylindrical conversion


 **************************************************
               begin flow input
 **************************************************



   input file read on unit   15
   **********************



   input file data
   ***************
   rm     ( reference mach number        ) =      0.7500
   gamma  ( specific heat ratio          ) =      1.4000
   cfl    ( cfl number (+ time accurate  ) =      5.0000
   vt     ( =1, time acc., =0, local time ) =     0.0000
   vis2   ( 2nd order dissipation factor ) =      0.5000
   vis4   ( 4th order dissipation factor ) =      0.0156
   fncmax ( maximum iterations           ) =     10.0000
   rest   ( 1 = restart solution         ) =      0.0000
   save   ( 1 = save solution            ) =      0.0000
   alpha  ( angle of attack              ) =      0.0000
   fistep ( recalculate time step        ) =      1.0000
   fnprnt ( 1 = print flowfield          ) =      1.0000
   fiprnt ( 1 = print partial flowfield  ) =      1.0000
   finvvi ( 0 = inviscid flow solution   ) =      0.0000
          ( 1 =  viscous flow solution   ) =
   epsx   ( i implicit smoothing factor  ) =      1.5000
   epsy   ( j implicit smoothing factor  ) =      1.5000
   epsz   ( k implicit smoothing factor  ) =      1.5000
   iunint ( # it. for unsteady plot3d file ) =   9999.0000
   itimei ( #it between time step update ) =      1.0000
   iturbi ( #it between turb. model update ) =    1.0000
   iturbb ( #it before starting turb model ) =  9999.0000
   diam   ( blade diameter (ft)          ) =      9.0000
```

```
tref    ( total temperature (deg. r)       ) =     518.7000
pref    ( total pressure    (lbf/ft**2)  ) =    2116.8000
rgas    ( gas constant (ft-lbf/lbm-deg r)) =    1716.2600
pr      ( prandtl number                  ) =       0.7000
prt     ( turbulent prandtl number        ) =       0.9000
```

non-dimensional reference values calculated as:
****
```
rho0    ( reference density               ) =       0.7660
u0      ( reference axial velocity        ) =       0.8413
v0      ( reference radial velocity       ) =       0.0000
w0      ( reference theta velocity        ) =       0.0000
ei0     ( reference internal energy       ) =       2.6011
h0      ( reference enthalpy              ) =       3.5000
p0      ( reference pressure              ) =       0.6886
t0      ( reference temperature           ) =       0.8989
dmu0    ( reference viscosity             ) =       0.9195
omega   ( rotational speed                ) =      -1.9016
```

non-dimensional value for row   2 calculated as:
****
```
omega   ( rotational speed                ) =      -1.9016
```

non-dimensional value for row   3 calculated as:
****
```
omega   ( rotational speed                ) =       0.0000
```

non-dimensional value for row   4 calculated as:
****
```
omega   ( rotational speed                ) =       0.0000
```

dimensional reference values calculated as:
****
```
rho0    ( density             (lbm/ft**3) ) = 0.182151E-02
```

```
        u0      ( axial velocity    (ft/s)       ) =   793.825
        v0      ( radial velocity   (ft/s)       ) = 0.000000E+00
        w0      ( theta velocity    (ft/s)       ) = 0.000000E+00
        ei0     ( total int. energy (ft-lbf/lbm) ) = 0.231558E+07
        h0      ( total enthalpy    (ft-lbf/lbm) ) = 0.311578E+07
        p0      ( total pressure    (lbf/ft**2)  ) =   1457.57
        t0      ( total temperature (deg. r)     ) =   466.247
        dmu0    ( viscosity         (lbf-s/ft**2)) = 0.343739E-06
        omega   ( rotational speed  (rad/s)      ) = -199.351
        rpm     ( rev/min           (rev/min)    ) = -1903.66


additional blade row   2 values calculated as:
*********************************************
        omega   ( rotational speed  (rad/s)      ) = -199.351
        rpm     ( rev/min           (rev/min)    ) = -1903.66


additional blade row   3 values calculated as:
*********************************************
        omega   ( rotational speed  (rad/s)      ) = 0.000000E+00
        rpm     ( rev/min           (rev/min)    ) = 0.000000E+00


additional blade row   4 values calculated as:
*********************************************
        omega   ( rotational speed  (rad/s)      ) = 0.000000E+00
        rpm     ( rev/min           (rev/min)    ) = 0.000000E+00


nondimensionalization values calculated as:
*********************************************
        rho     ( density           (lbm/ft**3)  ) = 0.237783E-02
        u       ( axial velocity    (ft/s)       ) =   943.517
        ei      ( total int. energy (ft-lbf/lbm) ) =   890224.
        p       ( total pressure    (lbf/ft**2)  ) =   2116.80
        t       ( total temperature (deg. r)     ) =   518.700
        dmu     ( viscosity         (lbf-s/ft**2)) = 0.373852E-06
        omega   ( rotational speed  (rad/s)      ) =   104.835
```

135

length (length   (ft)   ) = 9.00000

input data successfully entered
****

set up block data:
****

cell area terms calculated for grid block   #   1
cell area terms calculated for grid block   #   2
cell area terms calculated for grid block   #   3
cell area terms calculated for grid block   #   4

cell volume terms calculated for grid block #   1
cell volume terms calculated for grid block #   2
cell volume terms calculated for grid block #   3
cell volume terms calculated for grid block #   4

****
            begin restart input
****

restart file name: nasa.restart.old
****

number of blocks (nblksf):   4
****

restart file block sizes
****

block	imx	jmx	kmx
1	130	18	18
2	130	18	18
3	98	18	18

