NASA/CR—1999-206604

TADS—A CFD-Based Turbomachinery Analysis and Design System With GUI
Version 2.0—User's Manual

M.J. Koiro, R.A. Myers, and R.A. Delaney
Allison Engine Division of Pratt & Whitney, Indianapolis, Indiana

Prepared under Contract NAS3–27350, Task 10

National Aeronautics and
Space Administration

Glenn Research Center

May 1999

Contents

List of Figures

List of Tables

Chapter 1

Summary

The primary objective of this study was the development of a CFD (Computational Fluid Dynamics) based turbomachinery airfoil analysis and design system, controlled by a GUI (Graphical User Interface). The computer codes resulting from this effort are referred to as *TADS* (Turbomachinery Analysis and Design System). This document is intended to serve as a User's Manual for the computer programs which comprise the *TADS* system, developed under Task 18 of NASA Contract NAS3-25950, *ADPAC* System Coupling to Blade Analysis & Design System GUI and Task 10 of NASA Contract NAS3-27394, *ADPAC* System Coupling to Blade Analysis & Design System GUI, Phase II - Loss, Design, and Multi-stage Analysis.

TADS couples a throughflow solver (*ADPAC*) with a quasi-3D blade-to-blade solver (*RVCQ3D*) in an interactive package. Throughflow analysis and design capability was developed in *ADPAC* through the addition of blade force and blockage terms to the governing equations. A GUI was developed to simplify user input and automate the many tasks required to perform turbomachinery analysis and design. The coupling of the various programs was done in such a way that alternative solvers or grid generators could be easily incorporated into the *TADS* framework. Results of aerodynamic calculations using the *TADS* system are presented for a highly loaded fan, a compressor stator, a low speed turbine blade and a transonic turbine vane.

Chapter 2

Introduction

Traditionally, airfoils have been designed by stacking 2-D sections to create a 3-D model. While 3-D analysis has become common, 3-D design has not. Today, pseudo 3-D design is accomplished by adjusting 2-D parameters in response to 3-D analysis. This approach benefits from a large experience base in 2-D design and a good understanding of the 2-D design parameters.

There are CFD codes available to perform the full 3-D analysis of the complicated flows associated with 3-D airfoils, but they are slow and require large amounts of computer memory. While advances in computer technology and in solution algorithms are reducing the penalties associated with 3-D modeling, routine design is still not practical with these tools.

The objective of this program is to produce a turbomachinery airfoil design and analysis package built on the traditional approach, but using modern analytical techniques. This new Turbomachinery Analysis and Design System (*TADS*) couples a throughflow solver with a quasi-3D blade-to-blade solver in an interactive package. The coupling is done in such a way that alternative solvers or grid generators can be easily incorporated into the *TADS* framework.

TADS is a an interactive turbomachinery design system which provides a user-friendly means of managing the jobs and files associated with a coupled throughflow/blade-to-blade analysis. It is controlled by a Graphical User Interface (GUI), which simplifies user input and automates the many required tasks. The coupled analysis encompasses the following design activities:

1. axisymmetric grid generation
2. through-flow calculation
3. airfoil slicing
4. blade-to-blade grid generation
5. blade-to-blade calculation
6. streamline resolution
7. body force resolution

A coupled throughflow and blade-to-blade analysis requires many steps, repeated iteratively. Figure 2.1 shows the work flow of a typical analysis. A converged analysis is achieved when the meridional streamlines are settled in the throughflow analysis and the mean stream surface is settled in the blade-to-blade analysis. Each analysis provides the solution surface for the other, and iteration is required to determine the final shapes. In practice, only one iteration is required to achieve an acceptable solution in many analysis cases. When the design mode is invoked, the shape of the blade is unknown, so arriving at a final blade geometry may take several iterations. *TADS* is flexible enough that the user can run alternating analysis and design mode iterations to converge on the desired blade shape.

TADS is composed of independent programs which are able to interact via a controlling program. A graphical user interface (GUI) serves as the control program and the user's point of contact with the program modules. The program modules are computational codes and their associated pre- and post-processors. This type of scheme allows modules to be added, deleted or modified with little or no effect on the controlling program. It also allows modification of the controlling program independent of the program modules. In order to leave each code as a stand-alone module the I/O routines were modified to conform to a common standard. The disadvantage to this approach is the many files created during an analysis clutter the directory. Although the clutter is unfortunate, these files provide a built-in restart capability for the analysis.

These program modules are grouped into seven functional divisions referred to as component modules. Table 2.1 may give a better understanding of this organization. As shown in the table, the component modules correspond to the seven design activities listed above. For example, *ADPAC* is a program module for the "Throughflow Calculation" component group.

Also, *TADS* shares data between component modules. This insures the data integrity of the individual component solutions. In addition to facilitating data I/O, *TADS* acts as a file manager/bookkeeper by utilizing a file naming convention which forces a consistent file naming strategy.

The graphical user interface allows the user to interactively query, alter and submit a design analysis in an organized and compact environment. The layout of the GUI helps to guide the user through the design process, while maintaining the flexibility required by an interactive system. The GUI was programmed in X Windows for portability. It also provides the means for executing component modules on remote machines in order to take advantage of heterogeneous system hardware and resources.

Visually, the GUI consists of a series of windows or panels:

- Main panel

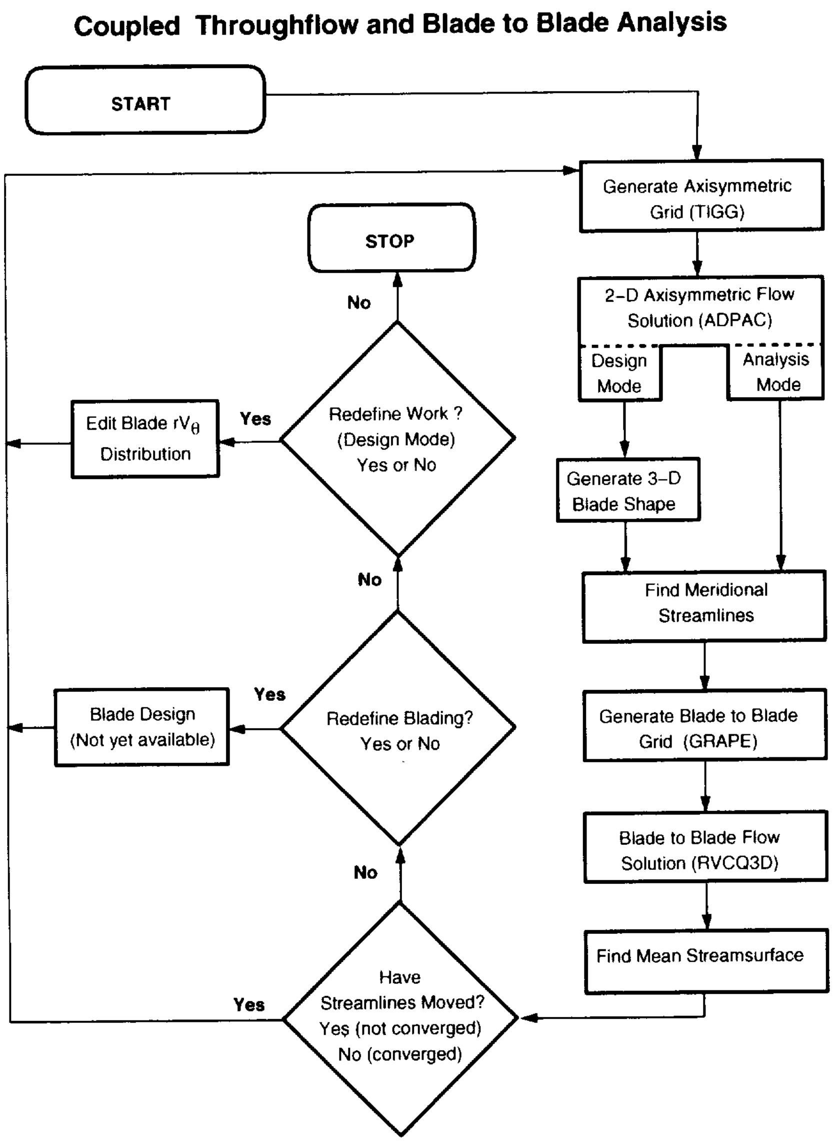

Coupled Throughflow and Blade to Blade Analysis
START
Generate Axisymmetric Grid (TIGG)
STOP
No
2-D Axisymmetric Flow Solution (ADPAC)
Design Mode
Analysis Mode
Edit Blade rV_θ Distribution
Yes
Redefine Work ? (Design Mode) Yes or No
Generate 3-D Blade Shape
No
Find Meridional Streamlines
Blade Design (Not yet available)
Yes
Redefine Blading? Yes or No
Generate Blade to Blade Grid (GRAPE)
No
Blade to Blade Flow Solution (RVCQ3D)
Find Mean Streamsurface
Yes
Have Streamlines Moved? Yes (not converged) No (converged)

Figure 2.1: The coupled throughflow and blade-to-blade analysis is an iterative, multi-step process.

Table 2.1: Coupled analysis organization.

Component Module	*Program Module*	*Executable(s)*		
Axisymmetric Grid Generation	*TIGG*	intigg	tiggc3d	
	BATCH TIGG	intigg	tiggc3d	
Throughflow Calculation	*ADPAC* (analysis)	adpacbc	bodyf	adpac
	VIADAC	n/a		
Slicer	*SLICER*	radsl	slicer	
Blade-to-Blade Grid Generation	*GRAPE*	grape		
Blade-to-Blade Calculation	*RVCQ3D*	rvcq3d		
	B2BADPAC	adpac		
	PGASC	n/a		
	TSONIC	n/a		
Mean Streamline Finder	*MEANSL*	restack	meansl	

- Message panel
- Remote processor setup panel
- Standardized data input panels
    - Slice independent panels
    - Slice dependent panels
- Specialized data input panels
- Stand-alone pre- and post-processor units

Chapter 3

Conventions and Nomenclature

This chapter explains some of the nomenclature and conventions used throughout this manual and the GUI.

3.1 Typographic Conventions

- Items that are selectable are set in **bold** type.
- Path and file names are in `fixed-width type`.
- Program module names are *EMPHASIZED UPPERCASE.*
- Executable and shell script names are *emphasized.*
- Environment variables are in `fixed-width type`.
- UNIX commands are set in **bold** type.
- Optional parameters are set in brackets, [ ].
- Key presses are enclosed in <>, e.g. < *RETURN* >.
- Keyboard entries are in `fixed-width type` and generally terminated with < *RETURN* > or < *ENTER* > unless otherwise stated.

3.2 Nomenclature

Definitions of terms, acronyms and abbreviations used in this document are provided below.

panel	Window
control	Widget (toggle button, text entry box, push-button etc...)
"Local"	Machine on which *TADS* is executed
`casename`	User supplied case name
ASCII	American Standard Code for Information Interchange
HTML	HyperText Markup Language
PLOT3D	NASA graphics flow visualization program
TADS	Turbomachinery Analysis and Design System

3.3 GUI Conventions

Specific controls have individual behaviors and appearances. Unless otherwise stated, the following conventions are used by the GUI:

3.3.1 Windows

Closing a window from the decorations box (usually in the top left corner for X-windows systems) will close the parent window and force *TADS* to terminate (non-gracefully). Resizing a window will automatically rescale the contents to fill the new window. Pop-up windows are secondary windows that appear off a main window without closing the main window. They are dependent on their main window such that if the main is closed or icon'ed, the pop-up is closed or icon'ed as well.

3.3.2 Mouse Buttons

Unless explicitly stated otherwise, the term "click" means to move the mouse until the cursor is over the desired control, then pressing and releasing the left mouse button. "Double clicking" is simply performing two "clicks" in rapid succession.

At this time, the right and middle mouse buttons have no function.

3.3.3 Pulldown Lists

A downward pointing arrowhead indicates a pulldown list. The list is displayed by clicking on the arrowhead with the left mouse button. To make a selection from the list, click on the desired item.

3.3.4 Toggle Buttons

To activate a toggle button move the mouse until the cursor is over the button and click the left mouse button. A filled toggle box is considered to be on, and an empty box is off.

3.3.5 Radio Buttons

Radio buttons are a group of toggle buttons which are mutually exclusive of each other. In other words, only one may be on at any time; however, one must always be on. When one button is clicked on, all the others are forced to off.

3.3.6 Push-buttons

A push-button is activated by clicking on it with the left mouse button.