```
            4            98            18            18
```

restart data for block #   1 entered
restart data for block #   2 entered
restart data for block #   3 entered
restart data for block #   4 entered

FLOW QUANTITIES FOR BLOCK #1

MASS AVERAGING OVER i = CONSTANT SLICES

GRID SLICE	AREA	MASSFLOW RATE	TOTAL PRESSURE	TOTAL TEMPERATURE	STATIC PRESSURE	STATIC TEMPERATURE

spanwise integration for grid surface i = 129

j slice #	percent span	mass flow	ptotal	ttotal	swirl angle
2	0.000 - 0.022	0.388604E-03	1.12050	1.04476	-27.5691
3	0.022 - 0.052	0.559818E-03	1.12535	1.04529	-27.8464
4	0.052 - 0.094	0.799983E-03	1.12826	1.04510	-27.8023
5	0.094 - 0.148	0.113206E-02	1.13259	1.04478	-27.1676
6	0.148 - 0.217	0.156474E-02	1.13696	1.04483	-26.2078
7	0.217 - 0.301	0.208925E-02	1.14238	1.04527	-25.0896
8	0.301 - 0.397	0.264947E-02	1.14800	1.04656	-24.0559
9	0.397 - 0.500	0.315033E-02	1.15495	1.04885	-23.3043
10	0.500 - 0.603	0.347579E-02	1.16192	1.05161	-22.6446
11	0.603 - 0.699	0.352727E-02	1.16726	1.05458	-22.2051
12	0.699 - 0.783	0.329510E-02	1.16925	1.05701	-21.8015
13	0.783 - 0.852	0.285540E-02	1.16869	1.05906	-21.4849
14	0.852 - 0.906	0.231871E-02	1.16527	1.06065	-21.3212
15	0.906 - 0.948	0.179275E-02	1.16143	1.06218	-21.1105
16	0.948 - 0.978	0.133604E-02	1.15755	1.06373	-20.9239
17	0.978 - 1.000	0.972663E-03	1.15467	1.06496	-20.6821

129   0.5499E-01 0.3191E-01 0.1156E+01 0.1053E+01 0.9246E+00 0.9882E+00
 processing completed
 execution normally terminated

5. STANDARD MESH BLOCK CONFIGURATIONS

In this section, a description of several standard mesh block configurations for ducted/unducted fans and other turbomachinery-related geometries are given. The standard configuration list defines a series of pretested mesh block configurations for various turbomachinery-related calculations of interest to gas turbine engine designers. The standard configurations, in conjunction with the *SETUP* and *ROTGRID* programs described in Chapters 6 and 8, respectively, attempt to remove some of the burdens of data preparation involved in an *ADPAC-AOACR* solution for inexperienced users. This particular configuration list does not imply any limitations on the *ADPAC-AOACR* code itself. In fact, the user is encouraged to try alternate mesh configurations within the limitations of the *ADPAC-AOACR* multiple block boundary condition application strategy.

5.1 Description of Standard Configurations

Each of the predefined standard configurations are defined below. A corresponding graphic describing the individual configurations are given in Figures 5.1-5.10. In the description of a given configuration, the mesh size is assumed to be represented by the values $imax, jmax, kmax$ as limits in the i, j, and k coordinate directions, respectively.

Standard Configuration #1

Title:	Single-passage turbomachinery H-grid
Mesh Type:	Single Block H-grid
Number of Mesh Blocks:	1
Flow Type:	Steady

139

Geometry: Compressor rotor, stator, unducted fan, propfan
Number of Blade Rows: 1
Mesh Generation Program: *TIGG3D, CHGRIDV2, MULAC*, and others

Standard Configuration #1 consists of a single H-type mesh block discretizing a single blade passage of a turbomachinery blade row or propfan, as shown in Figure 5.1. The mesh is spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax). Blade hub/tip clearances may be represented by removing the blade displacement in the clearance region above or below the blade. Inner boundaries on this mesh include the blade leading and trailing edge i indices, and the blade base and tip j indices. The passage endwalls ($j=1$, $j=$jmax) are represented as solid surfaces (for a freestream outer boundary, see Standard Configuration #2). This is a relatively simple mesh to generate, and a number of grid generation schemes are available to construct this type of mesh. The advantage of this mesh system lies in it's simplicity and consistency with the manner in which turbomachinery airfoils are defined and analyzed experimentally (streamline-like radial planes, constant axial planes). The disadvantages of this mesh system are that the airfoil leading and trailing edges are poorly defined because of the sheared mesh system, and grid points are often not used economically in the far field because of the requirement for mesh clustering along the periodic boundaries in the vicinity of the blade (although it is possible to construct meshes with 2-D mesh blocks in the far field to reduce this inefficiency).

Standard Configuration #2

Title: Single-passage ducted propfan H-grid
Mesh Type: Two block H-grid
Number of Mesh Blocks: 2
Flow Type: Steady
Geometry: Ducted fan
Number of Blade Rows: 1
Mesh Generation Program: *TIGG3D* used with *ROTGRID*, and others

ADPAC-AOACR Standard Configuration #1
Single Passage Turbomachinery H-Grid

Configuration Information

Mesh Type   : Single Passage H-Grid
Number of Blocks : 1
Flow Condition   : Steady
Applications   : Turbomachinery/Propfan
                    Steady Flow Aerodynamics

Turbomachinery Cascade

Computational Domain

Solid wall or freestream
outer boundary

i–j view

Can specify blade
tip – tip clearance

j–k view

Airfoils

Can specify blade
base – hub clearance

Airfoil

j

i

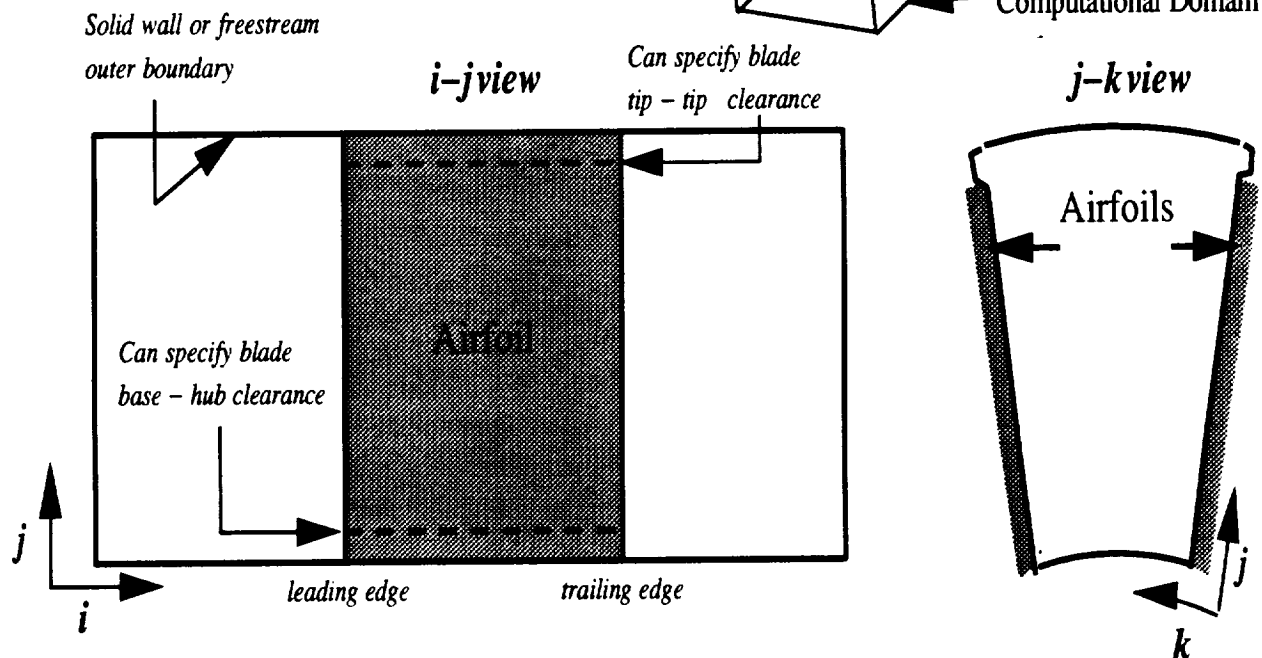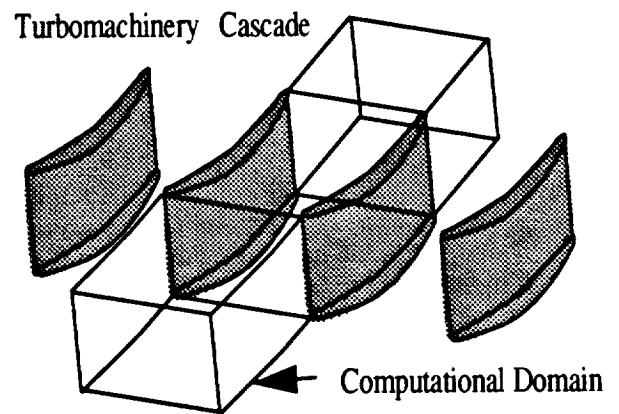*leading edge*   *trailing edge*

j

k

Figure 5.1:   Standard Configuration #1 Geometry and Multiple Block Mesh Structure

Standard Configuration #2 consists of a pair of H-type mesh blocks discretizing a single blade passage of a ducted fan blade row, as shown in Figure 5.2. Both mesh blocks are spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces (k=1 and k=kmax) of mesh block #1. A hub spinner boundary may be imposed as an embedded feature in mesh block #1. A blade tip clearance may be imposed by removing the blade displacement to satisfy spatial periodicity. The two mesh blocks are divided by the duct and a pair of mating surfaces which extend upstream and downstream from the duct leading and trailing edges. The duct itself is therefore represented as a gap between the two mesh blocks. The mesh points defining the mating surfaces are common to both mesh blocks. Inner boundaries on this mesh include the blade leading and trailing edge i indices, the duct leading and trailing edge i indices, the hub spinner leading and trailing edge i indices, and the blade tip j index. This mesh is best generated using the *TIGG3D* grid generation program in conjunction with the *ROTGRID* program to convert the single block *TIGG3D* mesh into the two block system described above, and to remove any duplicate mesh lines normally added by the *TIGG3D* program for ducted geometries. The advantage of this mesh system lies in it's simplicity and consistency with the manner in which ducted fan airfoils are defined and analyzed experimentally (streamline-like radial planes, constant axial planes). The disadvantages of this mesh system are that the airfoil and duct leading and trailing edges are poorly defined because of the sheared mesh system, and grid points are often not used economically in the far field because of the requirement for mesh clustering along the periodic boundaries in the vicinity of the blade and along radial planes in the vicinity of the duct (although it is possible to construct meshes with 2-D mesh blocks in the far field to reduce this inefficiency).

Standard Configuration #3

Title:	Single-passage ducted propfan C-grid
Mesh Type:	Five block combination C-H grid
Number of Mesh Blocks:	5
Flow Type:	Steady

ADPAC-AOACR Standard Configuration #2
Single Passage Ducted Fan H-Grid

Configuration Information

Mesh Type : Single Passage H-Grid
Number of Blocks : 2
Flow Condition : Steady
Applications : Ducted Fan Steady
Flow Aerodynamics

Ducted
Fan

Computational
Domain

j-k view

i-j view

Block #2   Free-flow Far Field Boundary

Duct

Block #1

Airfoil

Duct

Airfoils

j

i

spinner
trailing edge

blade
leading edge

blade
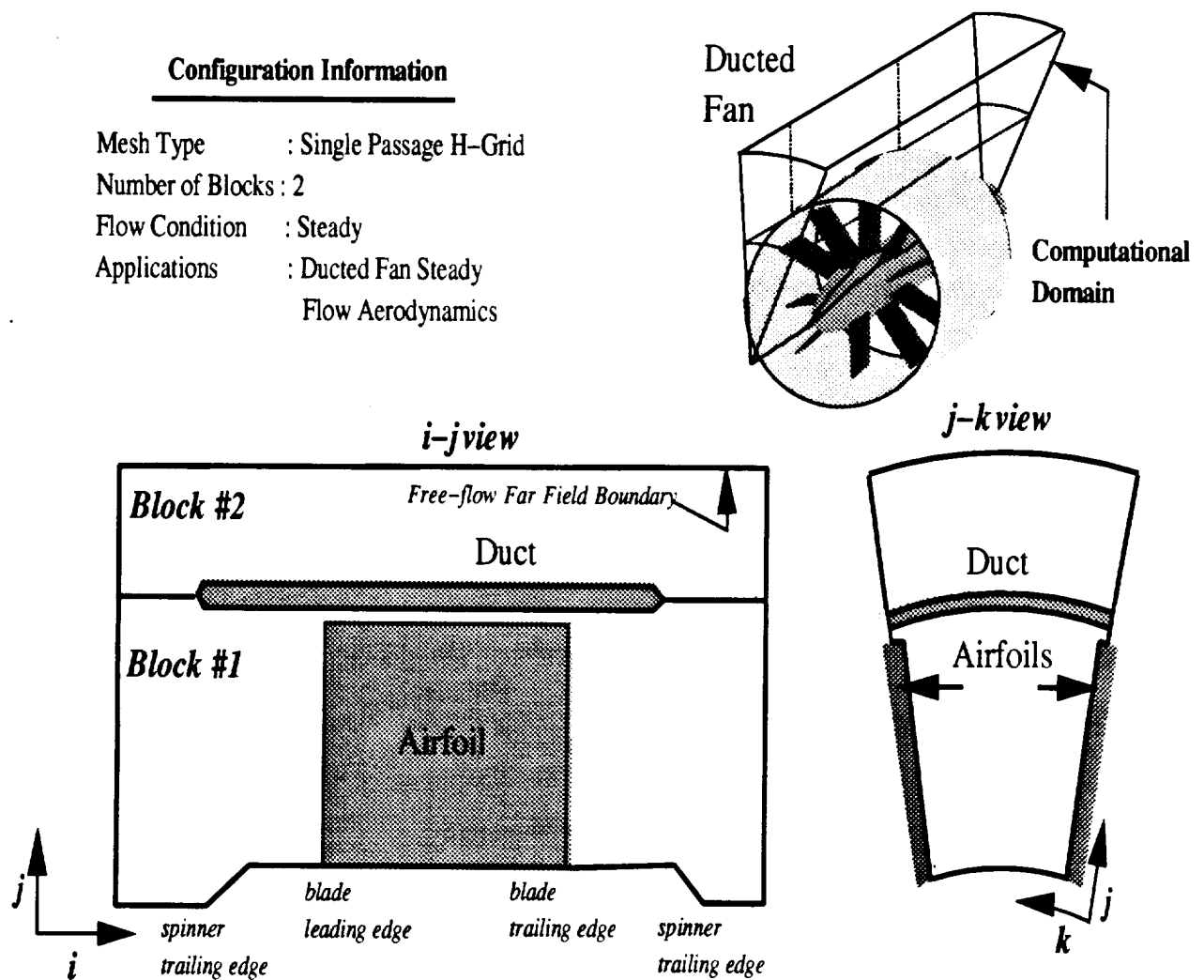trailing edge

spinner
trailing edge

j

k

Figure 5.2:   Standard Configuration #2 Geometry and Multiple Block Mesh Structure

Geometry: Ducted fan

Number of Blade Rows: 1

Mesh Generation Program: *CHGRIDV2* and others

Standard Configuration #3 consists of a five block mesh system employing combined C and H type meshes, as shown in Figure 5.3, to discretize a single blade passage of a ducted fan blade row. All mesh blocks are spatially periodic, except in the vicinity of the blades themselves in block #1. The blades are represented as a circumferential displacement on the periodic surfaces (k=1 and k=kmax) of mesh block #1. A hub spinner boundary may be imposed as an embedded feature in mesh block #1. A blade tip clearance may be imposed by removing the blade displacement to satisfy spatial periodicity. The mesh blocks share common points along all of the mating surfaces between neighboring mesh blocks. Inner boundaries on this mesh include the blade leading and trailing edge i indices, the C-grid/H-grid mating i indices, the duct trailing edge i indices, and the hub spinner leading and trailing edge i indices. This mesh is best generated using the *CHGRIDV2* grid generation program. The advantage of this mesh system lies in the detailed representation of the duct leading edge permitted by the embedded C-grid, and the ability to efficiently cluster points about the duct surface and wake. The disadvantages of this mesh system are that the airfoil leading and trailing edges are poorly defined because of the sheared mesh system, and grid points are often not used economically in the far field because of the requirement for mesh clustering along the periodic boundaries in the vicinity of the blade and along radial planes in the vicinity of the duct. Another disadvantage of this mesh system is the complexity of the boundary specifications required to couple the mesh blocks, and the possibility of poor convergence as a result of the many inner domain block boundaries.

Standard Configuration #4

Title: Full rotor unducted propfan H-grid

Mesh Type: Blade passage H-grid

Number of Mesh Blocks: 1*N (N=Number of Blades)

Flow Type: Unsteady

144

Single Passage Ducted Fan C-H Grid
ADPAC-AOACR Standard Configuration #3

Configuration Information

Mesh Type   : Single Passage 5–Block C–H Grid System
Number of Blocks : 5
Flow Condition   : Steady
Applications   : Ducted Fan Steady
                     Flow Aerodynamics

Ducted
Fan

Computational
/ Domain

j–k view

i–j view

Block #5

Free-flow Far Field Boundary

Block #3 (C-Mesh)

C–gridduct
trailing edge

Block #2

Airfoil

Block #4

Block #1

j

i

spinner
trailing edge

blade
leading edge

blade
trailing edge

spinner
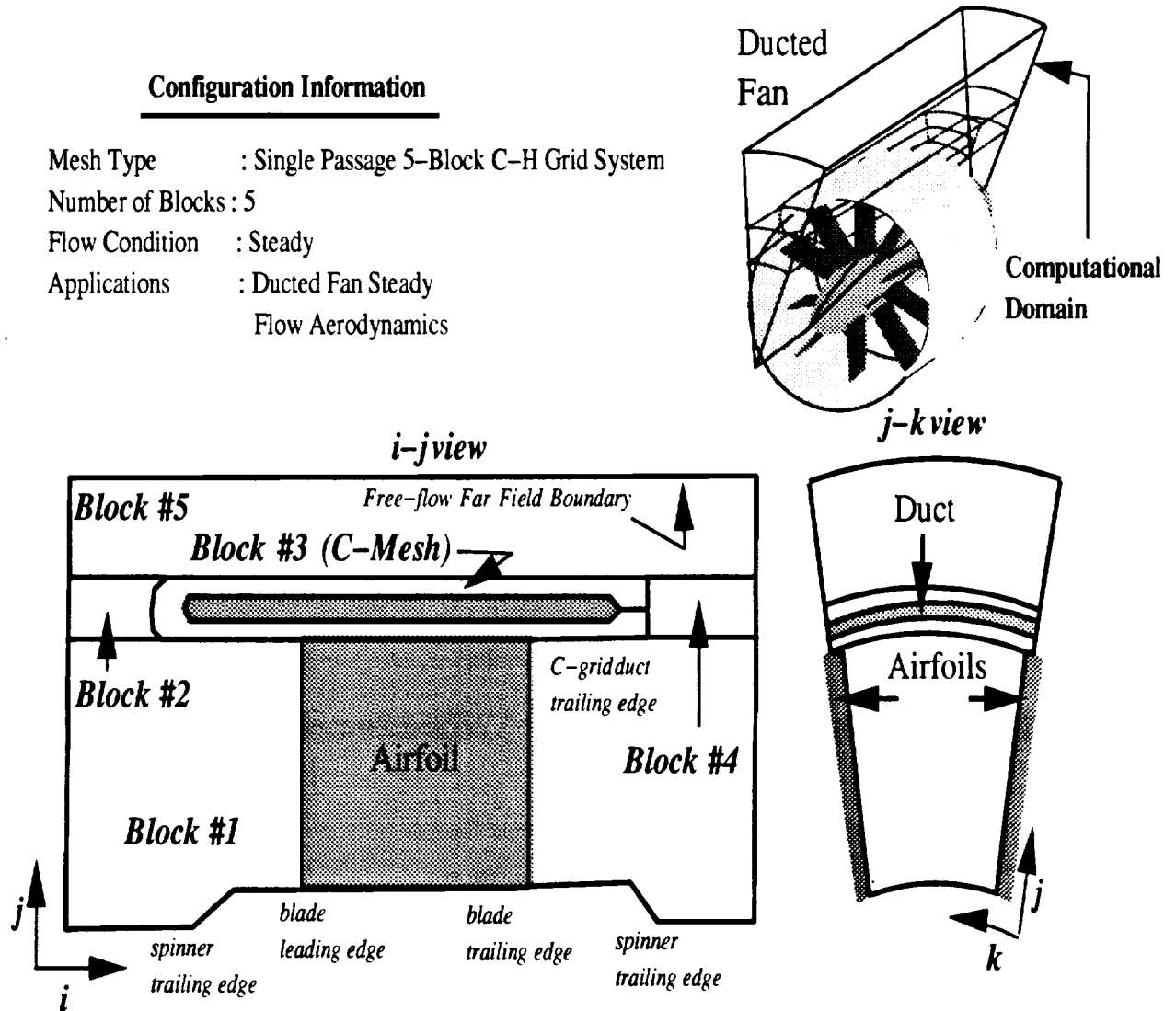*trailing edge*

Duct

Airfoils

j

k

Figure 5.3:   Standard Configuration #3 Geometry and Multiple Block Mesh Structure

145

Geometry: Propeller, profan, unducted fan, compressor

Number of Blade Rows: 1

Mesh Generation Program: *TIGG3D, CHGRIDV2, MULAC* then use *ROTGRID*, or others

Standard Configuration #4 consists of multiple grid blocks discretizing a complete blade row of an unducted fan or internal turbomachinery blade row using a single H-type mesh block per blade passage. This mesh system is illustrated in Figure 5.4. Each blade passage mesh block is similar in form to the mesh systems in Standard Configuration #1, above. The individual blade passage mesh blocks are spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces (k=1 and k=kmax) of each mesh block. Blade hub/tip clearances may be represented by removing the blade displacement in the clearance region above or below the blade. Inner boundaries on this mesh include the blade leading and trailing edge i indices, and the blade base and tip j indices. The passage endwalls (j=1, j=jmax) are represented as either solid surfaces or a freestream outer boundary. This is a relatively simple mesh to generate, and a number of grid generation schemes are available to construct this type of mesh for a single blade passage. The *ROTGRID* program can be used to create the duplicate blade passage mesh blocks for the full rotor geometry. Periodic surfaces between adjacent blade passages share common mesh points. The advantage of this mesh system lies in it's simplicity and consistency with the manner in which turbomachinery airfoils are defined and analyzed experimentally (streamline-like radial planes, constant axial planes). The disadvantages of this mesh system are that the airfoil leading and trailing edges are poorly defined because of the sheared mesh system, and grid points are often not used economically in the far field because of the requirement for mesh clustering along the periodic boundaries in the vicinity of the blade (although it is possible to construct meshes with 2-D mesh blocks in the far field to reduce this inefficiency).

Standard Configuration #5

Title: Full rotor ducted propfan H-grid

146

ADPAC-AOACR Standard Configuration #4
Full Rotor Unducted Fan H-Grid

Configuration Information

Mesh Type   : Full Rotor H-Mesh System
Number of Blocks : 1*Number of Blades
Flow Condition   : Unsteady
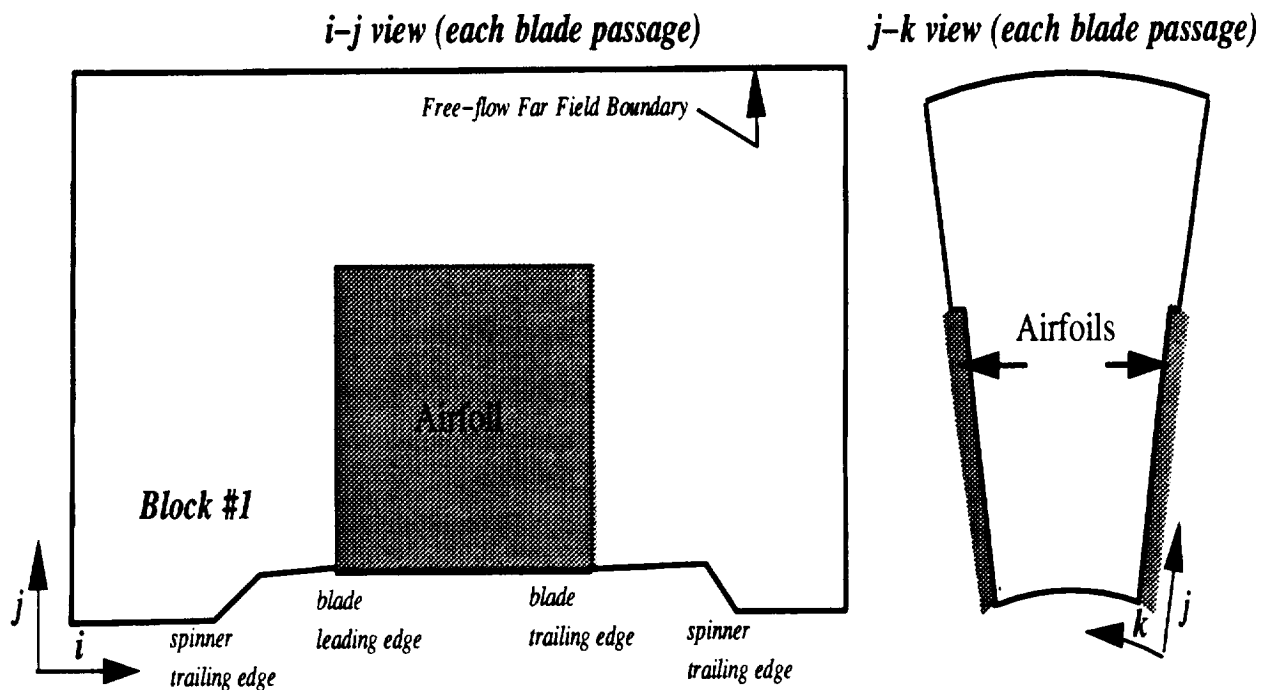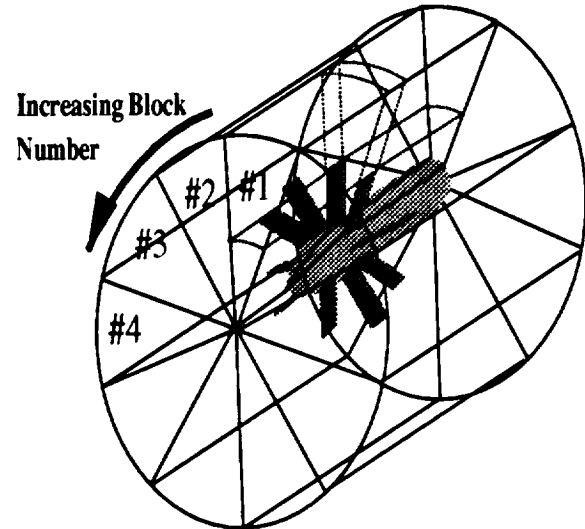Applications   : Unducted Fan at Angle of Attack

i-j view (each blade passage)

j-k view (each blade passage)

Figure 5.4: Standard Configuration #4 Geometry and Multiple Block Mesh Structure

Mesh Type:	2-Block H-grid per blade passage
Number of Mesh Blocks:	2*N (N=Number of Blades)
Flow Type:	Unsteady
Geometry:	Ducted Fan
Number of Blade Rows:	1
Mesh Generation Program:	*TIGG3D, CHGRIDV2, MULAC* used with *ROTGRID*, and others

Standard Configuration #5 consists of multiple grid blocks discretizing a complete blade row of a ducted fan using a pair of H-type mesh blocks per blade passage. Each blade passage consists of two mesh blocks in the same manner as the mesh system in Standard Configuration #2, above. The mesh block structure and numbering scheme is illustrated in Figure 5.5. The mesh blocks for a given blade passage are spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax) of the inner mesh block. A hub spinner boundary may be imposed as an embedded feature in the inner mesh block. A blade tip clearance may be imposed by removing the blade displacement to satisfy spatial periodicity. The two mesh blocks per blade passage are divided by the duct, and mating surfaces which extend upstream and downstream from the duct leading and trailing edges. The duct itself is therefore represented as a gap between the two mesh blocks in each passage. The mesh systems share common points along all of the mating surfaces between neighboring mesh blocks. Inner boundaries on this mesh include the blade leading and trailing edge i indices, the duct leading and trailing edge i indices, the hub spinner leading and trailing edge i indices, and the blade tip j index. A single blade passage mesh is best generated using the *TIGG3D* grid generation program in conjunction with the *ROTGRID* program to convert the single block *TIGG3D* mesh into the two block system per blade passage, and ultimately a full rotor mesh system. The advantage of this mesh system lies in it's simplicity and consistency with the manner in which ducted fan airfoils are defined and analyzed experimentally (streamline-like radial planes, constant axial planes). The disadvantages of this mesh system are that the airfoil and duct leading and trailing edges are poorly defined because of the sheared mesh system, and grid points are often not used economically in the far field because