3.3.7 Text Boxes

To enter data in a text box, the box must first be selected by clicking on it with the left mouse button (this will cause the text box to be highlighted). Then text may be typed from the keyboard. Double clicking on the text will display any existing text in reverse video mode. This text will be overwritten if any text is input by the user. Most text boxes encountered in this program will accept the data after pressing <ENTER> or <TAB>, or after clicking on another control (including one which exits the current panel).

3.3.8 Action Buttons

Action buttons are push buttons which control file creation, modification and/or execution. Action buttons are found throughout *TADS*, however, most of them are located at the bottom of input panels.

3.4 File Formats

All files used by *TADS* are ASCII text, native binary or SDB binary. SDB is a library of I/O routines which create platform independent binary data as opposed to native binary which is platform dependent. Each supported platform has a SDB library available to perform the necessary conversions. Using SDB, any platform can read binary data created by any other platform. Supported platforms include Cray, Silicon Graphics, IBM RS/6000, Sun, etc. The binary data structure of SDB is equivalent to reading and writing binary data in C on a Silicon Graphics workstation. SDB is documented in [11]. All *TADS* files, except native binary files, are platform independent, so any program task can be performed on any supported machine without loss of generality.

The only files using the native binary format are the database files. These files are not required for restart if the corresponding ASCII files exists (which they normally should if the database file exists). If both types of files exist, the information from the native binary files is over-ridden by data in the ASCII files. The native binary files should be removed before executing subsequent runs of *TADS* on a "local" machine which is of a different platform type than the original run.

Most of the binary files used by *TADS* are geometry or flow data files. All geometry or flow data files are written in *PLOT3D* format using SDB. Specifically, they are 3-D, whole, multiple grid files, in accordance with the definitions in [10], pp 162-165. The only exceptions being the 2-D, single grid files used for the blade-to-blade analyses.

3.5 File Naming

The files created or used by *TADS* use the `casename.extension` file name convention adopted from *ADPAC*. The user specifies a case name for the problem, and each file needed by *TADS* is assigned a unique extension. This way, multiple airfoils could be run in the same directory. There is also much less confusion about which files were created by *TADS*. Some programs, notably the grid generators and quasi 3-D solvers expect files with specific names for input and output. These files do not follow the convention adopted for *TADS*. This is not a serious problem unless multiple runs of the same program must be made in the same directory. Multiple runs would require multiple files with the same name, resulting in overwritten data or confusion about the contents of files. While it would be possible to write scripts to rename or symbolically link files to the expected names, it is clearer and simpler to create subdirectories to contain these files. *TADS* creates a subdirectory for each blade row to be analyzed. Within these row directories, another set of subdirectories is created for each blade-to-blade section to be analyzed. Within these subdirectories, some files do not conform to the naming convention, but confusion is avoided because the subdirectories themselves are named descriptively.

A complete list of input and output file names and descriptions can be found in Appendix A.

Chapter 4

Preparing Input for TADS

The *TADS* system requires different files for input depending on whether the user is running the analysis or design mode. Extra files are also necessary for running the throughflow loss model. In the analysis mode, four things are needed as input: a casename, a Cartesian description of the airfoil, a description of the meridional flowpath, and aerodynamic data. For the design mode, five things are needed as input: a casename, a description of the meridional flowpath, aerodynamic data, a body force file, and a rV_θ file. The minimum file set required to start either the analysis or design mode of *TADS* is listed in Table 4.1: All other information needed by *TADS* has either a default value which can be reset in an input panel, or is generated internally by another part of the analysis.

4.1 Airfoil Description

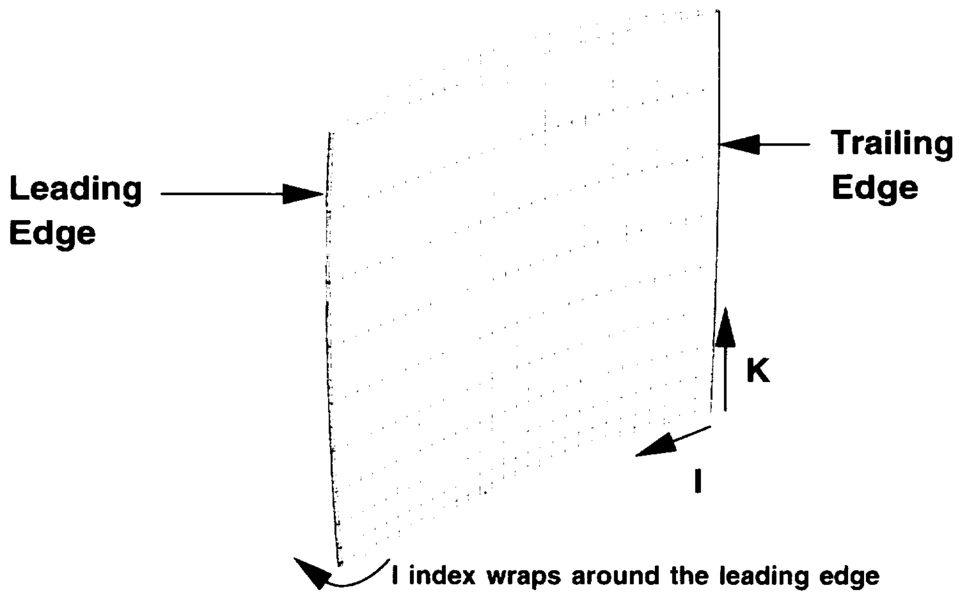
The airfoil is input as a 3-D Cartesian surface in two parameters. This surface can be envisioned as the first contour in an O-grid, Figure 4.1. The first parameter, I, wraps clockwise from the trailing edge around the airfoil to form a closed surface, when viewed from above. The starting point of the I indices doesn't matter, only the wrapping **direction**. When choosing I values, the user must be sure to set the proper tangency points in the `casename.tdsaro` file (see 4.3). The J index is set to 1, corresponding to the first contour in a right-handed O-grid. The K index is the number of spanwise point in the airfoil description. *TADS* expects to receive the airfoil description with the machine axis aligned with the X direction. The file is a 3-D, whole, multiple grid, binary file, in accordance with the definitions in [10], pp 162-165. The file is written in *PLOT3D* format using SDB, and is named `casename.tdsblad`, following the *TADS* convention. The coordinates should be in inches.

The distribution of points in the airfoil description should follow some basic guidelines. First, there must be sufficient resolution of all geometric features, specifically the leading edge and trailing edge. Second, the spanwise distribution of points must be smooth. The airfoil definition is sliced along the meridional streamlines for use in the blade-to-blade analysis. The shape of the airfoil is found along the streamline by splining each spanwise row of points and finding the intersection with the meridional streamline. If the spanwise point distribution is not smooth, the spline through those points will be less accurate, degrading the fidelity of the blade-to-blade analysis. The analogy between the airfoil definition and an O-grid is appropriate: the point distribution in the airfoil definition is acceptable if it would make an acceptable blade surface in a 3-D O-grid.

Table 4.1: Required input data files.

Name	*Mode*	*Format*	*Description*
casename.tdsblad	Analysis	*PLOT3D* (SDB)	airfoil geometry
casename.tdspath	Analysis, Design	ASCII	flowpath geometry
casename.tdsaro	Analysis, Design	ASCII	aerodynamic data
casename.bf.1	Design	*PLOT3D* (SDB)	*ADPAC* 2-D blockage/body force file for block #1
casename.rvtdesign	Design	ASCII	*PREDESIGN* input file
casename.rvt.1	Design	ASCII	*ADPAC* 2-D rV_θ file for block #1

Airfoil Description Input File *casename.tdsblad*



File is PLOT3D, 3D, whole, multiple grid, written using the SDB library

File contains surface description of the airfoil, wrapped clockwise from the trailing edge

The I index wraps around the airfoil from the trailing edge

The J index is 1 (constant)

The K index runs from hub to tip

Figure 4.1: The airfoil shape is defined as a surface in two parameters.

4.2 Flowpath Description

The meridional flowpath is defined by two lines in the (X, R) plane. The file is in ASCII free format, and is named **casename.tdspath**, according to the *TADS* convention. A sample input file is found in Appendix B.

This the format of this file is simple. The first line is a comment indicating that the hub surface definition follows. The second line contains the number of points in the hub surface definition. After this is the series of (X, R) coordinate pairs, with one pair on each line, in inches. The shroud definition follows the pattern of the hub definition. Immediately following the hub definition, is a comment line indicating that the shroud definition follows. Then comes the number of points in the shroud definition and the coordinate pairs as before.

The flowpath definition will be splined for use in many of the *TADS* modules. The definition should be resolved well enough that the spline accurately represents the surface. No particular placement of the points is required, and the number of points describing the hub and shroud is independent. The only restrictions on the flowpath definition are that there must not be any repeated points, and that the definition must be monotonic in the flow direction.

4.3 Aerodynamic Data

The aerodynamic data file contains tables of information at the airfoil leading and trailing edges. Following the *TADS* convention, the name of this file is **casename.tdsaro**. The file is in ASCII format and is read with free format by FORTRAN subroutines. A sample input file is found in Appendix C.

The **casename.tdsaro** file is largely self-documenting, with comment lines preceding each data entry. The contents of the comment lines are ignored, but there must be at least a blank line where each comment belongs.

Three comment lines precede the first item. The first item is a flag which indicates which type of machine is being analyzed (at present, the only acceptable value is 0 for axial machines). One comment line precedes the second item. The second item is the number of radial stations for which there is aerodynamic data. This number must correspond to the number of table entries which follow, but does not need to correspond to the number of spanwise points in the airfoil description (**casename.tdsblad**) or to the number of meridional streamlines to be used in the analysis.

Two comment lines precede the table of aerodynamic conditions at the leading edge. The table has two groups of data, with four entries per line. The first group consists of the radius, total pressure, total temperature, and the axial location. The radius and axial values define the locations at which the aerodynamic conditions are to be held, in inches. The locations of the aerodynamic data generally correspond to the leading edge (or trailing edge) of the airfoil. However, these definitions are never used to represent geometry, and are therefore somewhat arbitrary. The only restrictions are that there should be no repeated points, and the values should increase monotonically in the spanwise direction. The total pressure should be in pounds per square inch, and the total temperature should be expressed in degrees Rankine.

The second group is preceded by a single comment line, and consists of the radius, and the three Mach number components. The radius is repeated from above and is included for visual convenience. The Mach number components are the axial Mach number, the absolute circumferential Mach number and the radial Mach number.

The trailing edge table follows the leading edge table. Following the pattern of the leading edge table, there are two comment lines and then a group of four parameters: the radius, static pressure total temperature, and axial location. The radial and axial values define the locations at which the aerodynamic conditions are to be held. The static pressure should be expressed in pounds per square foot, and the total temperature should be expressed in degrees Rankine. Two comment lines also precede the second group, which consists of the radius, the axial Mach number, the absolute circumferential Mach number, and the radial Mach number. In the current release of *TADS* , the static pressure and the total temperature are not used. The Mach number components are used in the one-dimensional extrapolation routine which sets the pressure at the exit boundary in the throughflow analysis.

Following the trailing edge table are lines containing thermodynamic information and geometric properties. The ratio of specific heats (γ) and the gas constant follow two comment lines. The gas constant is expressed in the customary units of foot-pounds force per pound mass degree Rankine. Two more comment lines precede three geometric parameters: the wheel speed (in revolutions per minute), the tip clearance (in inches) and the number of blades.

Finally, two comment lines precede the airfoil tangency points. In traditional airfoil design programs, airfoils are defined in four segments: the pressure and suction surfaces, and the leading and trailing edges. The tangency points are those points in the airfoil description which denote where the leading and trailing edges join the pressure and suction surfaces. *TADS* uses these points when locating the mean camber line of the airfoil and when creating the blade-to-blade grid definitions. The mean camber line is determined from the pressure and suction surfaces. These surfaces are defined as the segments of the airfoil between the appropriate tangency points. Figure 4.2 shows the approximate location of the tangency points (and their appropriate abbreviations in the `casename.tdsaro` file) on an example airfoil. It is not required that these points actually define a joint between segments, but they should be chosen so that the pressure and suction surfaces don't contain the high curvature regions of the leading and trailing edges. If *TADS* issues messages indicating that it can't locate the mean camber line, adjust the tangency points away from the leading and trailing edges and rerun.

The tangency points are prescribed in clockwise order, starting at the leading edge. They are ordered as follows: the suction surface leading edge, the suction surface trailing edge, the pressure surface trailing edge, and the pressure surface leading edge. These values correspond to the I index in the airfoil definition.

4.4 Axisymmetric Body Force File

The body force file is a standard 2-D blockage/body force file for *ADPAC*. During a *TADS* iteration, it is created by the *bodyf* module. The five body force arrays are set to 0.0 and the blockage, λ is given as:

$$\lambda = \frac{\theta_{PS} - \theta_{SS}}{(2\pi/N)}$$

where N is the total number of blades in a given blade row.

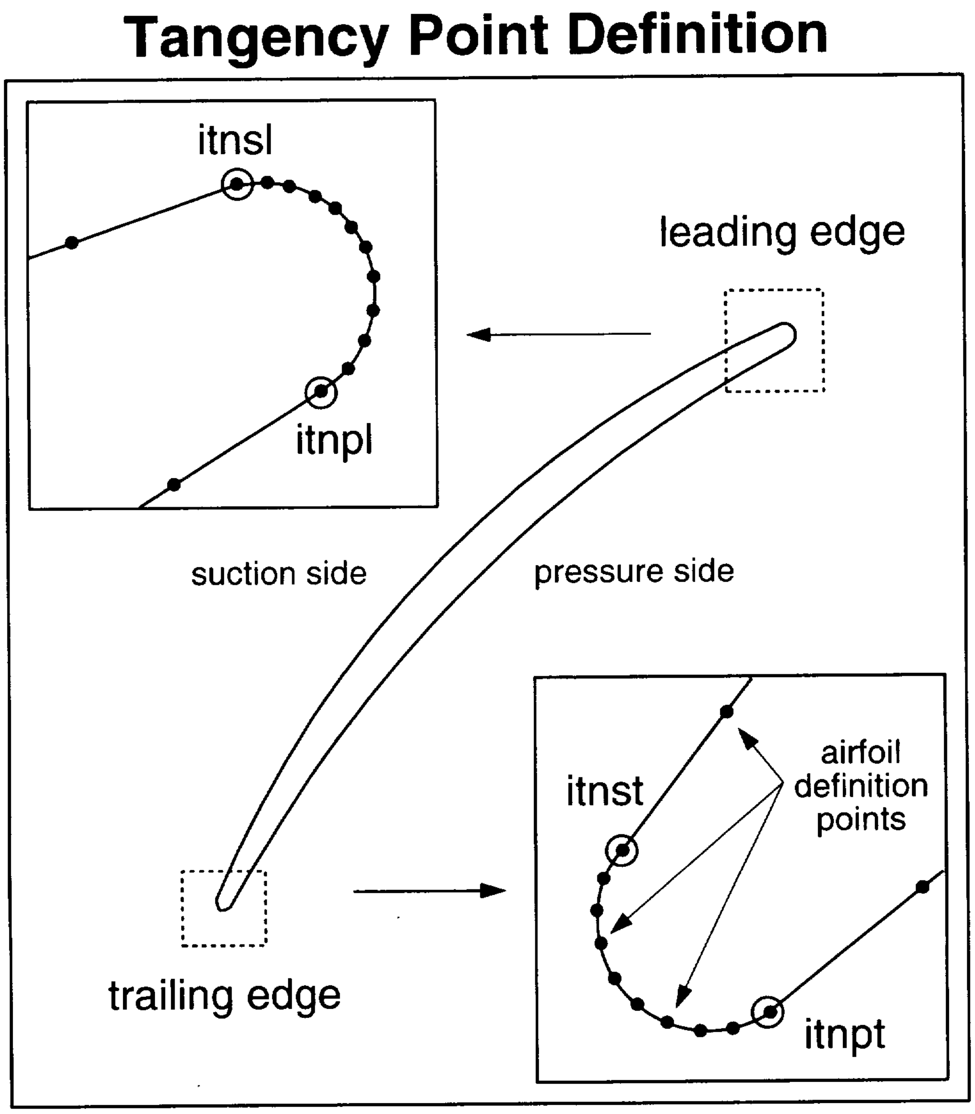


Figure 4.2: *TADS* tangency point locations on an example airfoil shape. Abbreviations shown correspond to the text labels in the `casename.tdsaro` file.

The body force file has to be generated outside of *TADS* because the implicit assumption when running the design mode is that there is no blade geometry available. Without a blade geometry, there is no way to set the blockage value at a given X,R location. This shortcoming of the design mode is the impetus for an enhanced capability called the "true" design mode. This mode, which is currently under development for *TADS* , will allow the user to specify a type of airfoil (e.g. 65 Series, Double Circular Arc, etc.). *TADS* will then calculate the appropriate blockage for use in *ADPAC*.

4.5 rV_θ Design File

The rV_θ design file is an output file **`casename.rvtdesign`** generated by the *TADS* pre-processor (*PREDESIGN*). It is a keyword driven file very similar to an *ADPAC* **`casename.boundata`** file. Following the *TADS* convention, the rV_θ file is named **`casename.rvtdesign`**. A sample input file can be found in Appendix D.

The rV_θ design file contains most of the extra information required by *ADPAC* and *BODYF*. Radial profiles of rV_θ vs. radius specify how to apply work in rotors and blades or swirl in stators and vanes. Units of rV_θ are consistent with those in the *ADPAC* file described in the following section.

Data for specifying the axial distribution of rV_θ between the leading and trailing edges is also present. The POWER header under the AXRVTDIST keyword specifies the exponent on the quarter sine wave distribution, It affects the rV_θ distribution as follows:

$$(rV_\theta)_x = (rV_\theta)_{LE} + \Delta(rV_\theta) * sin\left(\left[\frac{(x - x_{LE})}{C_{axial}} * \frac{\pi}{2}\right]^{POWER}\right)$$

where x is axial distance, the LE subscript denotes a quantity at the leading edge, and C_{axial} is axial chord. Setting a POWER value of 0.0 causes the pre-processor to generate a linear distribution.

The rV_θ file is both read and written by the TADS design mode pre-processor. If it does not already exist, the pre-processor will create it using default values. It is mentioned in the this chapter because the user has the ability to create/edit this file directly just like any other input file. However, care must be taken that it is not altered in such a way that it is made unreadable by the pre-processor.

4.6 rV_θ File

The rV_θ file is a 3-D, multiple block, binary solution file in *PLOT3D* format, but written in ASCII. It is in ASCII so that the user can more easily create, check and edit the file. Because the design mode is a recent addition to *ADPAC*, this ability to debug/check the rV_θ file directly is important for code development and ease of use. Once the design mode becomes more mature, the rV_θ file format will be changed to SDB binary.

The format of the rV_θ file closely resembles an *ADPAC* restart file (the exact format and FORTRAN statements required are given in Appendix E) because values are given at the grid cell centers. Each value in the file is the product of the cell center radius and absolute circumferential velocity. Radius units are consistent with grid units and velocity is given in ft/s. As in an *ADPAC* restart file, the order of the indices is consistent with the

ADPAC grid file (I and J indices represent axial and radial directions respectively and the K index, the tangential direction, is set to 1.)

Following the *TADS* convention, the file is named `casename.rvt.#`, where the # suffix denotes the block number. As with the *ADPAC* body force file, the suffix numbering scheme allows the user to assign rV_θ values to individual blocks in a multiple block calculation. **NOTE:** although a separate `casename.rvt.#` is required for each row where a design mode calculation is being performed, the file itself is still multi-block. The block number and filename are specified in the *ADPAC* input file exactly like the body force file.

During a *TADS* run, the *BODYF* routine creates the rV_θ file, so it is not a true required input file. It is included here because the user may possess some alternate means to specify a rV_θ distribution which is not based on a quarter sine wave method.

4.7 Total Pressure Loss Data

The total pressure loss file contains a table of information at the blade trailing edge. Following the *TADS* convention, the name of this file is `casename.tpl.1`. As with the *ADPAC* body force file, the suffix numbering scheme allows the user to assign total pressure loss coefficient values to individual blocks in a multiple block calculation. The block number and filename are specified in the *ADPAC* input file exactly like the body force file as well. The file is in ASCII format and is read with free format by FORTRAN subroutines. A sample input file is found in Appendix F.

The `casename.tpl.1` file is largely self-documenting, with comment lines preceding each data entry. The contents of the comment lines are ignored, but there must be at least a blank line where each comment belongs.

Two comment lines precede the first data item. The first item is a flag which indicates the type of loss coefficient that is being applied at the trailing edge of the blade row. A value of 1 indicates that the loss coefficients are for compressors which are defined as:

$$\overline{\omega} = \frac{\overline{P}'_{T_{ideal2}} - \overline{P}'_{T_2}}{\overline{P}'_{T_1} - \overline{p}_1}$$

A value of 2 indicates turbine loss coefficients which are given as:

$$\overline{\omega} = \frac{\overline{P}'_{T_{ideal2}} - \overline{P}'_{T_2}}{\overline{P}'_{T_2} - \overline{p}_2}$$

A value of 3 is an alternate loss formulation given by:

$$\overline{\omega} = \frac{\overline{P}'_{T_{ideal2}} - \overline{P}'_{T_2}}{\overline{P}'_{T_1}}$$

where the P_T indicates a total pressure, p denotes a static pressure, a $'$ superscript refers to a relative quantity, and the leading and trailing edge locations are 1 and 2 respectively.

The second item is the number of radial stations for which there is aerodynamic data. This number must correspond to the number of table entries which follow, but does not need to correspond to the number of spanwise points in the airfoil description (`casename.tdsblad`) or to the number of meridional streamlines to be used in the analysis. The table has three columns of data consisting of the radius, the loss coefficient value, and the axial location. The radius and axial location values are in inches and the radius points should be monotonically increasing with no repeating points. The axial location values are not used by *ADPAC*, but are present more as a reference for the user.

The last data item is a flag that defines the two I indices between which the total pressure loss will be applied. Currently this option is not fully coded and the default entry of " 0 0 " tells *ADPAC* to apply the total pressure loss from the leading edge index to the trailing edge index. Hence, the table of loss coefficient values should correspond to the blade trailing edge.

Chapter 5

Main Panel

The function of the main panel (shown in Figure 5.1) is to direct work flow. The work being directed can be divided into three categories: configuration, data input and execution. However, a more logical breakdown organizes these categories into five operating modes:

1. Edit Programs
2. Edit Data
3. Edit/Run
4. Run
5. Edit Machines

The program mode selector (see Figure 5.2) sets the active mode and determines the appearance of the component group controls (see Figure 5.3). These controls give the user the flexibility to loop on specific aspects of a solution by managing input data and component module execution. Depending on the active mode, these controls display the name and status for each component module's active program module or it's associated execution host. The operation and appearance of this modal display is discussed below for each operation mode.

The main panel also contains five action buttons in the lower right corner. There is a **Quit** button to exit the GUI, a **Shell** button to open a UNIX shell in the current working directory and a **Setup** button to configure remote execution of component modules. The two other buttons, **Post** and **Design**, are used to activate the *TADS* post processor (*POST*) and the design mode pre-processor (*PREDESIGN*).

5.1 Edit Programs Mode

The "Edit Programs" mode allows the user to select which program modules to execute. For example, the user may select TIGG or BATCH TIGG as the axisymmetric grid generator.

The available choices are displayed in a pulldown list which is activated by clicking on the downward pointing arrowhead located to the right of the program labels. Note, only component groups with more than one option will display the arrowhead.