148

of the requirement for mesh clustering along the periodic boundaries in the vicinity of the blade (although it is possible to construct meshes with 2-D mesh blocks in the far field to reduce this inefficiency).

Standard Configuration #6

Title:	Full Rotor Ducted Fan C-H Grid System
Mesh Type:	Five Block C-H Combination
Number of Mesh Blocks:	5*N (N=Number of Blades)
Flow Type:	Unsteady
Geometry:	Ducted Fan
Number of Blade Rows:	1
Mesh Generation Program:	*CHGRIDV2* used with *ROTGRID* and others

Standard Configuration #6 consists of multiple grid blocks discretizing a complete blade row of a ducted fan using a combination C-H mesh system per blade passage. Each blade passage consists of five mesh blocks in the same manner as the mesh system in Standard Configuration #3, above. The mesh block structure and numbering scheme is illustrated in Figure 5.6. The mesh blocks for a given blade passage are spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax) of the inner mesh block. A hub spinner boundary may be imposed as an embedded feature in the inner mesh block. The mesh systems share common points along all of the mating surfaces between neighboring mesh blocks. A single blade passage mesh system may be generated using the *CHGRIDV2* grid generation program. The full rotor mesh can then be constructed by using the *ROTGRID* program. The advantage of this mesh system lies in the detailed representation of the duct leading edge permitted by the embedded C-grid, and the ability to efficiently cluster points about the duct surface and wake. The disadvantages of this mesh system are that the airfoil leading and trailing edges are poorly defined because of the sheared mesh system, and grid points are often not used economically in the far field because of the requirement for mesh clustering along the periodic boundaries in the vicinity of the blade and along radial planes in the vicinity of the duct. (although it is possible

ADPAC-AOACR Standard Configuration #5
Full Rotor Ducted Fan H-Grid

Configuration Information

Mesh Type	: Full Rotor H–Mesh System
Number of Blocks	: 2*Number of Blades
Flow Condition	: Unsteady
Applications	: Ducted Fan at Angle of Attack

Increasing Block Number

#4 #2
#6 #3
#5

i–j view (each blade passage)   *j–k view (each blade passage)*

Block #2

Free-flow Far Field Boundary

Duct

Airfoil

Block #1

j
i

spinner
trailing edge

blade
leading edge

blade
trailing edge

spinner
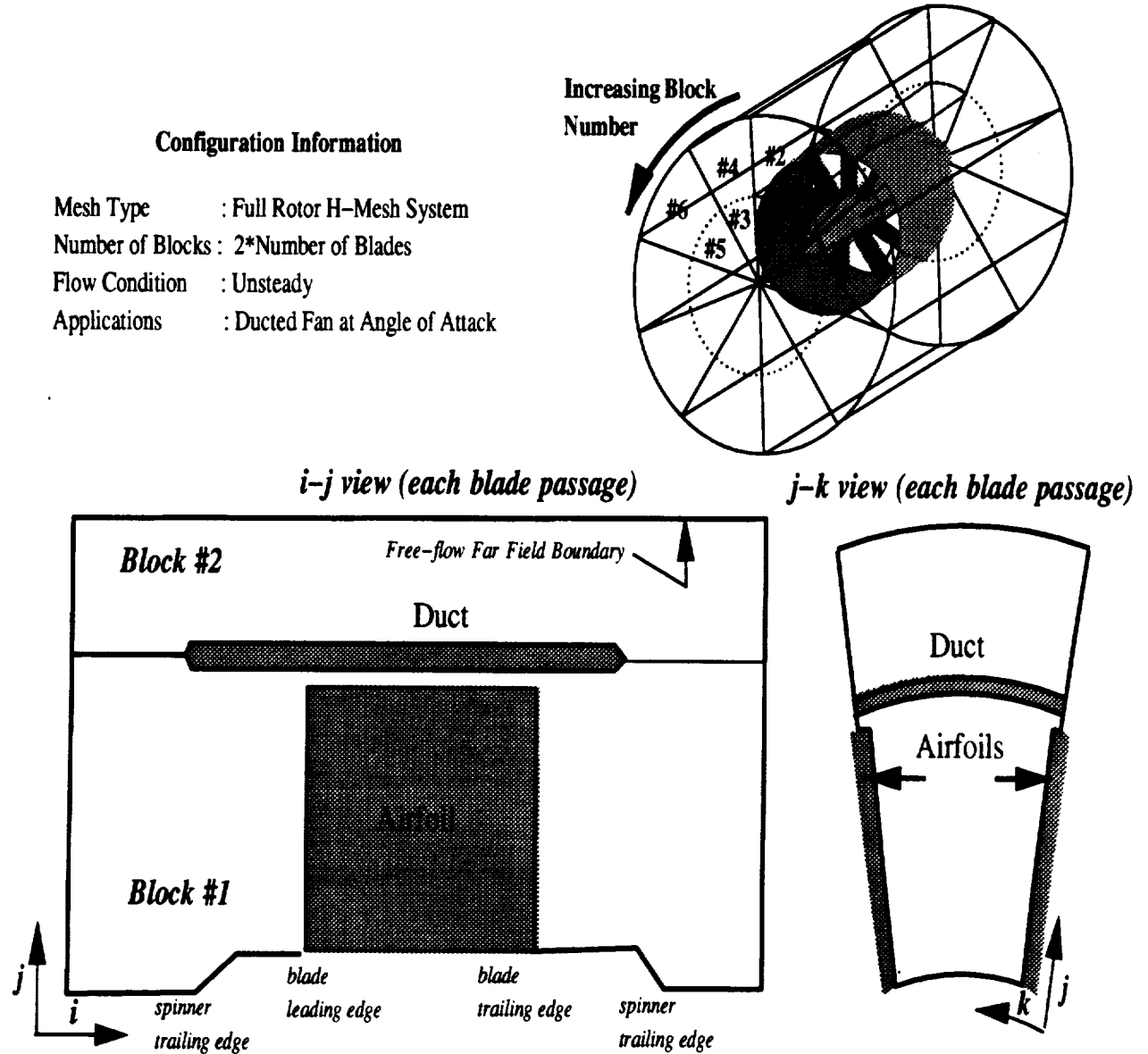*trailing edge*

Duct

Airfoils

k *j*

Figure 5.5: Standard Configuration #5 Geometry and Multiple Block Mesh Structure

150

to construct meshes with 2-D mesh blocks in the far field to reduce this inefficiency). Another disadvantage of this mesh system is the complexity of the boundary specifications required to couple the mesh blocks, and the possibility of poor convergence as a result of the many inner domain block boundaries.

Standard Configuration #7

Title:	Multiple Blade Row, Multiple Passage H-Grid
Mesh Type:	Single Block H-grid per Blade Passage
Flow Type:	Unsteady
Geometry:	Counterrotating propeller, profan, unducted fan, compressor
Number of Blade Rows:	Arbitrary
Mesh Generation Program:	*TIGG3D* and *ROTGRID*, or others

Standard Configuration #7 is a solution technique for analyzing the unsteady flow through a multiple blade row turbomachine. This configuration would be appropriate for the analysis of a counterrotating unducted fan, or for the internal flow in a multistage turbomachine. The mesh structure and boundary conditions are generated based on the time-dependent rotor-stator interaction solution approach discussed in Section 2.2. Several blade passages of each blade row are modeled, such that the total circumferential pitch for each blade row is constant. This implies that the number of blades in each row must be reducible to simple integer ratios (i.e. 3:4 for two blade rows, 3:4:6 for three blade rows, etc.), otherwise the number of passages which must be modeled for each blade row would be excessive. Each blade passage is discretized by a single H-type mesh block, similar to the mesh system described by Standard Configuration #1. The mesh block structure and numbering scheme is illustrated in Figure 5.7. The mesh block for a given blade passage is spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax) of each mesh block. At the interface between blade rows, the mesh surfaces share a common surface of revolution, and the axial and radial distribution of grid points in each mesh are identical at the

151

ADPAC-AOACR Standard Configuration #6
Full Rotor Ducted Fan C-H Grid

Configuration Information

Mesh Type	: Full Rotor C-H Mesh System
Number of Blocks	: 5*Number of Blades
Flow Condition	: Unsteady
Applications	: Ducted Fan at Angle of Attack

i-j view (each blade passage)
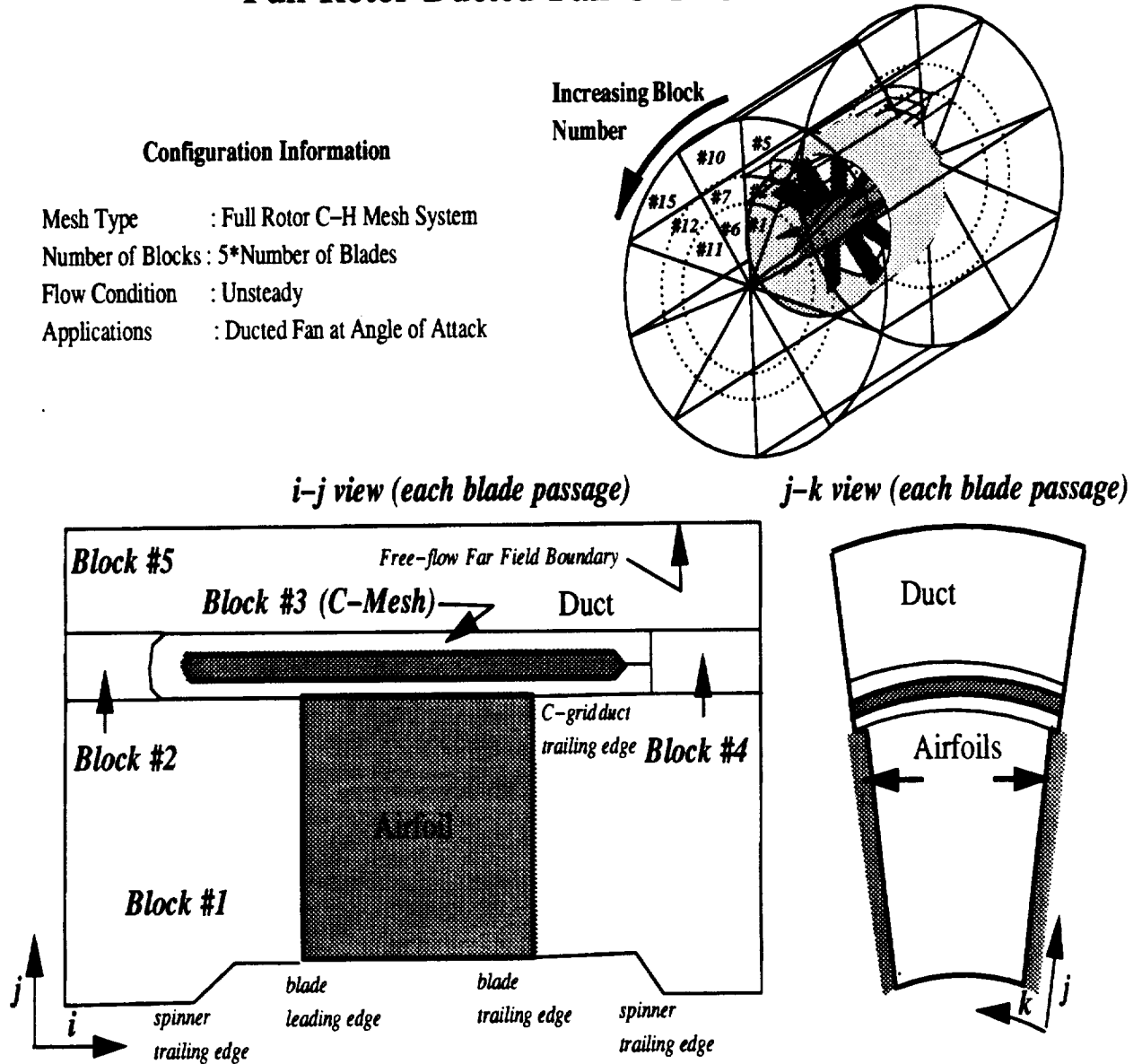
j-k view (each blade passage)

Figure 5.6: Standard Configuration #6 Geometry and Multiple Block Mesh Structure

152

interblade row interface (this reduces the interpolation of information between blade rows to the circumferential direction only).

Standard Configuration #8

Title:	Multiple Blade Row Multiple Passage Ducted Fan H-Grid
Mesh Type:	Two Block H-grid per Blade Passage
Flow Type:	Unsteady
Geometry:	Ducted fan
Number of Blade Rows:	Arbitrary
Mesh Generation Program:	*TIGG3D* followed by *ROTGRID*, or others

Standard Configuration #8 is a solution technique for analyzing the unsteady flow through a multiple blade ducted fan configuration, as shown in Figure 5.8. The mesh structure and boundary conditions are generated based on the time-dependent rotor-stator interaction solution approach discussed in Section 2.2. Several blade passages of each blade row are modeled, such that the total circumferential pitch for each blade row is constant. This implies that the number of blades in each row must be reducible to simple integer ratios (i.e. 3:4 for two blade rows, 3:4:6 for three blade rows, etc.), otherwise the number of passages which must be modeled for each blade row would be excessive. Each blade passage is discretized by a two H-type mesh blocks, similar to the mesh system described by Standard Configuration #2. The mesh block structure and numbering scheme is illustrated in Figure 5.8. The mesh blocks for a given blade passage are spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax) of the inner mesh block. For each blade passage, the two mesh blocks are divided by the duct and a pair of mating surfaces which extend upstream and downstream from the duct leading and trailing edges. The duct itself is therefore represented as a gap between the two mesh blocks. The mesh points defining the mating surfaces are common to both mesh blocks. The outer meshes are aligned circumferentially with the outer mesh distribution of the inner block. At the interface between blade rows, the mesh surfaces share a common

153

ADPAC-AOACR Standard Configuration #7
Multiple Blade Row/Multiple Passage Turbomachinery H-Grid

Configuration Information

Mesh Type	: Multiple Passage/Multiple Blade Row H-Grid
Number of Blocks	: Case Dependent Based on Number of Blade Rows and Number of Passages per Blade Row
Flow Condition	: Unsteady
Applications	: Turbomachinery Rotor/Stator Interaction

Multistage Turbomachine

Increasing Block Number

i-j view

Can specify blade base – hub clearance

time/space interpolation at blade row interface planes

Can specify blade tip – tip clearance

j-k view

Block #2   Block #1

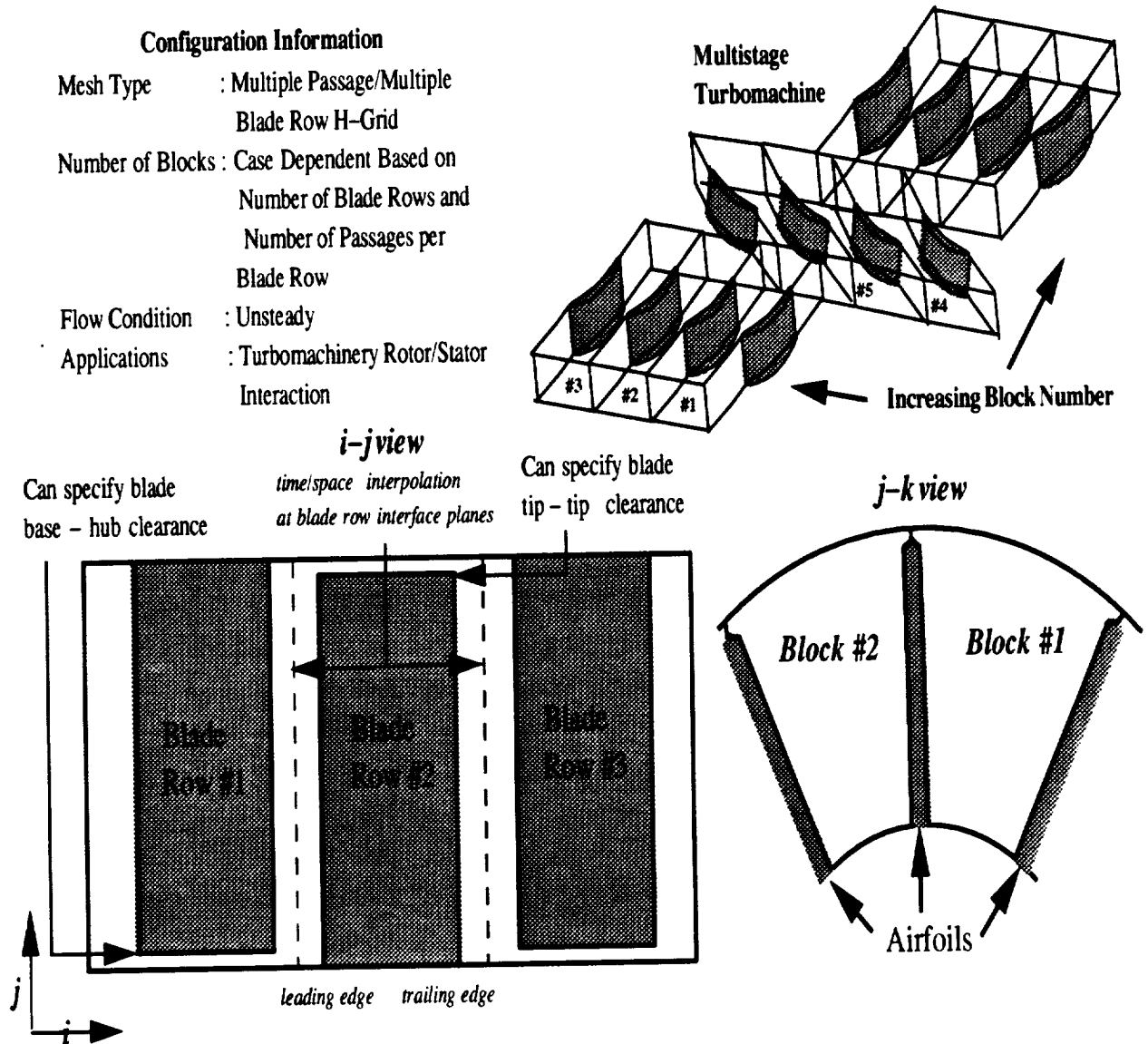Airfoils

leading edge   trailing edge

Figure 5.7:   Standard Configuration #7 Geometry and Multiple Block Mesh Structure

154

surface of revolution, and the axial and radial distribution of grid points in each mesh are identical at the interblade row interface (this reduces the interpolation of information between blade rows to the circumferential direction only).

<div align="center">

Standard Configuration #9

</div>

Title:	Multiple Blade Row Circumferential-Average H-Grid
Mesh Type:	Single Block H-grid per Blade Row
Flow Type:	Steady
Geometry:	Counterrotating propeller, profan, unducted fan, compressor
Number of Blade Rows:	Arbitrary
Mesh Generation Program:	*TIGG3D* and *ROTGRID*, or others

Standard Configuration #9 is a solution technique for analyzing the steady flow through a multiple blade row turbomachine. This configuration would be appropriate for the analysis of a counterrotating unducted fan, or for the internal flow in a multistage turbomachine. The mesh structure and boundary conditions are generated based on the mixing plane concept solution approach discussed in Section 2.2. A single blade passage in each blade row is modeled, with a surface of revolution mixing plane as the interface between blade rows. Each blade passage is discretized by a single H-type mesh block, similar to the mesh system described by Standard Configuration #1. The mesh block structure and numbering scheme is illustrated in Figure 5.9. The mesh block for a given blade passage is spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax) of each mesh block. At the interface between blade rows, the mesh surfaces share a common surface of revolution, and the axial and radial distribution of grid points in each mesh are identical at the interblade row interface (this simplifies the circumferential averaging operator).

<div align="center">

Standard Configuration #10

</div>

Title:	Multiple Blade Row Circumferential Average Ducted Fan H-Grid

<div align="center">

155

</div>

ADPAC-AOACR Standard Configuration #8
Multiple Blade Row/Multiple Passage Ducted Fan H-Grid

Configuration Information

Mesh Type   ; Two Block H-Grid

Number of Blocks:Depends on the number
of blade rows and the
number of passages per row

Flow Condition : Unsteady

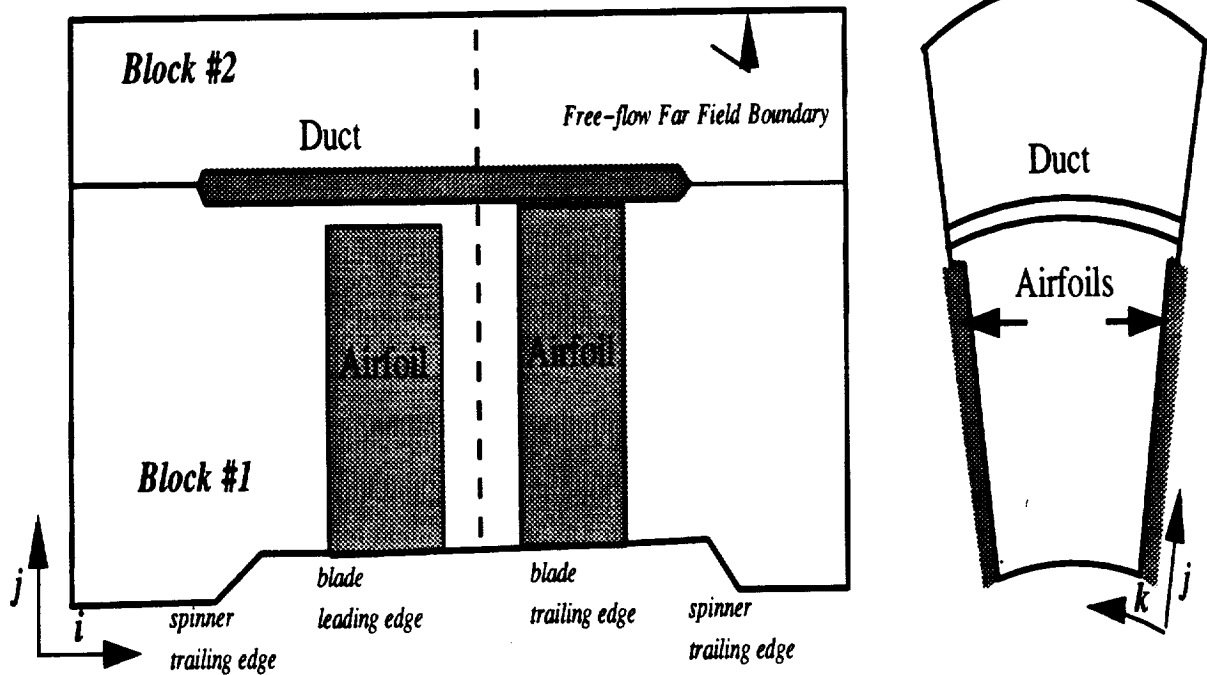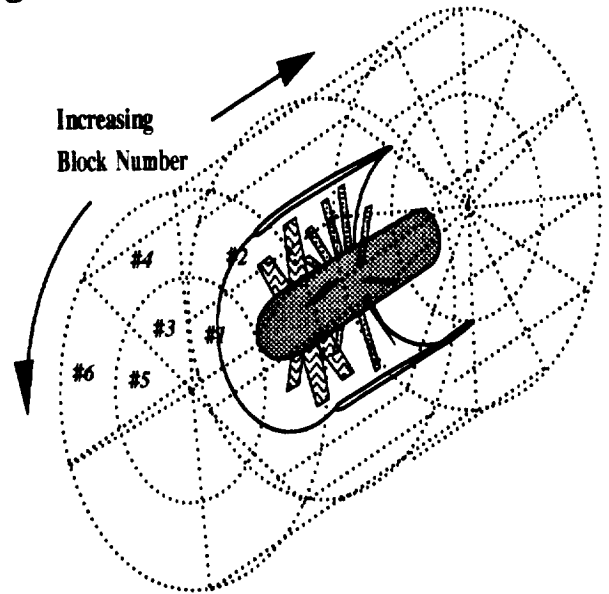Applications   : Rotor/Stator interaction for
a ducted fan

Increasing
Block Number

#4 #2

#3 #7

#6 #5

Block #2

Duct

Free-flow Far Field Boundary

Airfoil   Airfoil

Block #1

Duct

Airfoils

j

i

spinner
trailing edge

blade
leading edge

blade
trailing edge

spinner
trailing edge

k   j

Figure 5.8: Standard Configuration #8 Geometry and Multiple Block Mesh Structure

ADPAC-AOACR Standard Configuration #9
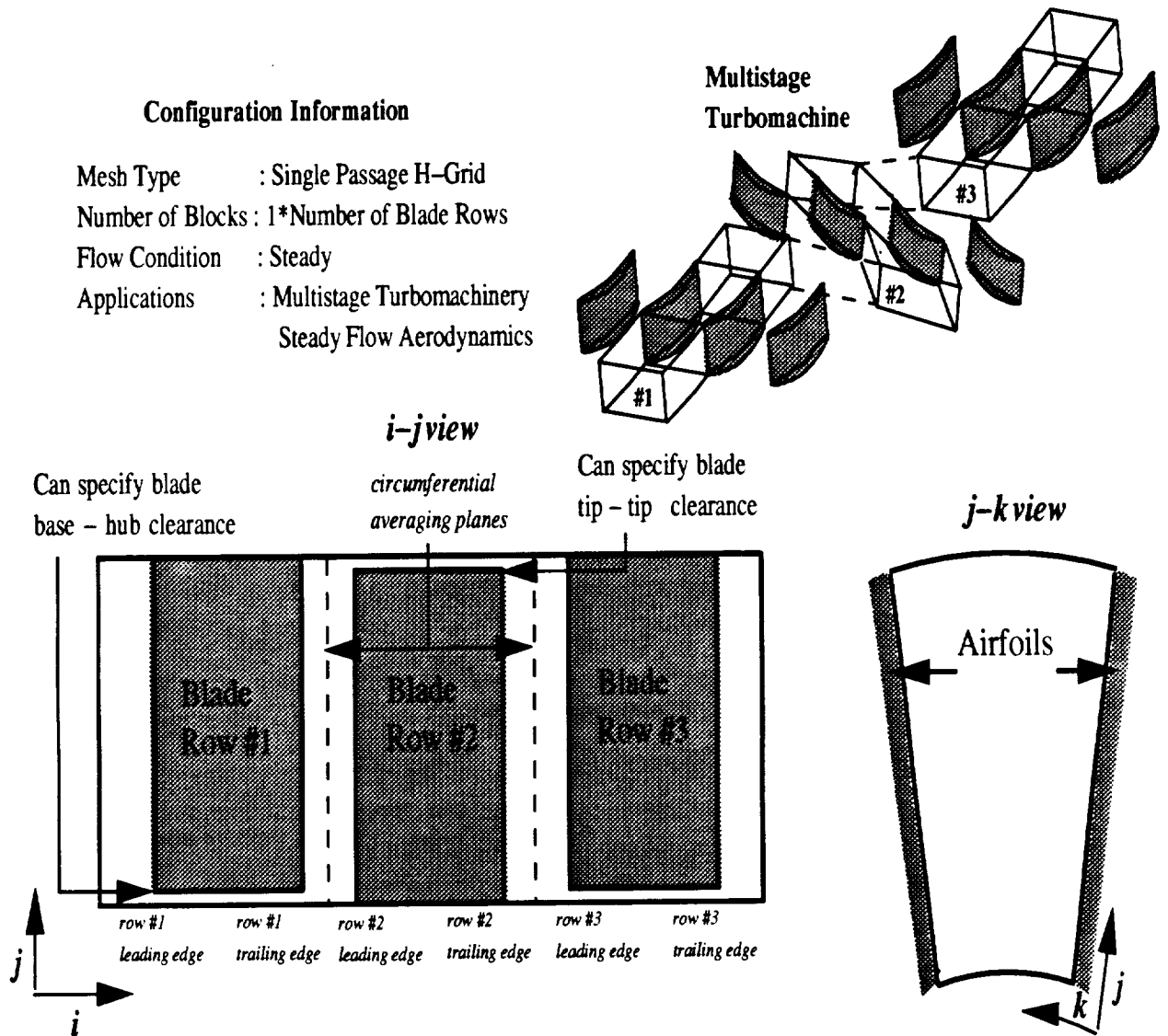# Multiple Blade Row Turbomachinery H-Grid
Using Circumferential Averaging

**Multistage
Turbomachine**

Configuration Information

Mesh Type	: Single Passage H–Grid
Number of Blocks	: 1*Number of Blade Rows
Flow Condition	: Steady
Applications	: Multistage Turbomachinery
	Steady Flow Aerodynamics

i–j view

Can specify blade
base – hub clearance

*circumferential
averaging planes*

Can specify blade
tip – tip   clearance

j–k view

Airfoils

Blade
Row #1

Blade
Row #2

Blade
Row #3

row #1	*row #1*	*row #2*	*row #2*	*row #3*	*row #3*
leading edge	*trailing edge*	*leading edge*	*trailing edge*	*leading edge*	*trailing edge*

j

i

k

j

Figure 5.9:   Standard Configuration #9 Geometry and Multiple Block Mesh Structure

Mesh Type:	Two Block H-grid per Blade Row
Flow Type:	Steady
Geometry:	Ducted fan
Number of Blade Rows:	Arbitrary
Mesh Generation Program:	*TIGG3D* followed by *ROTGRID*, or others

Standard Configuration #10 is a solution technique for analyzing the steady flow through a multiple blade ducted fan configuration, as shown in Figure 5.10. The mesh structure and boundary conditions are generated based on the mixing plane concept solution approach discussed in Section 2.2. A single blade passages of each blade row is modeled. Each blade passage is discretized by a two H-type mesh blocks, similar to the mesh system described by Standard Configuration #2. The mesh block structure and numbering scheme is illustrated in Figure 5.10. The mesh blocks for a given blade passage are spatially periodic, except in the vicinity of the blades themselves. The blades are represented as a circumferential displacement on the periodic surfaces ($k=1$ and $k=$kmax) of the inner mesh block. For each blade passage, the two mesh blocks are divided by the duct and a pair of mating surfaces which extend upstream and downstream from the duct leading and trailing edges. The duct itself is therefore represented as a gap between the two mesh blocks. The mesh points defining the mating surfaces are common to both mesh blocks. The outer meshes are aligned circumferentially with the outer mesh distribution of the inner block. At the interface between blade rows, the mesh surfaces share a common surface of revolution, and the axial and radial distribution of grid points in each mesh are identical at the mixing plane (this simplifies the circumferential averaging operator).

ADPAC-AOACR Standard Configuration #10
Multiple Blade Row/Circumferential Average Ducted Fan H-Grid

Configuration Information

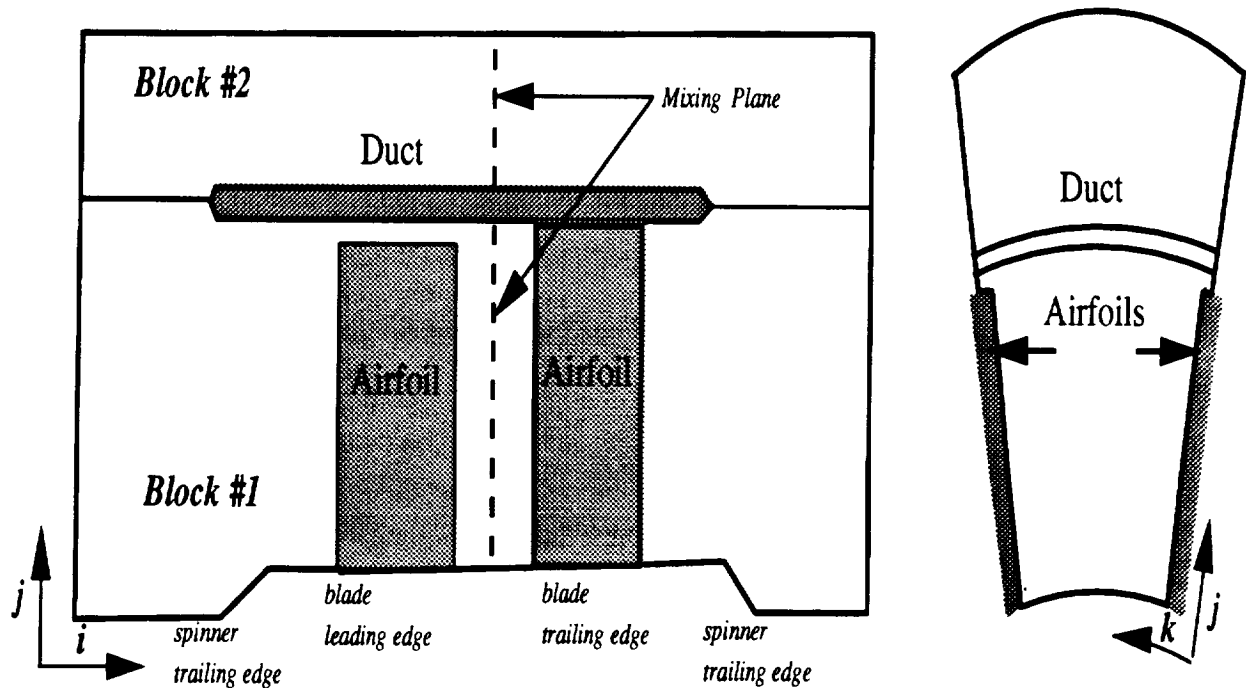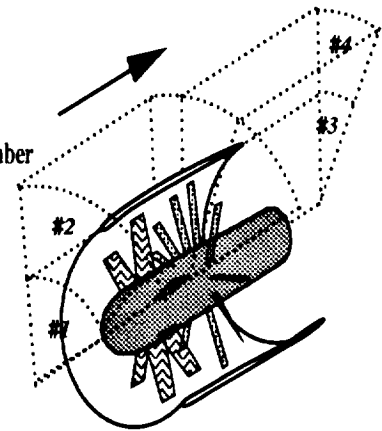|	
Mesh Type	; Two Block H-Grid
Number of Blocks:	2*Number of Blade Rows
Flow Condition	: Steady
Applications	: Multistage Analysis of
	a Ducted Fan Using Circumferential
	Averaging

Increasing
Block Number

Figure 5.10:   Standard Configuration #10 Geometry and Multiple Block Mesh Structure

6. *ROTGRID* PROGRAM DESCRIPTION

The standard distribution for the *ADPAC-AOACR* program includes a program which aids the user in setting up mesh files for each of the standard configurations listed in Chapter 5.0. This program, referred to simply as *ROTGRID*, is an interactive, menu based program which queries the user for the specific information needed to properly configure a multiple-block mesh for the *ADPAC-AOACR* code for the given configuration. *ROTGRID* then goes about the task of collecting mesh coordinate data and writing a corresponding mesh file for the *ADPAC-AOACR* solution. *ROTGRID* is set up specifically for mesh files initially generated using the *TIGG3D* grid generation program. Under some circumstances, meshes generated by the *CHGRIDV2* program are also treated properly. If *ROTGRID* cannot handle a particular mesh system properly, the user can assemble an *ADPAC-AOACR* mesh from a collection of isolated mesh blocks by using the program *MAKEADGRID* described in Chapter 7.0.

6.1 Configuring Maximum Array Dimensions

Maximum array dimensions in the *ROTGRID* program are set by the FORTRAN PARAMETER statements listed in the file **parameter.inc** included with the *ROTGRID* source program. A sample **parameter.inc** file with the corresponding parameter descriptions would appear as:

```
C*****************************************************************************
C
C  PARAMETER DESCRIPTION FILE:
C
C    imx --- > maximum number of grid elements in the i coordinate direction
```

161

```
C                for any given mesh block
C    jmx --- > maximum number of grid elements in the j coordinate direction
C                for any given mesh block
C    kmx --- > maximum number of grid elements in the k coordinate direction
C                for any given mesh block
C    nblks - > maximum number of grid blocks for the rotated grid
C    mrowmx --> maximum number of blade rows for the rotated grid
C
C---> Note: PARAMETER nrowmx must be >= 5.
C
         parameter(imx=225,jmx=50,kmx=91,nblks= 5,nrowmx=2)
```

6.2 Compiling the *ROTGRID* Program

. The *ROTGRID* program source directory contains a UNIX-based Makefile facility to automate compilation for a number of machines. In the directory containing the FORTRAN source of the *SETUP* code, compilation is performed by executing the command:

make *option*

The **make** command is standard on UNIX systems and automatically interrogates the file *Makefile* for instructions on how to perform the compilation. The *option* argument may be any of the variables listed below:

No argument - same as *link* below.

link This is the standard UNIX system compilation. This option will deliver a working executable on most UNIX systems which support standard naming conventions (*f77* as the standard compiler, etc.). The compilation includes basic compiler optimization (f77 -O).

cray This option is utilized when compiling the *ROTGRID* code on a Cray computer.

aix This option is used when compiling the *ROTGRID* standard code on an IBM RS-6000 workstation running the AIX operating system.

6.3   Running the *ROTGRID* Program

Once the code has been compiled, change directories to the location where the case of interest has been stored. The *ROTGRID* program requires that the mesh has already been generated, as it will seek out the mesh file *case.mesh*, where *case* is the case name used in the file naming convention (see Section 3.5).

The *ROTGRID* program is invoked by issuing the command:

*path/*rotgrid

where *path* is the relative or absolute pathname of the directory containing the *ROTGRID* executable file from the current local directory. For example, if the mesh file is in the directory

/usr/people/me/testcase

and the *ROTGRID* executable is in the directory

/usr/people/me/adpac-aoacr/src/rotgrid

then the commands

cd /usr/people/me/testcase
/usr/people/me/adpac-aoacr/src/rotgrid/rotgrid

would begin the *ROTGRID* program process.

Once the *ROTGRID* program is invoked, the user is asked to define the *case* name used in the file naming process (see Section 3.5). Following this, the user is asked to select the desired standard configuration. After the user selects one of the standard configurations, a series of specific questions pertaining to the requested configuration must be answered. Due to the large number of possibilities, it is not possible to explain all of the questions and answers in detail here. The responses to the program inquiries are intended to be fairly self-explanatory, and the user is urged to simply run the program for further details.

163

7. *MAKEADGRID* **PROGRAM DESCRIPTION**

The standard distribution for the *ADPAC-AOACR* program includes a program called *MAKEADGRID* which aids the user in setting up a multiple-block mesh file from isolated unformatted mesh files. This program is useful for creating *ADPAC-AOACR*-compatible multiple-block meshes from output from grid generation programs which are not supported by *ROTGRID* or which do not utilize the Scientific Database Library (SDBLIB). The *MAKEADGRID* program is an interactive program which queries the user for the number of blocks to be assembled for the final mesh, and then requests a file name for each of the individual mesh blocks. The user is then requested to name the final output file for the *ADPAC-AOACR*-compatible multiple-block mesh. The individual mesh blocks are assembled in the order in which the mesh file names are specified, so care must be taken to order these names appropriately.

7.1   Configuring Maximum Array Dimensions

Maximum array dimensions in the *MAKEADGRID* program are set by the FORTRAN PARAMETER statements listed in the source file **makeadgrid.f** included with the standard distribution. The PARAMETER statement and the descriptions of the various parameter variables appear at the top of the file as:

```
C
C
C makeadgrid: This program assembles an ADPAC-compatible mesh file
C             from selected other unformatted PLOT3D mesh files
C
C
C  Set parameter size for max grid block to be read in
C
```

165

```
C    imax --- > maximum number of grid elements in the i coordinate direction
C              for any given mesh block
C    jmax --- > maximum number of grid elements in the j coordinate direction
C              for any given mesh block
C    kmax --- > maximum number of grid elements in the k coordinate direction
C              for any given mesh block
C    nnames - > maximum number of grid blocks for final mesh
C

     parameter(imax=251, jmax=82, kmax=53)
     parameter(nnames = 100 )
```

7.2 Compiling the *MAKEADGRID* Program

· The *MAKEADGRID* program source directory contains a UNIX-based Makefile facility to automate compilation for a number of machines. In the directory containing the FORTRAN source of the *MAKEDGRID* code, compilation is performed by executing the command:

make *option*

The **make** command is standard on UNIX systems and automatically interrogates the file *Makefile* for instructions on how to perform the compilation. The *option* argument may be any of the variables listed below:

No argument - same as *link* below.

link This is the standard UNIX system compilation. This option will deliver a working executable on most UNIX systems which support standard naming conventions (*f77* as the standard compiler, etc.). The compilation includes basic compiler optimization (f77 -O).

cray This option is utilized when compiling the standard code on a Cray computer.

aix This option is used when compiling the standard code on an IBM RS-6000 workstation running the AIX operating system.

166

7.3 Running the *MAKEADGRID* Program

Once the code has been compiled, change directories to the location where the case of interest has been stored. The *MAKEADGRID* program requires that each individual mesh block for the final mesh be stored separately as a single-grid unformatted *PLOT3D* file .

The *MAKEADGRID* program is invoked by issuing the command:

path/makeadgrid

where *path* is the relative or absolute pathname of the directory containing the *MAKEADGRID* executable file from the current local directory. For example, if the mesh file is in the directory

/usr/people/me/testcase

and the *MAKEADGRID* executable is in the directory

/usr/people/me/adpac-aoacr/src/makeadgrid

then the commands

cd /usr/people/me/testcase
/usr/people/me/adpac-aoacr/src/makeadgrid/makeadgrid

would begin the *MAKEADGRID* program process.

7.4 Sample Session Using the *MAKEADGRID* Program

A sample session using the *MAKEADGRID* program for the mesh illustrated in Figure 2.4 is given below. In this case, the mesh was originally generated using a proprietary mesh generation program, and hence, required some manipulation in order to construct the multiblock mesh for an *ADPAC-AOACR* solution. The mesh consists of 3 mesh blocks (the O-grid about the airfoil, and 2 H-grid caps upstream and downstream of the O-grid) named **block1.mesh, block2.mesh**, and **block3.mesh..**

The *MAKEADGRID* session used to create the final mesh named **vbivane.mesh** is listed below. The user responses to the *MAKEADGRID* program are given in boldfaced type.

```
*************************************************

 MAKEADGRID - construction program for
              creating ADPAC-AOACR mesh files
              from selected PLOT3D unformatted
              mesh files.

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



 Enter the number of blocks

   3


 Enter the name of the           1 grid to process

 (Remember: each file must be unformatted PLOT3D style


   block1.mesh


 Enter the name of the           2 grid to process

 (Remember: each file must be unformatted PLOT3D style


   block2.mesh


 Enter the name of the           3 grid to process

 (Remember: each file must be unformatted PLOT3D style
```

block2.mesh

Getting grid sizes and extra info from grid files

Loop=	1 mg=	0 il,jl,kl=	129	33
	33			
Loop=	2 mg=	0 il,jl,kl=	17	33
	17			
Loop=	3 mg=	0 il,jl,kl=	17	33
	17			

Enter the file name for the final grid

vbivane.mesh

Final grid data in file
vbivane.mesh

Output file array size

Loop =	1 -->	129	33	33
Loop =	2 -->	17	33	17
Loop =	3 -->	17	33	17

Array sizes output to final file
Reading Grid Data from file
block1.mesh
il, jl, kl --->   129   33   33
Output grid data to final file
Reading Grid Data from file
block2.mesh
il, jl, kl --->   17   33   17
Output grid data to final file
Reading Grid Data from file
block3.mesh

il, jl, kl --->   17   33   17
Output grid data to final file

PROGRAM COMPLETED NORMALLY

8. *SETUP* PROGRAM DESCRIPTION

The standard distribution for the *ADPAC-AOACR* program includes a program which aids the user in setting up an input file and a boundary data file for each of the standard configurations listed in Chapter 5.0. This program, referred to simply as *SETUP*, is an interactive, menu based program which queries the user for the specific information needed by the *ADPAC-AOACR* code for the given configuration, and then goes about the task of writing a corresponding input and boundary data file for an *ADPAC-AOACR* solution.

8.1   Configuring Maximum Array Dimensions

Maximum array dimensions in the *SETUP* program are set by the FORTRAN PARAMETER statements listed in the file **parameter.inc** included with the *SETUP* source program. A sample **parameter.inc** file would appear as:

```
parameter( nbmax  = 100 )
parameter( ninvar = 100 )
```

The PARAMETER variable NBMAX determines the maximum number of mesh blocks which can be manipulated using *SETUP*. The PARAMETER variable NIN-VAR determines the maximum number of input file variables which is permitted and need not be changed unless the *SETUP* program is modified to incorporate additional input file variables (which would then require a corresponding modification to the *ADPAC-AOACR* program).

8.2   Compiling the *SETUP* Program

The *SETUP* program source directory contains a UNIX-based Makefile facility to automate compilation for a number of machines. In the directory containing the FORTRAN source of the *SETUP* code, compilation is performed by executing the command:

make *option*

The **make** command is standard on UNIX systems and automatically interrogates the file *Makefile* for instructions on how to perform the compilation. The *option* argument may be any of the variables listed below:

No argument - same as *link* below.

link   This is the standard UNIX system compilation. This option will deliver a working executable on most UNIX systems which support standard naming conventions (*f77* as the standard compiler, etc.). The compilation includes basic compiler optimization (f77 -O).

cray   This option is utilized when compiling the *SETUP* code on a Cray computer.

aix   This option is used when compiling the *SETUP* code on an IBM RS-6000 workstation running the AIX operating system.

8.3   Running the *SETUP* Program

Once the code has been compiled, change directories to the location where the case of interest has been stored. The *SETUP* program requires that the mesh has already been generated, as it will seek out the mesh file *case.mesh*, where *case* is the case name used in the file naming convention (see Section 3.5).

The *SETUP* program is invoked by issuing the command:

*path/*setup

where *path* is the relative or absolute pathname of the directory containing the *SETUP* executable file from the current local directory. For example, if the mesh file is in the directory

172

```
/usr/people/me/testcase
```

and the *SETUP* executable is in the directory

```
/usr/people/me/adpac-aoacr/src/setup
```

then the commands

```
cd /usr/people/me/testcase
/usr/people/me/adpac-aoacr/src/setup/setup
```

would begin the *SETUP* program process.

Once the *SETUP* program is invoked, the user is first requested to enter the *case* name used in the *ADPAC-AOACR* file naming convention (see Section 3.5). Following a series of questions used to define the flow condition, the user indicates which standard configuration is desired, and the *SETUP* program proceeds from there to construct a standard input file and a boundary data file for the given configuration based on the user input. The grid information indicated in the appropriate standard configuration illustration (Figures 5.1-5.10) should be available, as well as the desired flow conditions. The interactive questions and responses for the *SETUP* program are intended to be self-explanatory. and the user is urged to simply run the program for further details.

9. INTERACTIVE GRAPHICS DISPLAY

The *ADPAC-AOACR* program is equipped with an option which permits real time interactive graphics display of flow data in the form of colored contours or velocity vectors on geometries represented by wiremesh grid surfaces. The interactive graphics are based largely on routines generated from the *PLOT3D* visualization program, and many of the features of this option should be familiar to anyone who has used *PLOT3D*. All interactive graphics must be displayed on a Silicon Graphics workstation, IRIX Operating System 4.0.1 or above. The graphics display can be operated on a single computing platform, or can be directed across a network for specific computer hardware configurations. Thus, it is possible to have a job running remotely on a Cray computer, with interactive graphics displayed locally on a network-connected Silicon Graphics workstation. When operating across a network which involves a non-Silicon Graphics computer, the communication program *AGTPLT-LCL* must be running on the local display device in order to capture the graphics commands issued by the remote compute server (details on *AGTPLT-LCL* are given below). A graphic illustrating the possible graphics display operating modes is given in Figure 9.1. It should be mentioned that the interactive graphics display was not a contracted feature, and was actually developed to aid in debugging the multiple block code. The description of this feature is included in this manual for completeness, but the user should be cautioned due to the immature nature of this portion of the code.

9.1 Setting up the Program

The first step in producing the real time interactive graphics display is to correctly compile the code to include the graphics libraries. This is accomplished by utilizing the appropriate option in the *ADPAC-AOACR* Makefile command (see Section 3.4).

ADPAC-AOACR Interactive Graphics Display
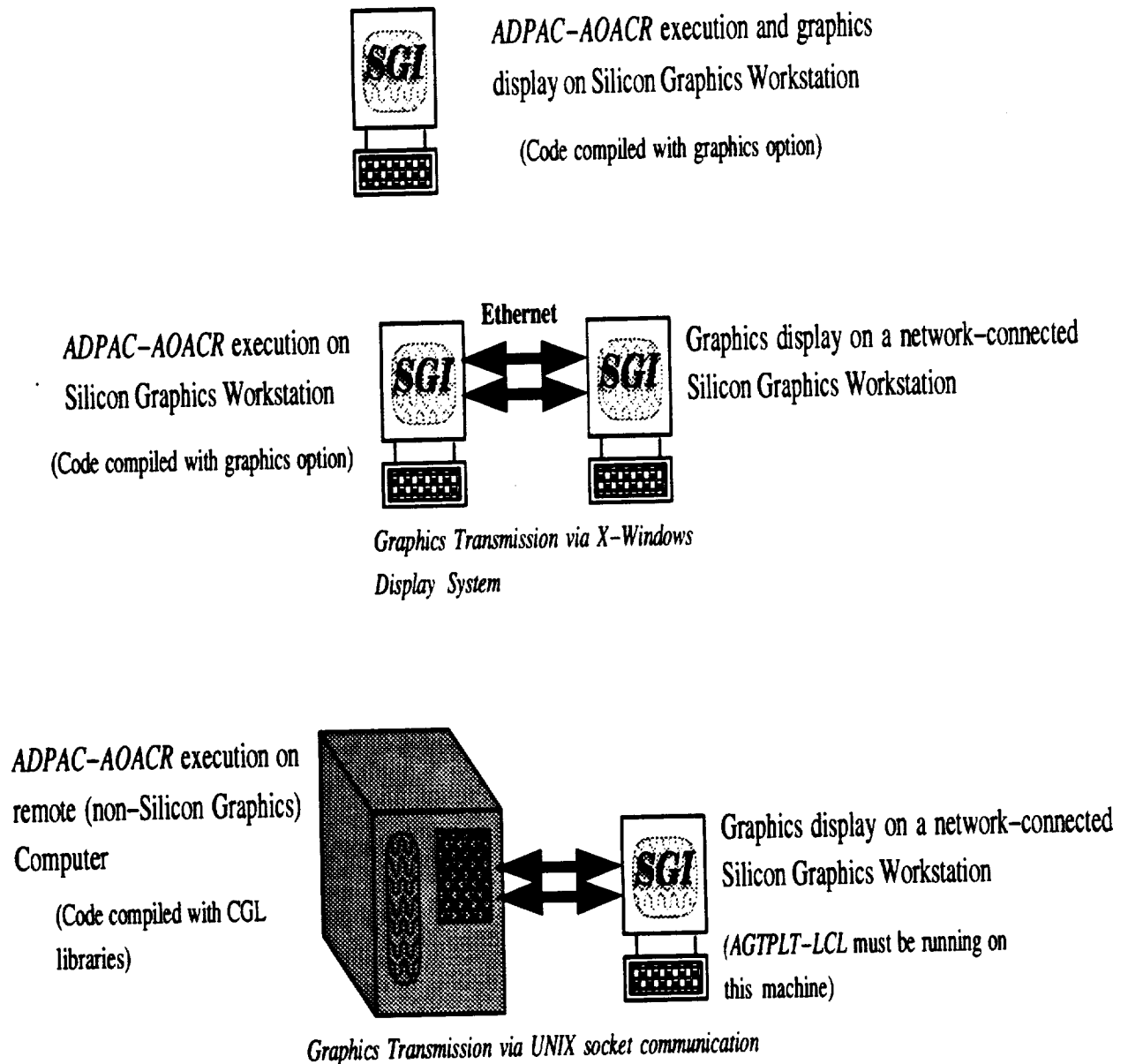## Computer Network Configuration Options

ADPAC-AOACR execution and graphics
display on Silicon Graphics Workstation

(Code compiled with graphics option)

ADPAC-AOACR execution on
Silicon Graphics Workstation

(Code compiled with graphics option)

Ethernet

Graphics display on a network-connected
Silicon Graphics Workstation

Graphics Transmission via X-Windows
Display System

ADPAC-AOACR execution on
remote (non-Silicon Graphics)
Computer

(Code compiled with CGL
libraries)

Graphics display on a network-connected
Silicon Graphics Workstation

(AGTPLT-LCL must be running on
this machine)

Graphics Transmission via UNIX socket communication

Figure 9.1: *ADPAC-AOACR* Interactive Graphics Display Network Configuration
Options

The valid graphics options include *graphics, pfagraphics, craygraphics, aixgraphics,* and *craygraphdbx.* These options incorporate various levels of the included graphics libraries for execution on various machines (again, see Section 3.4 for specific Makefile details).

Once the code has been correctly compiled to include the graphics libraries, several input parameters must be correctly initiated to engage the graphics subroutines during the execution of the code. The input keyword **FGRAFIX** must have a value of 1.0 to initiate any graphics instructions. The keyword **FGRAFINT** determines the number of time-marching iterations between graphics window updates. The keyword **FIMGSAV** is a trigger (0.0 - off, 1.0 - on) which determines whether periodic image capturing is enabled, and the keyword **FIMGINT** determines the number of time-marching iterations between image captures. Additional details concerning these input file keywords are available in Section 3.6.

9.2   Graphics Window Operation

Once the graphics window has been initiated on the local display, and the initial data has been plotted, the program continues and the graphics display data are updated every **FGRAFINT** iterations. This process will continue until the program terminates, or until the user interrupts the process by pressing the left mouse button once with input focus directed to the graphics display window. A short time later, (the delay may be quite long for a network which is burdened), the graphics display will freeze, and the computational portions of the program will be suspended in order to permit the user to interactively translate, rotate, or scale the graphics image to their liking. When the display has been frozen, the viewpoint of the display may be altered by one of several mouse controls. The left mouse button controls rotation, the right mouse button controls translation, and the middle mouse button controls scaling (zoom in, zoom out). The controlling mouse movements are illustrated in Figure 9.2. The mouse-directed viewpoint controls are identical to those used in *PLOT3D* [11]. Once the viewpoint has been altered, program control is returned to *ADPAC-AOACR* by hitting the ENTER key on the keyboard with input focus directed to the graphics window. At this point, the code will then return to the process of performing time-

177

marching iterations, with periodic updating of the graphics screen.

It is also possible for the user to change the plotting function by entering any one of the following characters with input focus directed to the graphics window at any time during the process:

Key Result
   p Set flow function to pressure contours
   2 Set flow function to velocity vectors

The surfaces plotted by the interactive graphics display is currently hardwired in the code. A wiremesh representation and the corresponding surface contours are generated for the $i=1$, $j=1$, and $k=1$ mesh surfaces. This restriction could be removed in future developments.

9.3 AGTPLT-LCL Program Description

The program $AGTPLT$-LCL is the receiving program for local graphics display of an $ADPAC$-$AOACR$ job running on a remote, network-connected computing platform. The $AGTPLT$-LCL program is a modified version of the NASA-AMES developed $PLOT3D$-LCL program. This program can only be run on a Silicon Graphics Workstation running at level 4.0.1 (or above) of the IRIX operating system. As such, compilation of the $AGTPLT$-LCL program has no options, and is performed simply by executing the command make in the $AGTPLT$-LCL source directory. Once initiated, the $AGTPLT$-LCL program waits for an outside process from $ADPAC$-$AOACR$ to communicate with the local workstation, and graphics commands received from the remote job are displayed locally.

An important consideration in setting up a remote calculation with local graphics display using $AGTPLT$-LCL is the manner in which the local display is defined in the calculation. The CGL libraries used to permit the network graphics instructions require an internet network address in order to properly transmit the graphics commands to the correct destination. This definition should be provided in the standard input file following the normal keyword parameters (see Section 3.6 for a sample file and keyword definitions). At the end of the standard input keyword data, the user

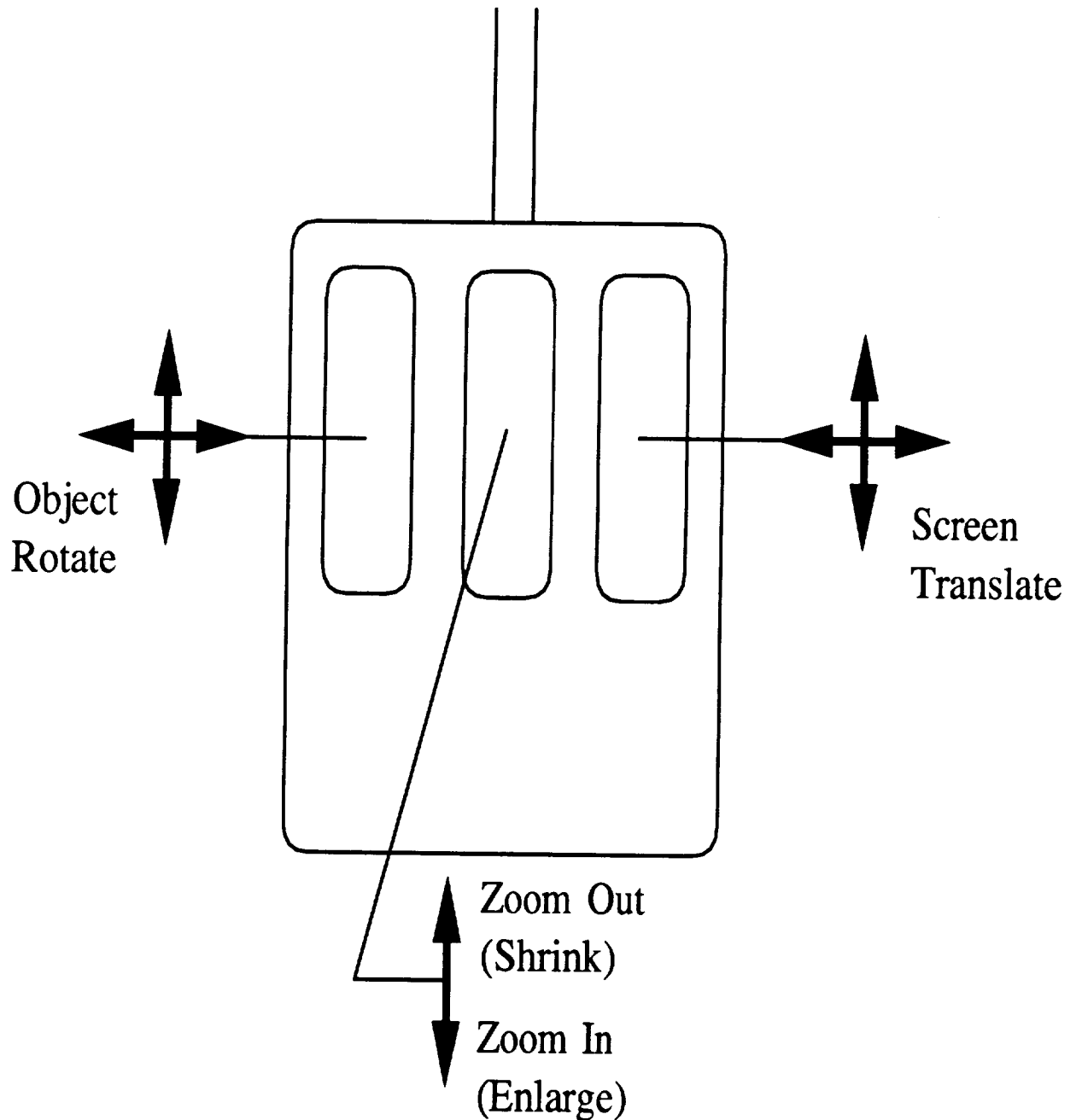# ADPAC-AOACR Interactive Graphics Display Mouse Control

Figure 9.2:   *ADPAC-AOACR* Interactive Graphics Display Mouse Control

should use an **ENDINPUT** statement to terminate the normal input stream. The **ENDINPUT** statement should then be followed by two blank lines, and then a line containing the destination network address of the local Silicon Graphics display device. This specification will ultimately be read by the CGL libraries in setting up the network connection.

The procedure to set up this network-connected graphics display option would be to start the job on the remote machine, and then immediately start the *AGTPLT-LCL* program on the local display. As long as the correct network address has been entered in the *case*.input file, then the remote program should begin communicating with the *AGTPLT-LCL* program, and the local graphics window will begin displaying the graphics instructions specified by the remote computing program.

REFERENCES

[1] Hall, E. J. and Delaney, R. A., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task V - Unsteady Counterrotation Ducted Propfan Analysis, Final Report", NASA CR 187126, NASA Contract NAS3-25270, 1992.

[2] Hall, E. J., Delaney, R. A., and Bettner, J. L., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task II - Unsteady Ducted Propfan Analysis, Final Report", NASA CR 187106, NASA Contract NAS3-25270, 1991.

[3] Hall, E. J., Delaney, R. A., and Bettner, J. L., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task II - Unsteady Ducted Propfan Analysis, Computer Program Users Manual", NASA CR 187105, NASA Contract NAS3-25270, 1991.

[4] Rao, K. V., and Delaney, R. A., 1990, "Investigation of Unsteady Flow Through a Transonic Turbine Stage: Part I- Analysis", AIAA Paper 90-2408.

[5] Jorgenson, P. C. E., and Chima, R. V., "An Unconditionally Stable Runge-Kutta Method for Unsteady Flows," AIAA Paper 89-0205, 1989.

[6] Adamczyk, J. J., "Model Equation for Simulating Flows in Multistage Turbomachinery," ASME Paper 85-GT-226, 1985.

[7] Dawes, W.N., "Multi-Blade Row Navier-Stokes Simulations of Fan Bypass Configurations", ASME Paper 91-GT-148, 1991.

[8] Crook, A. J., and Delaney, R. A., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task IV - Advanced Fan Section Analysis, Final Report", NASA CR 187128, NASA Contract NAS3-25270, 1992.

[9] Rai, M. M., "Unsteady Three-Dimensional Navier-Stokes Simulations of Turbine Rotor-Stator Interaction," AIAA Paper 87-2058, 1987.

[10] Whipple, D., "BDX-Binary Data Exchange Preliminary Information", NASA-Lewis Research Center, 1989.

[11] Walatka, P. P., and Buning, P. G., "PLOT3D User's Manual,", rough draft for NASA TM, 1988.

[12] Plessel, Todd, "SURF User's Guide,", NASA Ames Research Center, 1988.

[13] Walatka, P. P., and Buning, P. G., "FAST", NASA Ames Research Center, 1990.

[14] Hall, E. J., Delaney, R. A., and Bettner, J. L., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task I - Ducted Propfan Analysis", NASA CR 185217, NASA Contract NAS3-25270, 1990.

[15] Crook, A. J., and Delaney, R. A., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task III - Advanced Fan Grid Generation", NASA CR 187129, NASA Contract NAS3-25270, 1991.

APPENDIX A. *ADPAC-AOACR* DISTRIBUTION AND DEMONSTRATION INSTRUCTIONS

A.1 Introduction

This appendix describes the commands necessary to extract the *ADPAC-AOACR* source code and demo files from the standard distribution and run a complete test case for a ducted fan employing multiple blade rows. The standard *ADPAC-AOACR* distribution is a compressed *tar* file which can be decoded into the various parts by a sequence of commands on any standard UNIX system. The sequence listed below is intended to guide the user through the setup from the standard distribution up to and including a complete demonstration of a calculation for a ducted propfan employing multiple blade rows. The command sequence listed below should work on most systems employing the UNIX operating system. Since portions of this process are inherently machine-dependent, the exact commands listed here are for a Silicon Graphics Workstation running the IRIX Operating System, Revision 4.0.1. Alternate commands will be listed when a significant machine dependence exists.

A.2 Extracting the Source Files

The *ADPAC-AOACR* programs are distributed as a compressed *tar* file named

adpac-aoacr.tar.Z

This tar file requires roughly 22.0 megabytes of disk space. It should be possible to extract and run the code on any standard UNIX system from this distribution file. The first step necessary to extract the *ADPAC-AOACR* programs is to uncompress

the *tar* file with the command:

uncompress adpac-aoacr.tar.Z

This operation essentially replaces the compressed file *adpac-aoacr.tar.Z* with an uncompressed file *adpac-aoacr.tar*. The uncompressed *tar* file requires approximately 41.0 megabytes of disk space.

The next step is to extract the individual files and directories from the *adpac-aoacr.tar* file. The **tar** command will create a subdirectory named *adpac-aoacr* in the current directory, so it is up to the user to move the *adpac-aoacr.tar* file to a suitable initial directory before extracting the embedded subdirectories. Once the *tar* file is properly placed, the *ADPAC-AOACR* distribution may be extracted with the command

tar xvof adpac-aoacr.tar

(On some systems **tar xvf adpac-aoacr.tar** may be sufficient.) Execution of the UNIX list command **ls -l** will verify that the *adpac-aoacr* directory has been created. The complete extraction process will require about 90.0 Megabytes of disk space (to hold the *adpac-aoacr.tar* file and the extracted contents)..

A.3 Compiling the Source Code

After extracting the source files, the user is naturally interested in compiling the source files for execution. A UNIX-compatible *Make* facility is provided for each of the *ADPAC* programs. The *Makefile* which governs the compilation process is necessarily machine-dependent and requires that the user select from one of a number of preconfigured systems. The *Make* command is fully described in Section 3.4. If no option is specified in the *make* command, then the standard UNIX compilation is performed.

In order to begin the compilation, it is first necessary to enter the *adpac-aoacr* directory with the command:

cd adpac-aoacr

At this point, several files and directories will be available. By entering the UNIX command **ls -l**, a listing of the individual directories can be obtained. The output of the **ls** command will look something like:

demo/   manual/   report/   src/

A description of each of these listings is given below:

demo   This directory contains several geometry and flow input files for generating sample runs of the *ADPAC* codes.

manual   This directory contains the *LaTeX* source code for this manual. If *LaTeX* is installed on your system, it is possible to reproduce this document (excluding figures) with the command **latex manual**. The resulting device independent file *manual.dvi* may then be converted to *PostScript* or previewed on screen through a number of widely available routines.

report   This directory contains the *LaTeX* source code for the final report outlining the technical details of the *ADPAC-AOACR* codes. If *LaTeX* is installed on your system, it is possible to reproduce the final report (excluding figures) with the command **latex finalreport**. The resulting device independent file *finalreport.dvi* may then be converted to *PostScript* or previewed on screen through a number of widely available routines.

src   This directory contains all the FORTRAN source code for the *ADPAC-AOACR* programs including *SETUP, ROTGRID, MAKEADGRID,* and *AGTPLT-LCL.*

It is now possible to compile the *ADPAC-AOACR* code by issuing the commands

cd adpac

make

(On a Cray, the command **make cray** is appropriate, while on an IBM workstation **make aix** is appropriate.) The compilation of the executable module for *ADPAC-AOACR* will require roughly 20 megabytes of disk space.

A.4   Running the Distribution Demonstration Test Case

Once the *make* facility has properly completed compiling the *ADPAC-AOACR* source code, it is possible to run the test cases provided with the standard distribution. It is recommended that the sample cases be tested to verify proper compilation and extraction of the *ADPAC-AOACR* distribution.

In order to run the demonstration cases, it is necessary to begin in the *demo* directory. From the *ADPAC-AOACR* source code directory, the *demo* directory may be entered by issuing the command

<div align="center">

cd ../../demo

</div>

Several test cases are provided with the standard distribution to illustrate the operation of the code for many different applications. The commands needed to run any demo are similar, so only the case listed under the directory **nasa** will be explained in detail here.

After entering the demo directory, an ls command will indicate that the following subdirectories are available:

```
nasa/     type1/     type2/     type3/     type4/
type5/    type6/     type7/     type8/     type9/
type10/   type11/
```

These subdirectories contain the ducted fan demonstration case described above, as well as a sample case (at least partially) for each of the 11 standard configurations described in Chapter 5.0. To run the multiple blade row ducted fan demonstration case, enter the *nasa* subdirectory by issuing the command **cd nasa**. Now, the ls command reveals:

```
nasa.input          nasa.boundata       nasa.mesh
nasa.output.save    nasa.converge.save
```

The nasa directory contains the data to run a test case for the NASA 1.15 pressure ratio ducted fan. This geometry is representative of a 25:1 bypass ratio turbofan engine fan, and has been tested extensively both experimentally and numerically. This test case employs two blade rows (a rotor and a stator) and the multiple blade rows are treated using the circumferential averaging technique described in Section 2.2. The mesh corresponds to Standard Configuration #10, and the mesh and appropriate mesh indices are illustrated in Figure A.1. The multiple-block mesh for this test case is contained in *nasa.mesh*, and may be viewed using the *PLOT3D* program. The flow Mach number is 0.75, and the calculation is performed at 100% design speed (9167 rpm). For the purposes of this demonstration, an inviscid calculation using 3 levels of multigrid has been configured.

The next step in the solution process is to simply run the *ADPAC-AOACR* program for this case. The standard input file **nasa.input** and the boundary data file **nasa.boundata** are provided to run the program (these files are listed in this manual as sample files in Sections 3.6 and 3.7). The steady flow solution is generated by issuing the command

```
../../src/adpac/adpac <nasa.input >nasa.output
```

The computation time required to generate the steady state solution may take up to four hours on a workstation-class computer. Once the steady flow solution has been generated, the ls command will reveal the following files:

```
nasa.restart.new    nasa.p3drel     nasa.p3dabs
nasa.converge       nasa.input      nasa.output
nasa.converge.save  nasa.output.save
```

The file *nasa.restart.new* contains the restart file necessary to continue this run from the point of termination. The files *nasa.p3dabs* and *nasa.p3drel* contain the absolute and relative flow *PLOT3D* flow variable information, respectively. The file

NASA 1.15 Pressure Ratio Fan Test Case Description

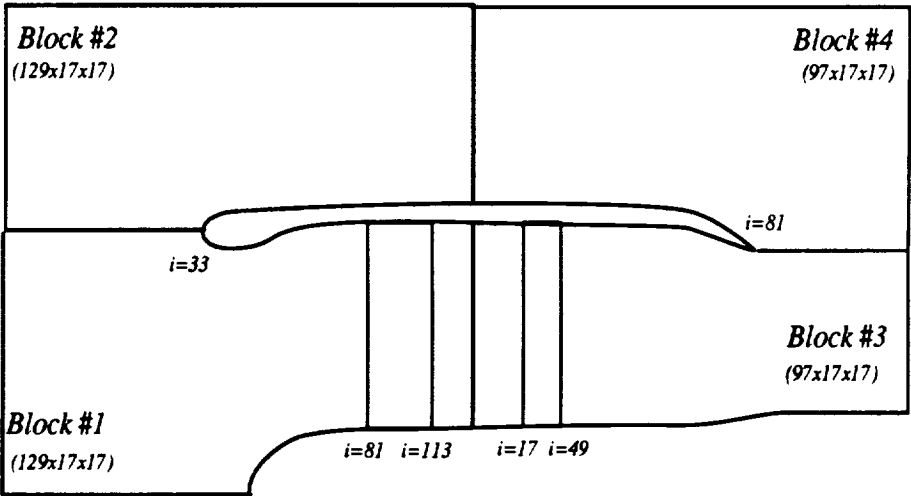## Axisymmetric Mesh View
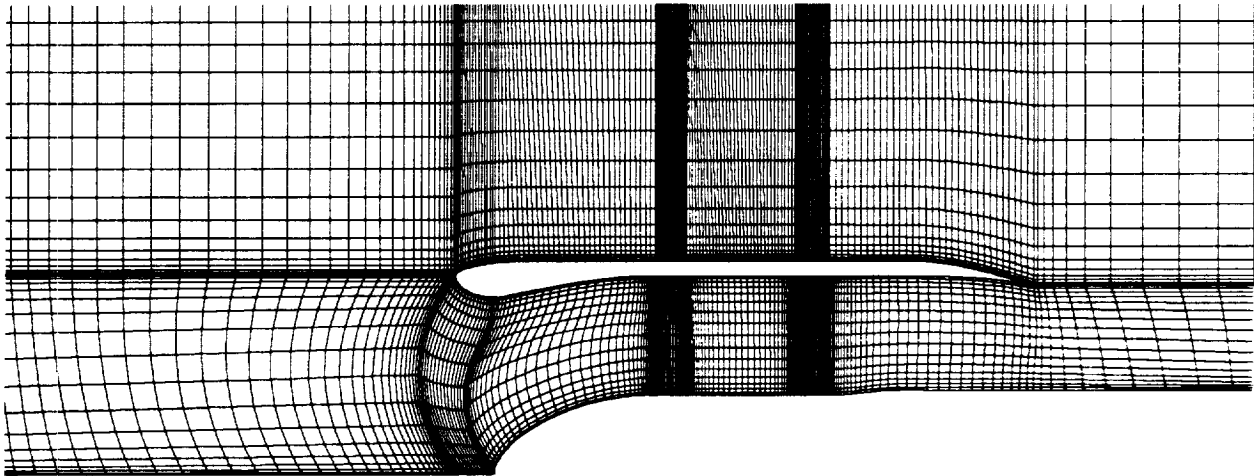
Mesh Block Structure

Figure A.1: NASA 1.15 Pressure Ratio Fan Test Case

nasa.output is the new standard output file, and should be compared with the file *nasa.output.save* to verify that the program has performed the calculation correctly. It may be of interest to examine these steady flow results with *PLOT3D* at this point (see Ref. [11] for details).

A plot of the convergence history for this case is given in Figure A.2. The "jumps" in the residual history are a result of the "full" multigrid startup procedure, and should not be considered inappropriate.

The standard output file **nasa.output** should be compared with the listing provided in Section 3.10 to make sure that the code has performed the calculation properly.
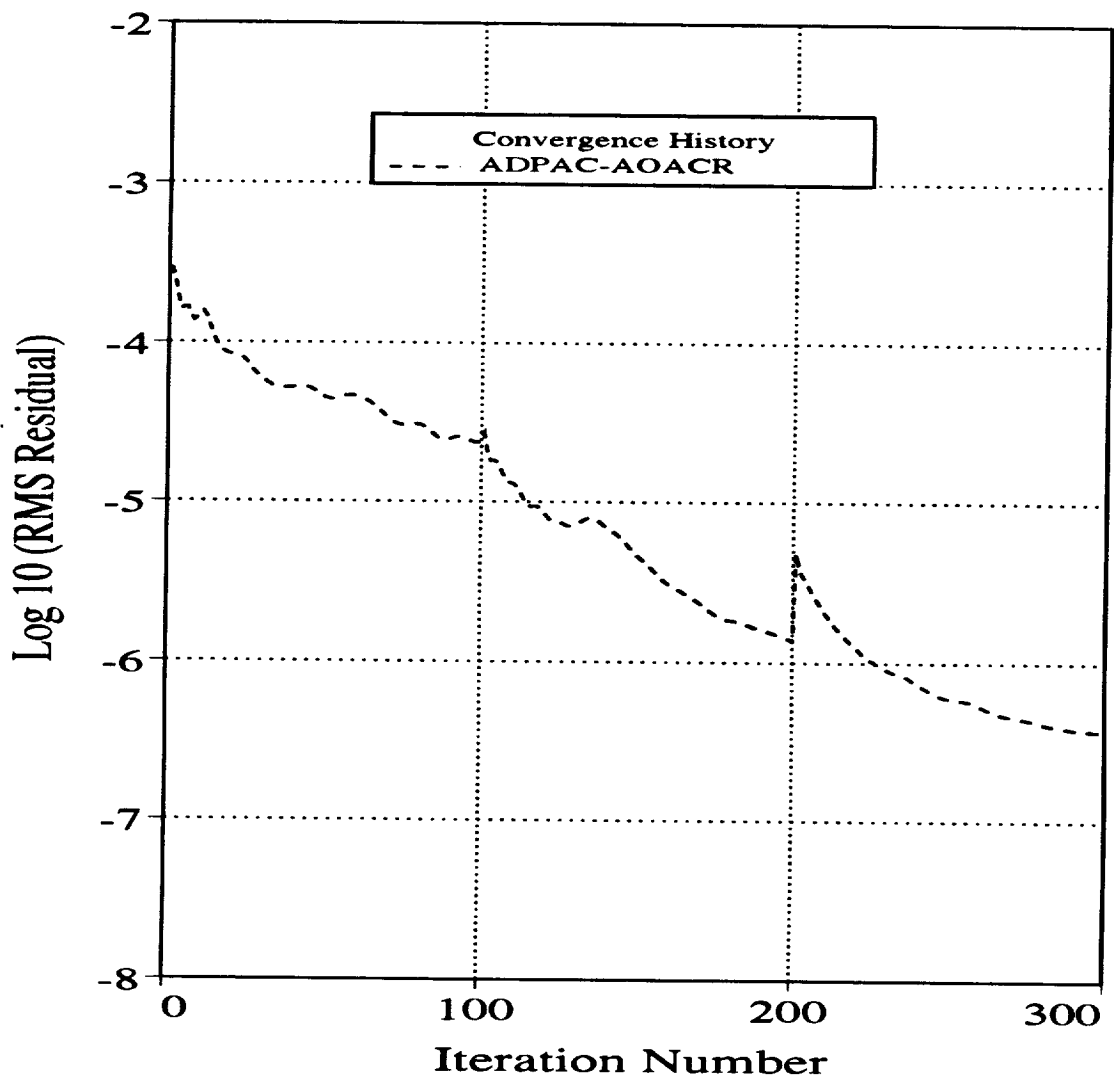
C - 3

Figure A.2: *ADPAC-AOACR* Convergence History for NASA 1.15 Pressure Ratio Fan Test Case

APPENDIX B. *ADPAC-AOACR* INPUT FILE QUICK REFERENCE

This appendix briefly lists the available input parameters for the *ADPAC-AOACR* code. A complete description of each variable is given in Section 3.6.

All keyword input lines are given in the following format:

```
KEYWORD = Value      Comment String
```

where KEYWORD is one of the recognized keywords described below, and Value is the specific value to be assigned to that variable. The input line must contain the equals sign (=) with one or more blanks on both sides in order to be recognized. The Comment String must be separated by one or more blank spaces from the Value. Therefore, the lines

```
DIAM    = 10.000
DIAM                              =                    10.000
DIAM    = 10.000    This is the diameter.
```

are valid keyword input lines assigning the value 10.0 to the variable associated with the keyword DIAM. Conversely, the lines