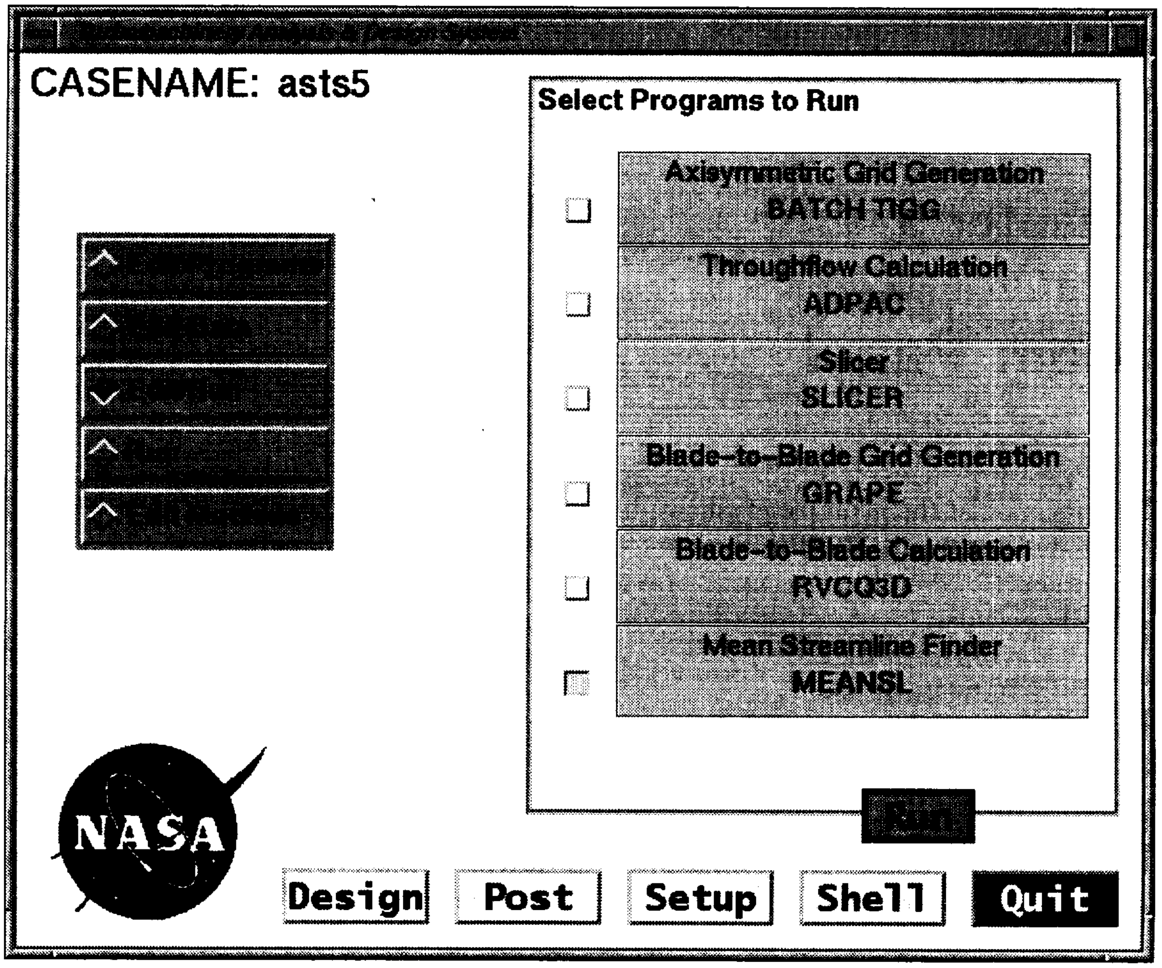


Figure 5.1: *TADS* main panel controls the coupled analysis.

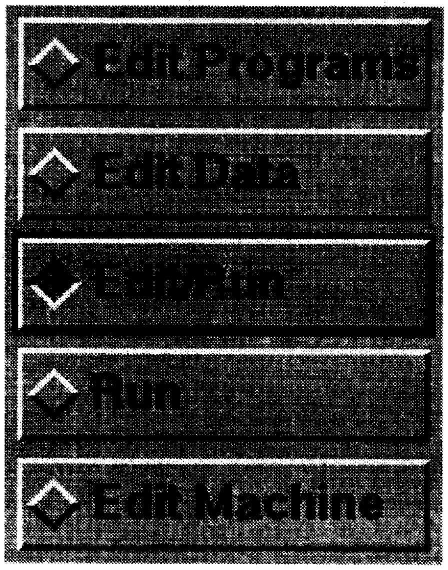


Figure 5.2: Program mode selector controls the GUI's appearance and execution.

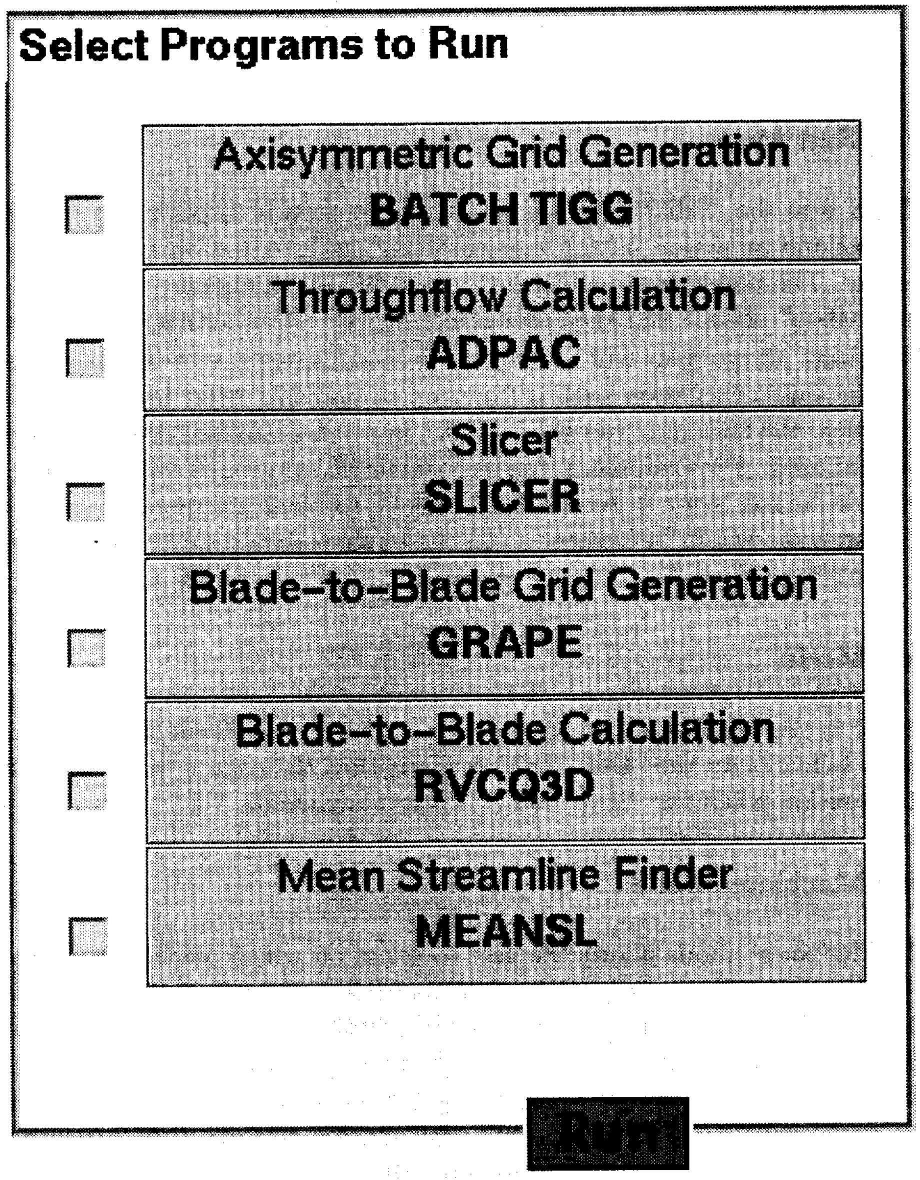


Figure 5.3: The component group controls change in appearance and function as the program mode is changed.

5.2 Edit Data Mode

When *TADS* is in the "Edit Data" mode, the program labels become push-buttons. When the push-button for a component is clicked, the main panel is replaced by the input panel for the component's active program module. In general, each component program has its own data input screen. However, in some instances different programs may share the same input screen (as is the case for TIGG and BATCH TIGG). The individual input windows are discussed later in this document.

5.3 Edit/Run Mode

When *TADS* is in the "Edit/Run" mode, a toggle button is displayed to the left of each program label and an action button labeled "Run" is displayed beneath the component group display. The toggle buttons determine which components are to be edited (see "Edit Data Mode" above) and then executed when the **Run** action button is clicked.

Warning: all modules will attempt to execute regardless of data availability. Therefore, the safest execution path is from top to bottom. After the loop has been completed once, the user may run the modules in a more random order; however, caution should still be used. For example, if the axisymmetric solution is modified, the slicer module should be run prior to rerunning the blade-to-blade solution. A more obvious example of an error would be trying to run the blade-to-blade solver before generating the grid.

5.4 Run Mode

When *TADS* is the **Run** mode its appearance is identical to "Edit/Run" mode. The only difference in behavior to the "Edit/Run" mode is that the data is not edited prior to program execution when the "Run" action button is clicked.

5.5 Edit Machines Mode

The "Edit Machines" mode allows the user to select on which machine a component module is to execute. For example, the user may select "Local" to run a program on the same machine which is running *TADS* or some remote machine to execute a module across the network. All component tasks default to the local machine unless *TADS* is being run as a restart, in which case the previous configuration data is restored from a file. Some programs, such as TIGGC3D must be run with specific machine combinations[1] in order for the interactive graphics (GL) to work properly.

To change the machine associated with a component module, the user simply selects the desired machine from the pulldown list of available machines for the particular component group. The list of available machines is controlled by the remote processor setup panel (see Chapter 6) and is the same for all component groups. This mode is only available if more than one machine exists in the list.

[1]Program must be run on and displayed to a machine that supports GL (e.g. SGI).

Chapter 6

Remote Processor Setup Panel

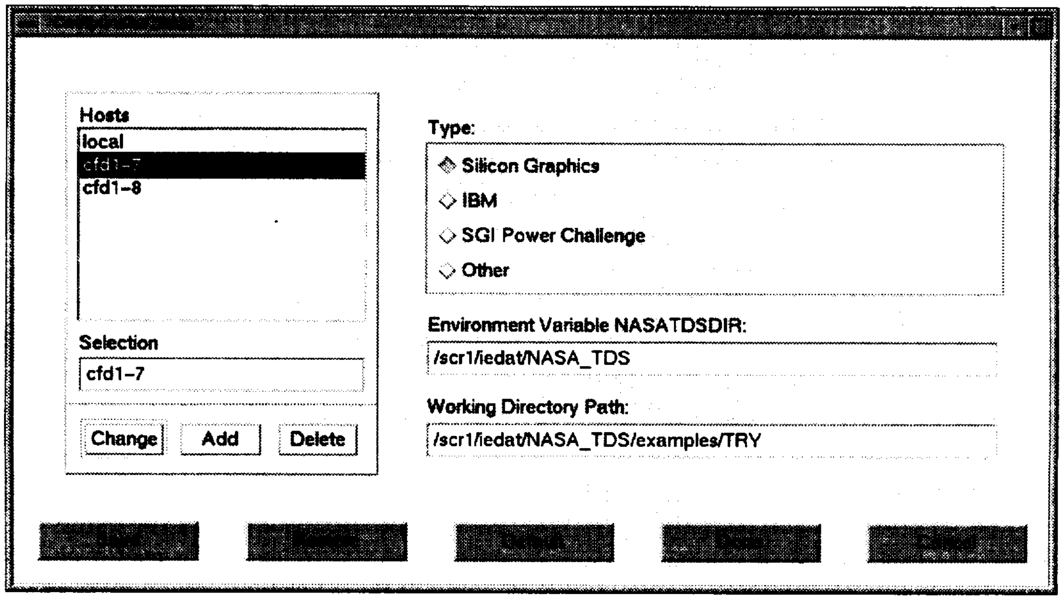
TADS has the capability to distribute tasks to networked machines. The remote processor setup panel allows the user to configure a list of available machines. From this panel (see Figure 6.1) the user may add new machines and/or delete or modify existing machines from the available list. The panel allows the user to specify machine name, manufacturer, path to the current working directory and the path to the executables for each machine on the list. This information is saved to a file in the local working directory and is accessed upon restart. The remote machine must have NFS file access to the local disk; however, it is not required that the directory paths on the remote machine be the same as the local machine, although that is the default value.

If the user copies or moves the working directory to another location, the paths in this file (`casename.configure`) must also be updated to reflect the new directory paths. Otherwise, *TADS* will attempt to continue operating on the files in the original path. These same precautions must be taken when *TADS* is run on a different "local" machine. *TADS* will automatically know the type of the machine it is running on; however, the original path names will be read from the `casename.configure` file.

The structure of the configuration panel allows the user to include the same machine more than once in the list of available machines. This feature is useful when different versions of the same program exist on one machine. The user can setup the second occurrence in the list to point at an applications directory different than the first. When this scenario is used, some confusion may occur because the list of machines displays the same machine name more than once with no distinguishing features. In this case the user must be aware of the order of the machines in the setup panel when choosing a machine from the main panel in the "Edit Machines" mode.

The action buttons located at the bottom of this panel have the same functions as those described for standardized input panels in Table 9.1.

Warning: A common error occurs when copying or moving an entire *TADS* case from one directory or file system to another. The user often fails to change the working directory path(s) in the setup panel (or directly in the `casename.configure` file).


Hosts

local

cfd1-7

cfd1-8

Selection

cfd1-7

Type:

[✓ Silicon Graphics]

IBM

SGI Power Challenge

Other

Environment Variable NASATDSDIR:

/scr1/iedat/NASA_TDS

Working Directory Path:

/scr1/iedat/NASA_TDS/examples/TRY

Figure 6.1: Program modules can be run on remote hosts configured using the Setup Panel.

Chapter 7

Pre-Design Module

During the development of the design mode for *TADS* , it was recognized that user would have to manage a substantial amount of data that would be difficult (a) to create outside of *TADS* and (b) to manipulate without a plotting interface that had a click-and-drag editing capability. These requirements were the impetus for creating the Design Mode Pre-processor Module. This first section of this chapter gives a brief overview on running this module. Subsequent sections detail features that are used to create a complete *TADS* design mode computation.

7.1 Running the Design Mode Pre-processor

The Design Mode Pre-processor Module, henceforth referred to as *PREDESIGN*, is a stand-alone module set up to run from the *TADS* main panel. When the "Design" pushbutton on the main panel is pressed, the main panel disappears and the *PREDESIGN* main panel appears. This main panel, shown in Figure 7.1, displays the primary flowpath and the leading and trailing edge blade definitions (from the `casename.tdspath` and `casename.tdsaro` files. The user has the option of modifying the shape and the number of points in the flowpath and blade leading and trailing edge definitions. There are several pop-up panels that can be invoked from the "Windows" pull-down menu. There are pop-ups for modifying the leading and trailing edge rV_θ profiles as well as the axial distribution along the blade axial chord. There is also a pop-up for choosing basic blade profile shapes, incidences and deviations, and leading and trailing edge aerodynamic quantities (although these latter three panels have limited functionality in the current version).

7.2 General Features

The Design Mode Pre-processor Module has an extensive set of functions for graphically manipulating design mode data. These functions are accessed through the edit menu on the main and pop-up panels (shown in Figure 7.1. The user has the ability to:

1. Add and delete points by clicking on a given area in a plot.
2. Move selected points in a plot by clicking and dragging (X- and Y-direction only moves are also available).

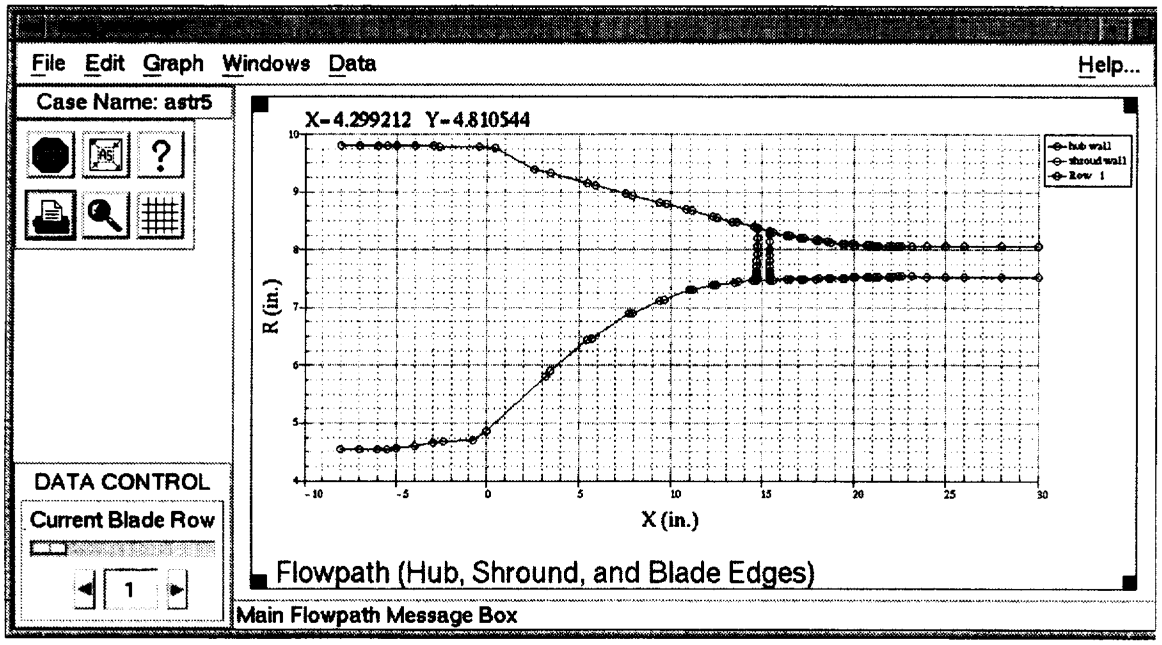


Figure 7.1: The *PREDESIGN* main panel is the starting point for a *TADS* design mode run.

3. Move selected points through a pop-up table (to allow for exact X and Y point placement).

Using the editing commands is rather straight forward, but there are a few general rules to follow.

Once the user has selected any one of the editing options, the mouse becomes active in the current plot. **Only** the currently selected plot is available for editing points. To edit another plot or data in another window, the user must exit the edit mode first. Any mouse clicks or dragging the mouse cursor out of the active plot window is ignored by *PREDESIGN* When deleting points, the user must make sure that at least two points remain in a given line. This will ensure a satisfactory definition for a given quantity in the resulting *TADS* files.

In addition to having a similar appearance and functionality to the *TADS* Post Processor, the Design Mode Pre-processor Module also has the identical file and graph manipulation functions (i.e. auto-scaling, title and legend set up, etc.). This commonality reduces the amount of time required to learn the both modules.

7.3 Pop-up Windows

On the main panel of the pre-processor, the "Windows" pull-down menu allows the user to pop-up 5 different windows for manipulating data. Each one of these pop-up windows has general (file, edit, graph) and specific pull-down menus required to create/modify data for a *TADS* design mode run.

The first pop-up panel allows the user to change rV_θ profiles at the leading and trailing edges of the current blade row. Since there are two plots in this window (as shown in Figure 7.2), the user must take care that a given plot has been selected (a left mouse

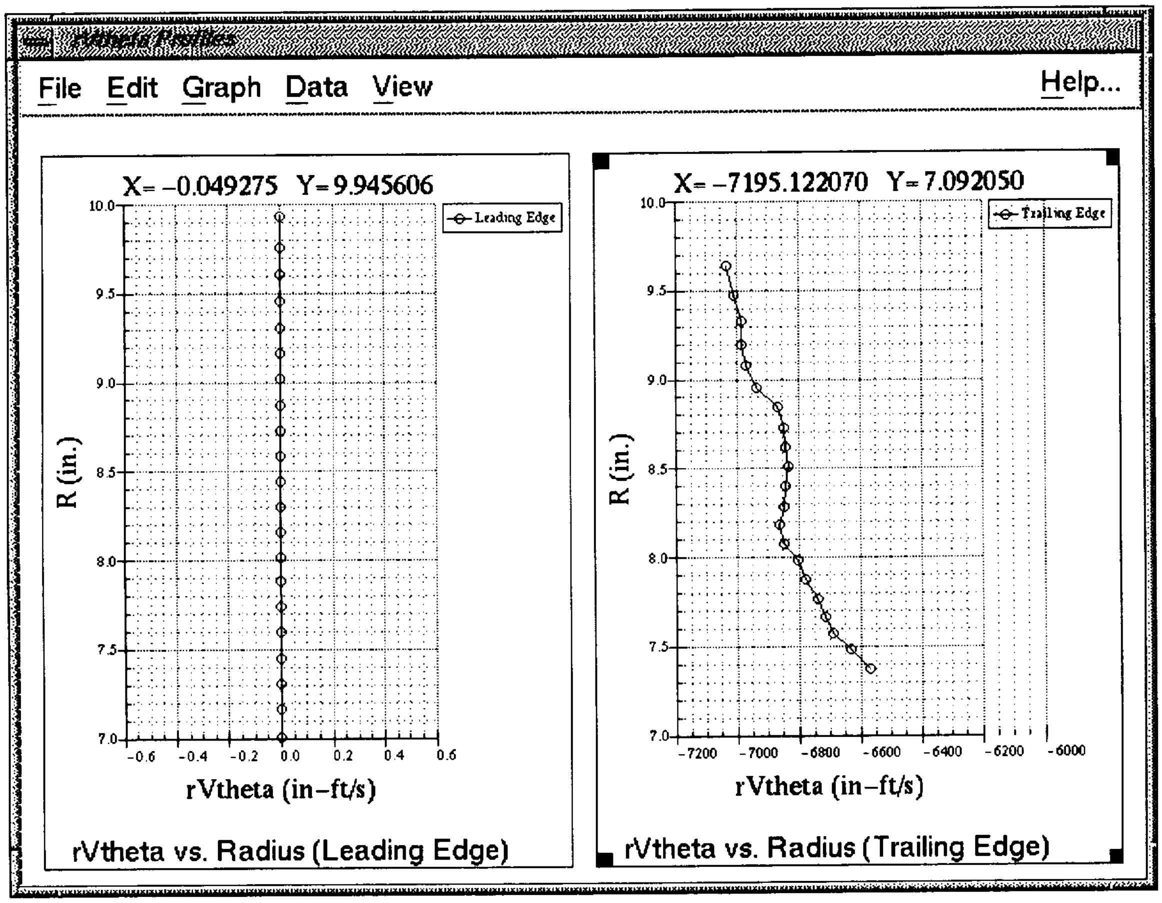


Figure 7.2: The radial distribution of rV_Θ at the leading and trailing edges of the blade row can be modified in the the *PREDESIGN* Radial Panel.

click activates given graph) before it is manipulated. The user may write the new rV_θ definition to the `casename.rvtdesign` file at any time. Note that the `casename.rvtdesign` file is **not** written automatically when the rV_θ panel is closed or upon exiting *PREDESIGN*; the user **must** cause the write. This ensures that the original `casename.rvtdesign` file is not written over by accident. If the `casename.rvtdesign` file does not exist when the pre-processor is invoked, the rV_θ profile is calculated using values from the `casename.tdsaro` file.

The second pop-up panel is used to modify the axial distribution of rV_θ. The user has a choice of linear or quarter sine wave (QSW) power distributions along the axial chord. These options are accessible through the "Data" pull-down menu shown in Figure 7.3 The first few options, when selected, change the axial distribution at every radial point. The "Table..." option allows the user to edit each radial point individually. The initial default setting is a QSW distribution of power 1.0 at every radial location.

The third pop-up panel lets the user choose a blade profile shape at a given radial location. The fourth panel is used to set incidence and deviation at the blade leading and trailing edges. The fifth and final panel is used to set the leading and trailing edge aerodynamic quantities which appear in the `casename.tdsaro` file. In the current version of *PREDESIGN*, panels three, four, and five are not fully functional because they were not part of the original *TADS* contract requirement. They have been included in their