```
DIAM= 10.000
DIAM =10.000
DIAM=10.000
```

are not recognizable keyword input lines (in spite of the presence of the keyword DIAM), because of the lack of proper placement of the blanks about the equals sign. All keyword values are either real numbers (which, in many cases, are converted to integers in the code) or character strings.

It is unnecessary to specify all possible keywords in every input file. The *ADPAC-AOACR* code is programmed with a default set of input variables such that the only

191

input variable which must be present is the **CASENAME** (described below) which is used to assign input/output file names. A list and description of all input keywords and their default values are listed below.

ADPAC-AOACR Standard Input File Keyword Description

KEYWORD	PAGE	DESCRIPTION
CASENAME	p. 33	Case name used to identify files
RMACH	p. 33	Reference Mach Number
FINVVI	p. 34	Viscous(=1.0)/Inviscid(=0.0) solution trigger
GAMMA	p. 35	Specific heat ratio
PREF	p. 35	Reference pressure (pounds force/ft**2)
TREF	p. 36	Reference temperature (degrees Rankine)
RGAS	p. 36	Gas constant (foot-pounds force per slug-degree Rankine)
DIAM	p. 37	Reference diameter (feet)
EPSX	p. 37	i coordinate implicit residual smoothing coefficients
EPSY	p. 37	j coordinate implicit residual smoothing coefficients
EPSZ	p. 37	k coordinate implicit residual smoothing coefficients
VIS2	p. 38	Second order damping coefficient
VIS4	p. 38	Fourth order damping coefficient
CFL	p. 39	Calculation CFL number
FNCMAX	p. 39	Number of fine mesh iterations
FMULTI	p. 42	Number of multigrid mesh levels
FFULMG	p. 42	Full multigrid initialization trigger
FTIMEI	p. 42	Iteration interval between time step updates
FTURBI	p. 43	Iteration interval between turbulence model updates
FTURBB	p. 43	Iteration number at which turbulence model is activated
PRNO	p. 44	Gas Prandtl Number
PRTNO	p. 44	Turbulent Prandtl Number
FSOLVE	p. 44	Solution scheme trigger (0.0, 1.0, 2.0)
FFILT	p. 45	Solution damping trigger
FRESID	p. 45	Solution residual smoothing trigger
FREST	p. 45	Solution restart trigger
FUNINT	p. 46	Iteration interval between instantaneous *PLOT3D* file output
FSUBIT	p. 47	Number of subiterations during multigrid coarse mesh cycles

APPENDIX C. *ADPAC-AOACR* BOUNDARY DATA FILE QUICK REFERENCE

This appendix briefly lists the available boundary data specification parameters for the *ADPAC-AOACR* code. A complete description of each variable is given in Section 3.7.

The *ADPAC-AOACR* boundary data file contains the user-specifiable parameters which control the application of boundary conditions on the multiple-block mesh during a time-marching solution. These boundary specifications determine the location of solid walls, input/output flow regions, and block-to-block communication paths.

During code execution, the boundary data file is read one line at a time as a character string, and each string is parsed sequentially to determine the specific program action in each case. The boundary data file utilizes a keyword input format, such that any line which does not contain a recognizable keyword is treated as a comment line. Therefore, the user may place any number of comments in the file (so long as the line does not contain a keyword input string in the form described below), and code execution is unaltered. A line in the boundary data file can also be effectively commented by inserting a # character in the first column. Therefore the lines

```
PATCH   1  1  K  K  P  M  I  J  1  17   1 129  1  17   1 129  1  17
PATCH   2  2  K  K  P  M  I  J  1  17   1 129  1  17   1 129  1  17
```

are acceptable boundary specifications; however, the lines

```
#PATCH  1  1  K  K  P  M  I  J  1  17   1 129  1  17   1 129  1  17
#PATCH  2  2  K  K  P  M  I  J  1  17   1 129  1  17   1 129  1  17
```

194

would be neglected.

Some boundary condition specifications require additional data beyond that incorporated in the boundary specification line. In these cases, described in detail for the specific boundary types later in this Section, the additional data is included immediately after the boundary specification line.

A list and brief description of all valid boundary data keywords and any additional data required for the given boundary condition is now presented below.

ADPAC-AOACR Boundary Data File Keyword Description

KEYWORD	PAGE	DESCRIPTION
ENDDATA	p. 75	Flag indicating the end of the boundary conditions
SSVI	p. 76	Solid surface, 3-D grid block, viscous
SS2DVI	p. 77	Solid surface, 2-D grid block, viscous
SSIN	p. 77	Solid surface, 3-D grid block, inviscid
SS2DIN	p. 78	Solid surface, 2-D grid block, inviscid
INLETG	p. 78	Inlet, 3-D grid block, generic
INL2DG	p. 78	Inlet, 2-D grid block, generic
INLETA	p. 79	Inlet, 3-D grid block, angle of attack
INL2DA	p. 81	Inlet, 2-D grid block, angle of attack
INLETT	p. 81	Inlet, 3-D grid block, turbomachinery
INL2DT	p. 83	Inlet, 2-D grid block, turbomachinery
EXITG	p. 83	Exit, 3-D grid block, generic
EXT2DG	p. 85	Exit, 2-D grid block, generic
EXITT	p. 86	Exit, 3-D grid block, turbomachinery
EXT2DT	p. 87	Exit, 2-D grid block, turbomachinery
EXITP	p. 88	Exit, 3-D grid block, patched
EXT2DP	p. 89	Exit, 2-D grid block, patched
FREE	p. 91	Far field, 3-D grid block
FRE2D	p. 91	Far field, 2-D grid block
KILL	p. 92	Solution kill specification, 3-D grid block
KIL2D	p. 93	Solution kill specification, 2-D grid block
PATCH	p. 95	Contiguous mesh communication patch, (3-D or 2-D)
PINT	p. 97	Non-contiguous mesh communication patch (3-D or 2-D)

APPENDIX D. *ADPAC* DISTRIBUTION LIST

ADPAC DISTRIBUTION LIST
NASA Contract NAS3-25270
Task Order #5
User's Manual CR-187125
Final Report CR-187126

GOVERNMENT AGENCIES:
NASA Headquarters
600 Independence Avenue, SW
Washington, DC 20546

      Attn: RJ/R. Whitehead
              RP/N. Nijahawan

NASA Lewis Research Center
21000 Brookpark Road
Cleveland, OH 44135

Attn:			
	J. J. Adamczyk	M.S. 5-9	(2 copies)
	C. L. Ball	M.S. 86-1	
	L. J. Bober	M.S. 77-6	
	D. R. Boldman	M.S. 86-7	
	B. Clark	M.S. 77-6	
	R. W. Claus	M.S. 142-5	
	J. H. Dittmar	M.S. 77-6	
	D. M. Fricker	M.S. 5-11	
	J. F. Groeneweg	M.S. 77-6	(4 copies)
	C. E. Hughes	M.S. 77-6	
	R. J. Jeracki	M.S. 77-6	
	C. M. Kim	M.S. 77-6	

199

198

L. M. Larosillieree	M.S. 77-6	
A. P. Kurkov	M.S. 23-3	
J. Lytle	M.S. AAC-1	
A. Mahajan	M.S. 23-3	
C. J. Miller	M.S. 77-6	(10 copies)
D. P. Miller	M.S. 77-6	
R. D. Moore	M.S. 77-6	
D. V. Murthy	M.S. 23-3	
F. E. Newman	M.S. 77-6	
L. D. Nichols	M.S. 142-5	
C. W. Putt	M.S. 142-2	(2 copies)
D. R. Reddy	M.S. 5-11	
T. S. Reddy	M.S. 23-3	
R. Srivastava	M.S. 23-3	
G. L. Stefko	M.S. 23-3	
R. P. Woodward	M.S. 77-6	
J. A. Ziemianski	M.S. 86-1	
Report Control Office	M.S. 60-1	(4 copies)
Tech. Utilization Office	M.S. 7-3	
AFSC Liaison Office	M.S. 501-3	

NASA Ames Research Center
Moffett Field, CA 94035

    Attn: Library   M.S. 202-3

NASA Langley Research Center
Hampton, VA 23681-0001

Attn:	F. Farassat	M.S. 460
	S. L. Hodge	M.S. 460
	J. Posey	M.S. 460
	D. Stephens	M.S. 462
	Library	M.S. 185

NASA Marshall Space Flight Center
Marshall Space Flight Center, AL 35812

    Attn: P. McConnaughey   ED32

NASA Scientific and Technical Information Facility
P.O. Box 8757
BWI Airport, MD 21240

     Attn: Accession Dept. (6 copies and the FF427 form)

INDUSTRIES
Aerojet TechSystems Company
Aerojet Propulsion Division
P.O. Box 13222
Sacramento, CA 95813

     Attn: G. E. Bache'   Dept. 9934

AIResearch Los Angeles Div
Allied-Signal Aerospace Co.
P.O. Box 2960
Torrance, CA 90509-2960

     Attn: T. Booth   Dept 93-209, T-42

Allison Gas Turbine Division, GMC Corp.
P.O. Box 420
Indianapolis, IN 46206-0420

Attn:	P. C. Tramm	Mail Stop T-144
	R. F. Alverson	Mail Stop T-14A
	D. W. Burns	Mail Stop T-14A
	A. J. Crook	Mail Stop T-14A
	R. A. Delaney	Mail Stop T-14A
	E. J. Hall	Mail Stop T-14A
	D. J. Helton	Mail Stop T-14A
	J. R. Rathman	Mail Stop T-14A
	D. A. Topp	Mail Stop T-14A

Boeing Commercial Airplane Company (BCAC)
P.O. Box 3707
Seattle, WA 98124-2207

Attn:	B. W. Farquhar	M.S. 6M-98
	W-H. Jou	M.S. 7H-96
	R. Cuthbertson	M.S. 79-84
	G. Sengupta	M.S. 7H-91

Douglas Aircraft Company Division
McDonnell Douglas Corporation
3855 Lakewood Blvd.
Long Beach, CA 90846

Attn: M. Joshi   M.S. 36-60
       F. Lynch   M.S. 36-60
       G. Page   M.S. 35-86
       W. Siegele   M.S. 202-15

Garrett Engine Division
Allied-Signal Aerospace Co.
P.O. Box 5217
Phoenix, AZ 85010

Attn: M. Jones
       W. Waterman

General Dynamics Convair
P.O. Box 80844
San Diego, CA 92138

Attn: B. Bergman   M.Z. 36-1240
       K. Taylor   M.Z. 54-6890

General Electric Company
Aircraft Engine Group
1 Neumann Way
Evendale, OH 45215

Attn: P. Gliebe   Mail Drop A-304
       C. Lenhardt   Mail Drop A-330
       M. Majjigi   Mail Drop A-319
       M. Pearson   Mail Drop A-317
       L. Smith   Mail Drop H-4
       C. Whitfield   Mail Drop A-304
       Y. M. Chen   Mail Drop A-323
       I Jennions   Mail Drop A-323

Hamilton Standard Division - UTC
Windsor Locks, CT 06096

Attn: F. B. Metzger   M.S. 1A-3-6

Jason Assoc @ DARPA SUBTECH Center
David Taylor NSRDC
Bethesda, MD 20084

        Attn: C. Knight   Bldg 17E, Rm 120

Lockheed-California Company
P.O. Box 551
Burbank, CA 91520

        Attn: Library

Lockheed Engineering & Sciences Company
144 Research Drive
Hampton, VA 23666

        Attn: M. H. Dunn   M.S. 904
                P. L. Spence   M.S. 904

Mcdonnell Douglas Helicopter Co.
5000 East Mcdowell Road
Mesa, AZ 85205-9797

        Attn: H. Tadghighi

Pratt & Whitney Aircraft - UTC
Commercial Products Division
400 Main Street
East Hartford, CT 06108

        Attn: D. B. Hanson   M.S. 165-11
                D. Hopwood   M.S. 162-07   (3 copies)
                D. Mathews   M.S. 165-11
                W. Lord   M.S. 169-23
                S. Tanrikut   M.S. 163-01
                T. Wynosky   M.S. 169-23

Pratt & Whitney Aircraft - UTC
Government Engine Business
P. O. Box 109600
West Palm Beach, FL 33410

Attn: F. Huber   M.S. 715-92

Rockwell International Corp.
Rocketdyne Division
6633 Canoga Avenue
Canoga Park, CA 91303

Attn: M. Sindir   WC77

Rohr Industries, Inc.
P.O. Box 878
Chula Vista, CA 92012-0878

Attn: Library

Sundstrand Corp.
P.O. Box 85757
San Diego, CA 92186-5757

Attn: C. Rodgers

Sverdrup Technology, Inc.
2001 Aerospace Parkway
Brookpark, OH 44142

Attn: B. Berkowitz   SVR-UNASS
       R. M. Nallasamy   SVRUNASS
       P. Tsai   SVR-UNASS
       O. Yamamoto   SVR-UNASS

Teledyne CAE
1330 Laskey Road
P.O. Box 6971
Toledo, OH 43612

Attn: E. Benstein

TEXTRON Lycoming
Stratford Divison
550 Main Street
Stratford, CT 06497-7593

Attn: A. Sehra   M.S. LSD10
         M. G. Zedan   M.S. LSD10

United Technologies Corporation Research Center
Silver Lane
E. Hartford, CT 06108

Attn: M. Barnett   M.S. 20
         R. Davis   M.S. 20
         D. Dorney   M.S. 20

Williams International
2280 W. Maple Road
P.O Box 200
Walled Lake, MI 48088

Attn: R. Pampreen

Wright Research and Development Center
Dayton, OH 45433

Attn: W. Troha   Mail Code POTC
         M. Stibich   Mail Code POTC

UNIVERSITIES

Georgia Institute of Technology
School of Aerospace Engineering
Atlanta, GA 30332-0800

Attn: Dr. L. N. Sankar

Mississippi State University
Department of Aerospace Engineering
P.O. Box A
Mississippi State, MS 39762

Attn: Dr. D. Whitfield
         Dr. M. Janus

Ohio State University
Department of Aeronautical
 and Astronautical Engineering
Columbus, OH 43220

Attn: Prof. G. M. Gregorek

Pennsylvania State University
Department of Aerospace Engineering
233 Hammond Building
University Park, PA 16802

Attn: Dr. B. Lakshminarayana

Purdue University
School of Aeronautics and Astronautics
W. Lafayette, IN 47907

Attn: Dr. J. P. Sullivan
        Dr. M. Williams

Purdue University
School of Mechanical Engineering
W. Lafayette, IN 47907

Attn: Dr. S. Fleeter

Texas A&M University
Aerospace Engineering Department
College Station, TX 77843-3141

Attn: Dr. K. D. Korkan

University of Arizona
Aerospace and Mechanical Engineering Department
Aero Building #16
Tucson, AZ 85721

Attn: Dr. E. J. Kerschen

University of Iowa
Institute of Hydraulic Research
Iowa City, IA 52242

Attn: F. Stern

University of Missouri-Rolla
Mechanical Engineering Department
Rolla, MO 65401-0249

Attn: Dr. Walter Eversman

University of Notre Dame
Aerospace & Mechanical Engineering
Fitzpatrick Hall of Engineering
Notre Dame, IN 46556

Attn: Dr. H. M. Atassi

Report Documentation Page

NASA
National Aeronautics and
Space Administration

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR 187125		

4. Title and Subtitle	5. Report Date
Investigation of Advanced Counterrotating Blade Configuration Concepts for High Speed Turboprop Systems, Task 5-Unsteady Counterrotation Ducted Propfan Analysis, Computer Program User's Manual	January 1993
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Edward J. Hall and Robert A. Delaney	
	10. Work Unit No.
	535-03-10

9. Performing Organization Name and Address	11. Contract or Grant No.
Allison Gas Turbine - Division General Motors Corporation P.O. Box 420, Speed Code T14A Indianapolis, IN 46206-0420	NAS3-25270
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	Contractor Report, Final
National Aeronautics and Space Administration Lewis Research Center 21000 Brookpark Road Cleveland, OH 44135-3191	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared in cooperation with NASA project Manager, Christopher J. Miller, NASA Lewis Research Center, Cleveland, Ohio

16. Abstract The primary objective of this study was the development of a time-marching three-dimensional Euler/Navier-Stokes aerodyanamic analysis to predict steady and unsteady compressible transonic flows about ducted and unducted propfan propulsion systems employing multiple blade rows. The computer codes resulting from this study are referred to as ADPAC-AOACR (Advanced Ducted Propfan Analysis Codes-Angle of Attack Coupled Row).

This report is intended to serve as a computer program user's manual for the ADPAC-AOACR codes developed under Task 5 of NASA Contract NAS3-25270, Unsteady Counterrotating Ducted Propfan Analysis.

The ADPAC-AOACR program is based on a flexible multiple blocked grid discretization scheme permitting coupled 2-D/3-D mesh block solutions with application to a wide variety of geometries. For convenience, several standard mesh block structures are described for turbomachinery applications. Aerodynamic calculations are based on a four-stage Runge-Kutta time-marching finite volume solution technique with added numerical dissiplation. Steady flow predictions are accelerated by a multigrid procedure. Numerical calculations are compared with experimental data for several test cases to demonstrate the utility of this approach for predicting the aerodynamics of modern turbomachinery configurations employing multiple Blade rows

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Navier-Stokes Turbomachinery Fan Multiple Block Ducted ADPAC	Unclassified, - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified		A06

NASA FORM 1626 OCT 86