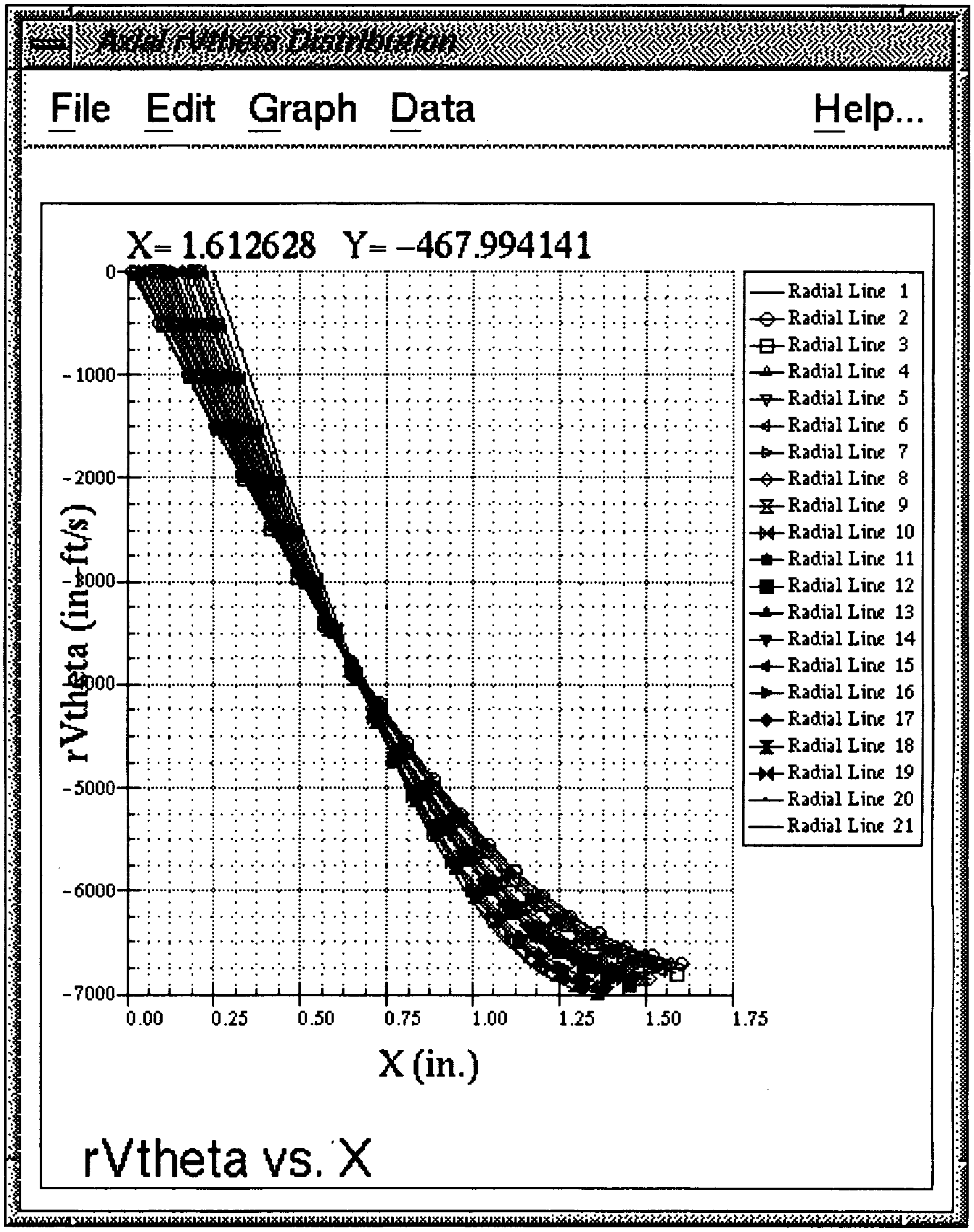


Figure 7.3: The axial distribution of rV_Θ can be modified in the the *PREDESIGN* Axial Panel.

partially functional form to provide a possible starting points for future work.

It should be noted that there is currently no means of directly specifying the blockage in the Pre-processor Module (for use in the `casename.bf.#` file of the subsequent *ADPAC* design mode run). This deficiency stems directly from the fact that most blades are designed from a thickness distribution normal to a mean camber line. In the *ADPAC* design mode computation, this mean camber line changes with every iteration. Hence, there is almost no way of guaranteeing a smooth blade shape because the original blockage factor is based on an assumed mean camber line (which may be radically different shape at the end of the *ADPAC* design mode run.

One possible remedy to this situation is to code blade generation routines directly into *ADPAC* Then, the value of the blockage could be updated by taking the current camber and blade specification (i.e. MCA, DCA, etc.) into consideration. The task of coding blading routines into *ADPAC* was begun, but it was felt that the level of effort was beyond the scope of the task 10 contract requirements.

7.4 Required Files

The Design mode Pre-processor Module requires no files *a priori*. However, when setting up a new case, it is more practical to make a few preliminary files and then modify the data inside the pre-processor. As in *TADS* , the user should supply a flowpath description (`casename.tdspath`) and an aerodynamic information file (`casename.tdsaro`). Upon initialization, the modules also looks for a design mode file designated (`casename.rvtdesign`). This file is used to store data from a pre-processor run such as axial distribution and specific rV_θ profile values. If this file is not found, the module reverts to default values for various sets of data. A description of the the `casename.rvtdesign` format and various module defaults can be found in Appendix D.

As stated above, the Pre-processor Module is a stand-alone program and as such can be invoked without ever running *TADS* . This can be advantageous if the user needs to set up a case where the data required to start up *TADS* is either sparse or unavailable.

Chapter 8

Post Processor Module

TADS has a built-in post processing capability for analyzing the large amount of data that is generated from a *TADS* iteration(s). It has the ability to display not only aerodynamic information, but convergence histories and geometry as well. The TADS post processor, henceforth referred to as *POST*, is a stand-alone module accessible through the "Post" pushbutton off of the *TADS* main panel.

8.1 Running POST

When invoked from the main panel, the main panel of the post processor, as shown in Figure 8.1, appears. It searches the working directory for the `casename.tdsaro`, `casename.tdsaxi` files to establish reference quantities. It then reads in the `casename.mesh` and `casename.restart.new` from the *ADPAC* axisymmetric run. From these files, *POST* creates a plot of the meridional grid in the main windows. From this main window, the user has the option of invoking several pop-up windows (from under the Windows pull-down menu) which are described in the Pop-up Windows section.

8.2 General Features

The post processor has various features that ease the viewing of data. Pull-down menus, pushbuttons, and slider bars allow the user to control every aspect of the displayed plots. The user MUST select a plot in a given window before many of the features detailed below can be utilized. This is accomplished by simply left mouse-clicking a plot in a given window on a chosen plot. Common menus are used wherever possible to create a consistent working environment. A good example of this idea are the File and Graph menus. They are common to every window (main and the pop-ups). The File menu has the following features:

1. Write Sets option: lets the user write individual X,Y data sets to file.
2. Read Sets option: lets the user import an X,Y data set into the currently selected plot.
3. Print option: brings up a window from which the user can choose to output monochrome or color postscript hard-copies to either a file or a local printer.

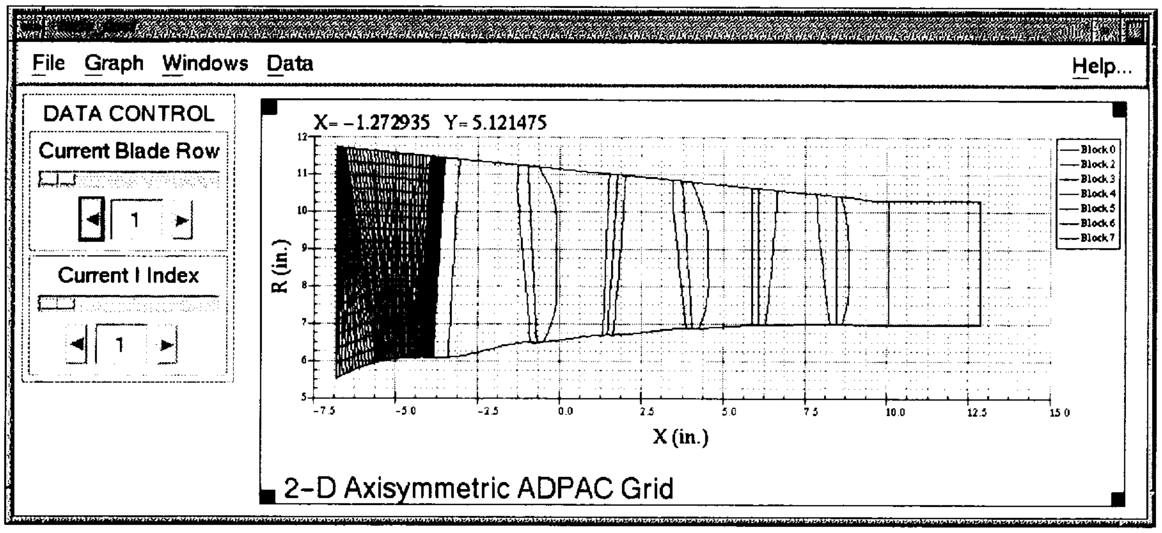


Figure 8.1: The *POST* main panel is the starting point for post processing *TADS* results.

4. Exit option: For pop-ups, this closes the individual window. For the main panel, it exits the post processor.

The format and options for the Graph pull-down menu were patterned from a X-Windows general plotting package called ACE/gr [9]. It has the following features:

1. The plot title, axes labels, and legend text can be changed. Different fonts, styles, and point sizes are also available.

2. The user can toggle certain features on and off such as the X,Y location tracker, the major and minor grid lines, the plot title and legend, and the plot axes labels and numbers.

3. Zooming in and out and auto-scaling (based on the currently displayed data) is available. The user can also specify X and Y ranges directly by using the World Scaling option.

4. The style and color of each line in the selected plot can be customized by using the Lines option.

8.3 Pop-up Windows

The *POST* module has several pop-up windows for analyzing different types of data. Each of these pop-ups can be invoked from the "Windows" pull-down off of the main panel. Because row and i-location (when applicable) control exist on the main panel and the data in the pop-ups often depends on which row or i-location is currently selected, there is a control mechanism for updating data in the pop-ups. In the Data pull-down menu of every pop-up, there is an "Update" pushbutton and "Continuous Update" toggle button. If there are a large number of windows active and the user wishes to reduce refresh overhead, then each window can be updated with the pushbutton. If the "Continuous Update" toggle is on, then the window is refreshed every time the row or i- location changes in the main panel. Note that some of the pop-ups, like the grid and contour windows, are not fully functional. This was done whenever development time would have been prohibitive or when a more capable software program (e.g. Plot3D for Contours) would clearly provide better results. These windows have been remained in case any future development is undertaken,

The first pop-up panel is used to view radial profiles of aerodynamic quantities. When the post processor reads in a solution file, circumferential averages at each axial and radial node are calculated. The user can then choose a row and "i" location on the main panel and the type of average (static pressure, total pressure static temperature, etc.) under the Data pull-down menu.

The second pop-up panel similar to the first, except it is used to view axial averages of aerodynamic quantities. Each axial average is computed by spanwise averaging the radial quantities at each "i" location. Since, for each radial point, not all of the x locations are coincident, the x point for the axial average is an average of each radial x point.

The third pop-up is used to view convergence histories. When this panel is invoked, he post processor searches for the appropriate convergence file. For *ADPAC* runs, there are four plots in the window. They show RMS error, maximum error, mass flow in/out, total pressure ratio and efficiency all vs. number of iterations. Under the Data pull-down menu, the user the option of choosing *ADPAC* (default), *RVCQ3D*, or *B2BADPAC* convergence. The *RVCQ3D* option invokes a dialog window so the user can choose an appropriate slice.

The remaining four pop-up windows are not active.

8.4 Example POST Session

In a typical *TADS* iteration, the user would probably use the post processor as follows:

1. After performing an axisymmetric *ADPAC* computation, invoke *POST* from the main panel.
2. Invoke the convergence window and check on the quality of the *ADPAC* convergence.
3. Invoke Radial and Axial Profile pop-up windows to examine the solution at various row and "i" locations.
4. Exit the *POST* and run the *SLICER* module.
5. After running several slices (or complete rows) of blade-to-blade solutions, the user would again invoke *POST* to examine convergence quality of the *RVCQ3D* or *B2BADPAC* computations. The ability to examine blade-to-blade aerodynamic data is not yet fully functional in *POST*.

Chapter 9

Input Panels

All the data input panels create an input file for the associated program module. There are several advantages to this approach.

The most obvious advantage is that restarting *TADS* from a previous case is transparent to the user. Also, should problems be encountered, the user may execute the modules outside of the GUI in order to isolate the source of the problem. The GUI does not provide the user access to all input variables, but is restricted to variables which are most likely to be of interest to the user. If the user wishes to modify input data which is not directly available through the GUI he/she may edit directly into the input file and make the desired modifications. This can be useful when new variables are added to a module. [1] Another benefit is that existing (non-*TADS*) input files may be used for initial input providing the files follow the *TADS* naming conventions and formats, and only data which is accessible from the GUI is used. Some modules have only limited use of this feature - see individual input panel descriptions for more information.

[1]For this to work properly, the user must **Shell** out from the main panel to edit the data. After editing is complete do not enter the input panel for the desired program module, because this will rewrite the input file (as will *TADS* initialization).

Table 9.1: Action buttons on standardized input panels control file creation, modification and restoration.

Save	Overstore current panel data to a file if changes have been made. If no changes have been made, then no action is taken.
Restore	Restore current panel data from a file. Any changes not saved prior to a **restore** are lost. This action button is only active if the input file exists (from a previous **save**).
Default	Reset current panel data to default values (except for locked data - see below). These defaults are setup specifically for *TADS*. This means they are not necessarily the same as the defaults stated in the formal documentation of the individual component modules. Any changes not saved prior to a **default** are lost.
Done	**Save** current data (see above) and then exit current panel. In some instances, this action button will force the execution of secondary component programs such as preprocessors. When this situation occurs it will be stated in the documentation. Also, a message will appear in the message panel indicating any programs being executed; however, this message may not be seen due to the short execution times of most of these secondary programs.
Cancel	Exit current panel without saving current changes. If a **save** has been done prior to **cancel**, secondary programs will be executed (if appropriate) as described above for **done**. If changes have been made to the data without a **save** being done, the user will be so informed and given the option to return to the current panel.

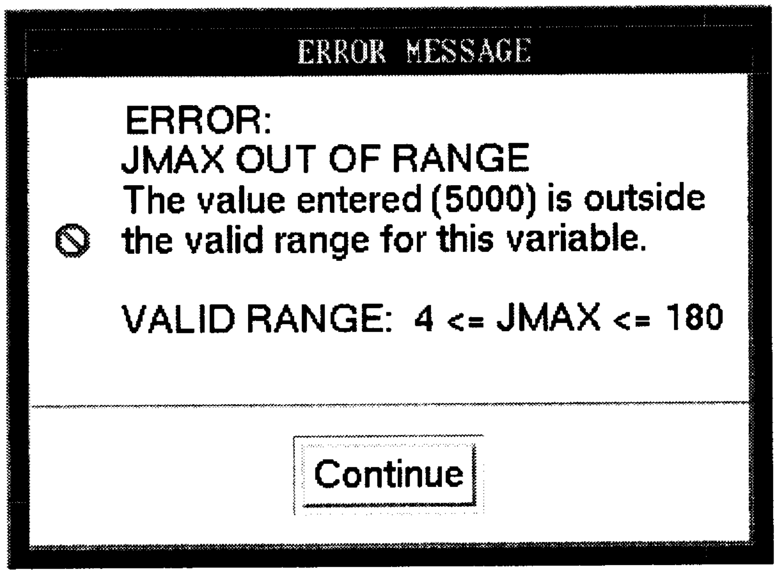
Generally, *TADS* draws upon three sources of data to create an input file.

1. Initial input files - basic flowpath/airfoil geometric and aerodynamic information
2. User data (via the GUI)
3. Internal data - output from *TADS* component modules

All of the input panels in *TADS* have a row of action buttons located across the bottom of the panel. Generally, these action buttons control file creation/modification and occasionally program execution. Unless specifically noted, these buttons behave as described in Table 9.1 for all input panels.

9.1 Standardized Data Input Panels

At present, all of the component module input panels, with the exception of the slicer module, are patterned after the same model. This model has three distinct subclasses: row dependent, slice dependent and slice independent. The input panels of these


ERROR MESSAGE

ERROR:
JMAX OUT OF RANGE
The value entered (5000) is outside the valid range for this variable.

VALID RANGE: 4 <= JMAX <= 180

Figure 9.1: The error panel will display the valid range.

subclasses appear almost identical, except for some additional controls on the row dependent and slice dependent input panels. The additional controls are described in greater detail later in this section.

For all of the input panel classes, most of the user data is scalar (non-array) in nature. This allows the input panels for each component module to be very similar. For purposes of this GUI, scalar data is divided into three classes: boolean, trigger and numeric. Boolean data have only two valid values (True/False, On/Off, 0/1, etc.,...) and are represented by a toggle button in the GUI. A textual description of the logical state is displayed with the toggle button to clarify what state is active. Trigger data have a finite list of valid values. These values are sequential and incremented by positive one for each choice. Trigger data generally have fewer than ten choices and are represented in the GUI by a pulldown list. Some trigger data contain textual descriptions in the pulldown list, but others may only list the numeric values. Numeric data may be real or integer values. Depending on the specific instance, a valid range may be enforced. Numeric data are represented by a text entry box in the GUI.

If a numeric entry fails an active bounds checker, an error panel appears (see Figure 9.1 for an example). The invalid number the user attempted to enter is displayed on the error panel along with the valid data range. The entered value is reset to its previous value and the user is prompted to click **OK** to continue.

The user data is also divided into locked and unlocked categories. Unlocked data is modifiable by the user, whereas locked data is not. Locked data is usually extracted from internal data, or is only available under special operating conditions. To distinguish between locked and unlocked data, the GUI uses shading, borders and pixmaps. When a data field is locked out, its background color is set to match that of the working panel, and its border and any associated pixmap are removed. An unlocked text/toggle field has a white background framed by a border and an unlocked pulldown field displays a bordered pixmap of a downward pointing arrowhead.

Most of the input panels are setup to display a description of the active control on the status bar (located near the top of the panel). Currently, the only description displayed is the name of the input variable.

All of the *TADS* input panels have some degree of row dependence. In some panels, like the *TIGG* and blade-to-blade modules, the dependence is strong and changing the current row changes a large number of input fields. For other modules, like *ADPAC*, most of the input fields are row-independent and changing rows has little discernible effect. Every panel has a row pulldown menu which both shows and lets the user choose the active (current) row.

Slice dependent component modules (such as the blade-to-blade solver) require an input file for each desired slice. To maintain slice to slice data integrity some inputs must be constant for all slices, while others are allowed to vary. Therefore, some normally independent inputs are required to conform to an overall scheme. Using random access binary files to create a relational database allows individual slices to share "common" data, and therefore insure slice data integrity. This compact organization more efficiently manages the data while it is being manipulated by the GUI and increases user productivity by allowing a single change to affect all slices at once. These benefits come at the cost of uniformity. The action buttons at the bottom of a slice dependent panel behave in a slightly different manner than they do for slice independent panels. These differences are documented in Table 9.2.

Slice dependent input panels appear almost identical to slice independent panels. One difference is the addition of a pulldown list at the top right of the panel next to the row pulldown list. The pulldown list allows the user to select an individual slice or all the slices. The only other visual difference is an additional action button labeled **Run** at the bottom of the panel. This action button executes the associated component module for the currently active slice and row and allows the user to debug input data without having to run solutions on all slices. This button also allows execution of all slices for a given row so that user does not have to run solutions on all rows. The button is only active in the "Edit Data" mode.

The multi-slice capabilities of the GUI required a visual method of distinguishing slice dependent and independent variables. This was done with the locked/unlocked feature described above. Data that must be held constant from slice to slice is locked out for individual slices. This also prevents the user from inadvertently changing slice independent data for a single slice. In the input panels, there's no need to distinguish between row dependent and independent quantities because there is no equivalent "All Rows" mode.

9.1.1 *TIGG* Input Panel

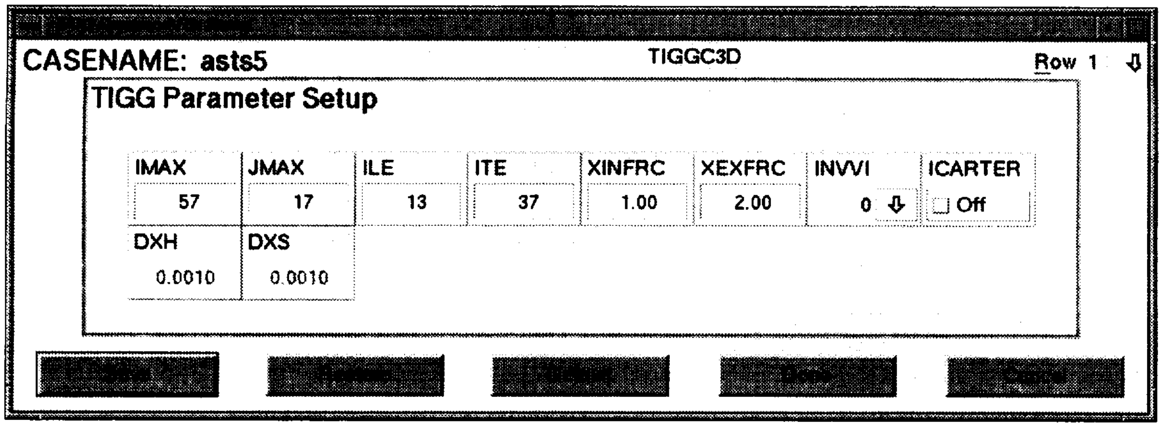
TIGG is a slice independent component module. It uses a standardized input panel as described in section 9.1. A representative screen image of it's GUI input panel is shown if Figure 9.2. The controls correspond to input described in the *TIGG* user's manual ([7]). This input panel is used for both *TIGG* and *BATCH TIGG* [2] component modules. This is possible because the input for both is identical.

There are a few extra triggers on the *TIGG* input panel that were not present in the *TADS* 1.0 version. The fields for DXH and DXS were put in place so that the user could modify the near wall grid spacing to give the resolution necessary for a no-slip wall

[2] *BATCH TIGG* is *tiggc3d* executed with the "-2d" flag to bypass the GL dependent GUI.

Table 9.2: Action buttons on "Slicer" input panel control file creation, modification, restoration and program execution.

Save	Overstore current panel data to the database file for the active slice if changes have been made. If no changes have been made, then no action is taken. The actual input files for the component modules do not get written out by a **save** as they do for slice independent panels.
Restore	Restore current panel data from the database file for the active slice. Any changes not saved prior to a **restore** are lost. The **restore** action button is inactive in the "All Slices" mode.
Default	Normal operation (see **Default** in Table 9.1).
Done	**Save** current data, write out component module input file for each slice and then exit current panel. Execution of secondary component programs is the same as described in Input Panels above.
Cancel	Write out component module input file for each slice without saving current changes and then exit current panel. Execution of secondary component programs and prompting for continuation is the same as described in Input Panels above.
Run	**Save** current data, write out input files and then execute current component module and related secondary programs. This action button appears only on multi-slice input panels. It is available only in the "Edit Data" mode and is deactivated for the "All Slices" option. This action is equivalent to running a component module in the "Edit/Run" mode (except for only one slice).


CASENAME: asts5 TIGGC3D Row 1

TIGG Parameter Setup

IMAX	JMAX	ILE	ITE	XINFRC	XEXFRC	INVVI	ICARTER
57	17	13	37	1.00	2.00	0	Off

DXH	DXS
0.0010	0.0010

Figure 9.2: *TIGG* input panel controls the axisymmetric grid generation.

condition. However, they have been made inactive because a true viscous mode for 2-D axisymmetric *ADPAC* is still being researched.

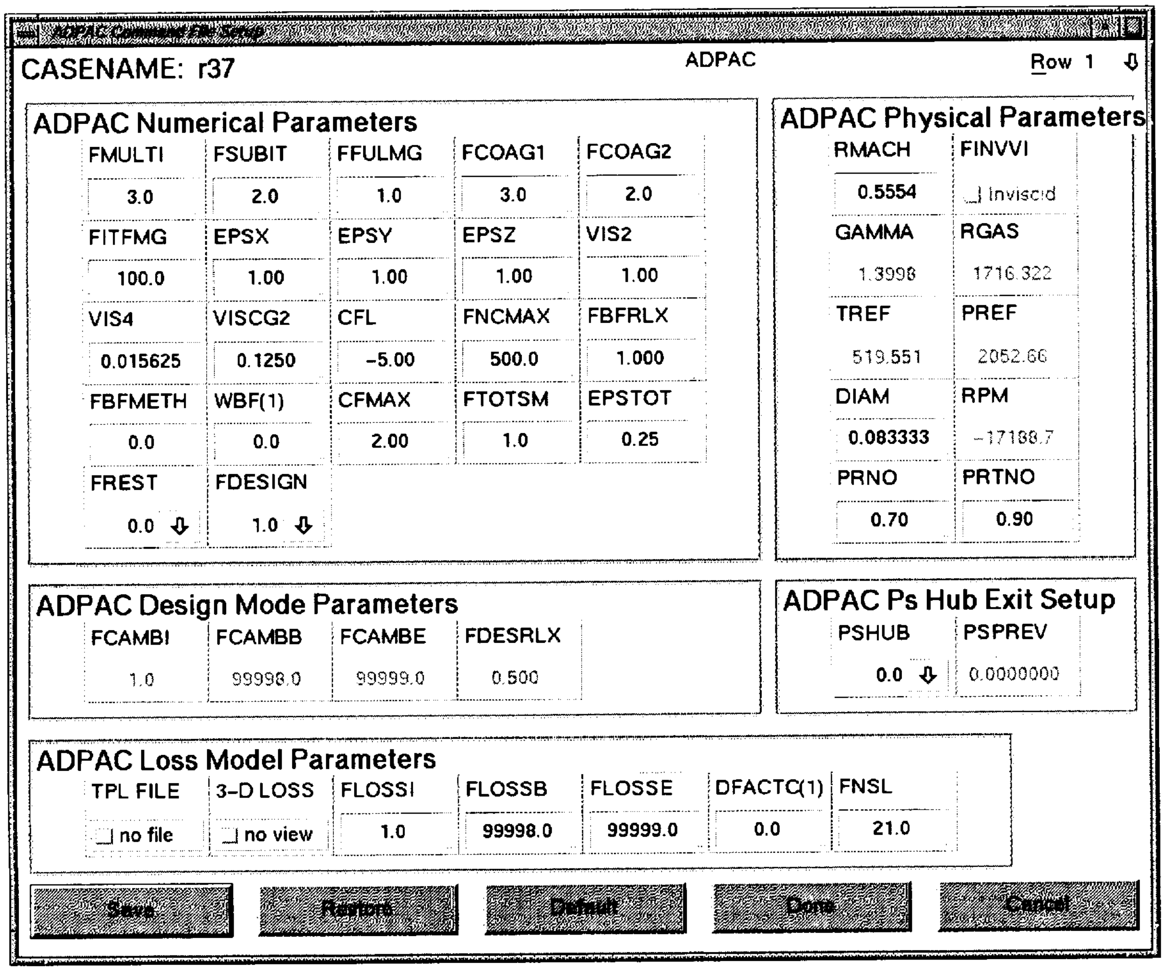
For multistage cases, if the user is not sure about what upstream or downstream spacing will result in a proper interface between adjacent blade rows, setting XEXFRC equal to 0.0 will force *INTIGG* to extend the downstream grid boundary to the upstream boundary of the following blade row. Similarly, setting XINFRC equal to 0.0 will extend the upstream boundary to the preceding blade row's downstream grid boundary. Setting both grid extent triggers to 0.0 for adjacent blade rows will result in a boundary that is half way between the trailing edge of the upstream blade and leading edge of the downstream blade. As a final check, if the grid extent triggers create a boundary which lies less than 5% chord from a blade or actually within a blade, then XINFRC and XEXFRC are set to 0.0 and the half way boundary method results.

When the *TIGG* input panel is invoked from the main panel, it reads in the `casename.tdsaxi` file to obtain the initial I and J maximum indices and the leading and trailing edge indices. These values are written to the native binary file `casename.tigg.db`. Any changes to the input fields are saved to the `casename.tigg.db` file if the user changes the current row or exits the panel. If the user exits the *TIGG* input panel, the binary file values are always written to the `casename.tdsaxi` file.

Warning: The file, `casename.tdsaxi` (located in the main working directory is only read once by *TADS* during initialization. However, this file is written out for each row every time the input panel is exited. The file which is read each time the input panel is displayed or the current row is changed is the native binary file `casename.tigg.db` in the working directory.

9.1.2 *ADPAC* Input Panel

ADPAC is a standardized, slice independent component module. A representative screen shot is shown if Figure 9.3. Most of the control labels correspond to the input keywords described in the *ADPAC* User's Manual [6]. Several of the remaining keywords are new additions to *ADPAC* and their complete specifications are given in Appendix G. The look and ease of use of the *ADPAC* input panel has been improved from the *TADS* 1.0 version. Input parameters are now sensibly grouped by their function within *ADPAC*.



CASENAME: r37 ADPAC Row 1

ADPAC Numerical Parameters

FMULTI	FSUBIT	FFULMG	FCOAG1	FCOAG2
3.0	2.0	1.0	3.0	2.0
FITFMG	EPSX	EPSY	EPSZ	VIS2
100.0	1.00	1.00	1.00	1.00
VIS4	VISCG2	CFL	FNCMAX	FBFRLX
0.015625	0.1250	-5.00	500.0	1.000
FBFMETH	WBF(1)	CFMAX	FTOTSM	EPSTOT
0.0	0.0	2.00	1.0	0.25
FREST	FDESIGN			
0.0	1.0			

ADPAC Physical Parameters

RMACH	FINVVI
0.5554	Inviscid
GAMMA	RGAS
1.3998	1716.322
TREF	PREF
519.551	2052.66
DIAM	RPM
0.083333	-17188.7
PRNO	PRTNO
0.70	0.90

ADPAC Design Mode Parameters

FCAMBI	FCAMBB	FCAMBE	FDESRLX
1.0	99998.0	99999.0	0.500

ADPAC Ps Hub Exit Setup

PSHUB	PSPREV
0.0	0.0000000

ADPAC Loss Model Parameters

TPL FILE	3-D LOSS	FLOSSI	FLOSSB	FLOSSE	DFACTC(1)	FNSL
no file	no view	1.0	99998.0	99999.0	0.0	21.0

Figure 9.3: *ADPAC* input panel controls the through-flow analysis.

The restart option (**FREST**) in *ADPAC* is not fully incorporated into this version of *TADS*. The trigger will cause *ADPAC* to attempt a restart; however, the restart file must be manually specified before the module is executed. This requires the user to provide the restart file before *TADS* is executed or to use the **Shell** feature on the main panel to perform the required file swapping/renaming.

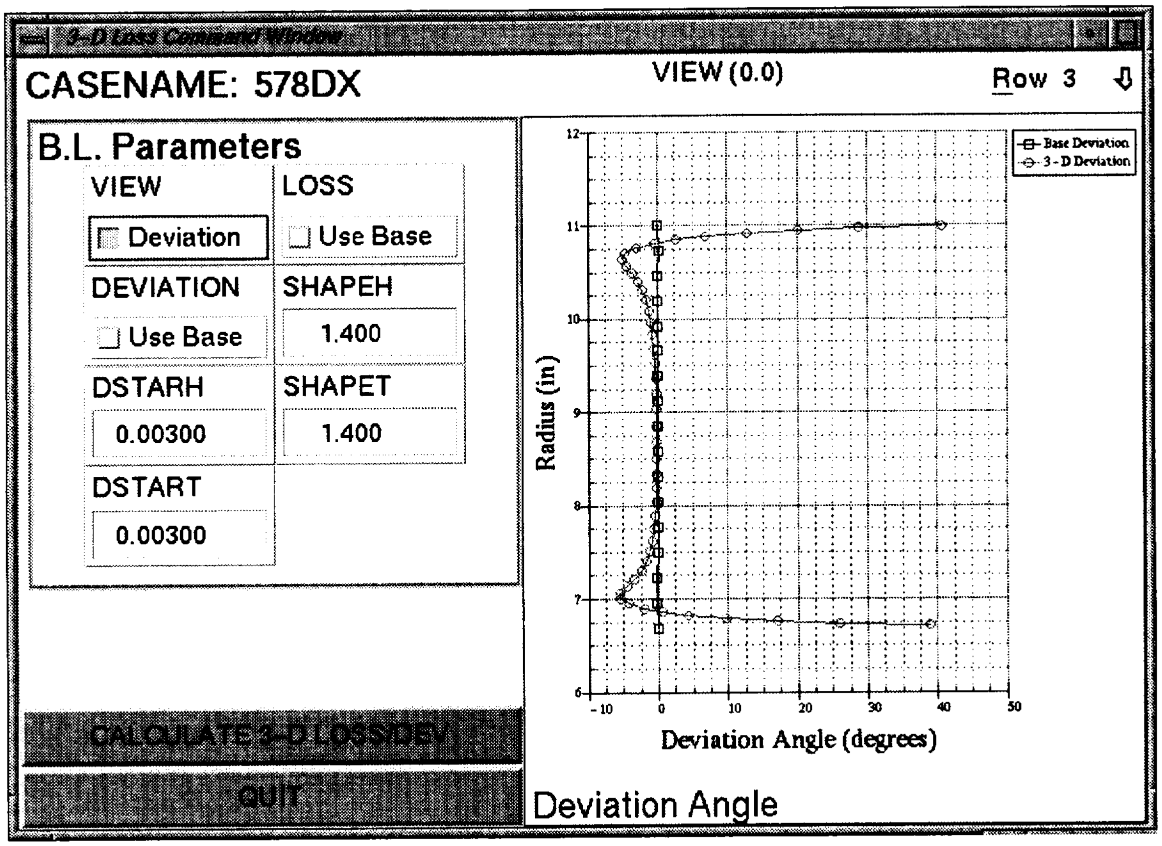
The short execution times demonstrated during the development phase of *TADS* reduced the priority of adding the additional logic and complexity required to perform the necessary file swapping/renaming. In other words, it runs so fast that more iterations are preferred over restarting. Another problem with this scheme is that the logic to allow the body force file (`casename.bf.1`) to be updated by *ADPAC* is not active. Therefore, *bodyf* will overwrite the body force file each time the *ADPAC* module is run from *TADS* .

In the previous version of TADS, *ADPAC* input parameters had to be arranged in the file `casename.adpac.input` just like they appeared on the input panel. Now, any order of input parameters may appear in the input file because the I/O routines are now keyword driven (just like *ADPAC* works internally). However, the input panel can only recognize a finite number of keywords, namely the ones that are pertinent to running 2-D axisymmetric cases. Similarly, it can only write those keywords that it recognizes. These conditions limit the user's ability to specify *ADPAC* input data prior to execution. Any legal *ADPAC* input parameters can be specified in any order, but initializing *TADS* will cause the *ADPAC* input file to be re-read and re-written with only recognizable keys.

A new parameter set called the ADPAC Ps hub group has been added to the *ADPAC* input panel. This group allows the user to view/modify the hub exit static pressure applied through the `casename.boundata` file. This field consists of pull-down menu (PSHUB) and a display area (PSPREV). The pull-down menu has three possible options. A value of 0.0 means that the trailing edge static pressure will be determined (in *adpacbc*) by using the turning through the blade row and by assuming 100% efficiency. Then, exit Mach numbers and mesh geometry are used to extrapolate that static pressure to the exit of the computational domain. In the 1.0 version of *TADS* , this was the <u>only</u> way to set the back pressure; the user would have to go outside of *TADS* GUI to directly modify the `casename.boundata` file. Choosing a value of 1.0 will trigger *adpacbc* to use the trailing edge hub static pressure from the `casename.tdsaro` file. This pressure is then extrapolated to the computational domain exit again using trailing edge Mach numbers and mesh geometry. A value of 2.0 for PSHUB allows the user to enter an exit hub static pressure value directly by un-locking the PSPREV field. The 2.0 option is particularly valuable when the user is running a compressor speed line or is analyzing a range of loadings on a blade geometry. The 2.0 option is also useful because the value is held in an external file (`casename.pshub`) so that the value is available even if the user exits TADS or runs through another complete TADS iteration.

9.1.3 *3DLOSS* Input Panel

3DLOSS is a standardized, slice independent component module. It is different than most other TADS module in that (1) it has a plot window associated with it and (2) it is present only to provide an extra input file for *ADPAC*. *3DLOSS* is so dependent on *ADPAC* that it should be a panel off of the ADPAC input panel. In order to maintain a standardized look to TADS, however, it was made into a true module.



CASENAME: 578DX VIEW (0.0) Row 3

B.L. Parameters

VIEW	LOSS
Deviation	Use Base
DEVIATION	SHAPEH
Use Base	1.400
DSTARH	SHAPET
0.00300	1.400
DSTART	
0.00300	

Figure 9.4: *3DLOSS* input panel controls the application of total pressure loss and deviation for *ADPAC*. The 3-D and base deviation is shown here.

3DLOSS is invoked from the main panel the same way as the *ADPAC* input panel. Once started, *3DLOSS* reads in the `casename.mesh`, `casename.restart.new` and `casename.tdsaxi`. Because it needs ADPAC output files, the user must perform a previous ADPAC axisymmetric computation. This is a primary requirement (and hindrance) to running the *3DLOSS* module, but it is one that cannot be circumvented because of the input data that the spanwise mixing model requires. *3DLOSS* also reads in the current total pressure loss coefficient profile if it exists. All of the input file information is used to plot total pressure loss and deviation vs. trailing edge radius. The user can toggle between the loss and the deviation plots through the VIEW button in the main text fields. When the "CALCULATE 3-D LOSS/DEVIATION" button on the panel is pressed, an external spanwise mixing code called *secflo* is run in the the `casename.row.#` directory corresponding to the currently selected row. *secflo* is a spanwise mixing code based on a method developed by Adkins and Smith, [1]. The output of the *secflo* program is two radial distributions. The first is the additional deviation created by the secondary flow and the other is additional total pressure loss. Each distribution is plotted against the original in the plot window as shown in Figure 9.4. The user then has the option of applying the base (original) loss or the 3-D loss to the ADPAC `casename.tpl.#` file for the current row. The option of applying the base or 3-D deviation is also available through the toggle on the DEVIATION field. A short description of each of the text field is given in Table 9.1.3.

Input Parameter	*Description*
VIEW	Toggles between deviation and loss coefficient data in the plot window.
LOSS	Lets the user choose whether to apply the base (original) loss coefficient profile or the profile that includes 3-D effects to the *ADPAC* `casename.tpl.#` file.
DEVIATION	Same as LOSS key. The deviation is applied to the axisymmetric *ADPAC* grid in the *bodyf* routine.
DSTARH	Hub inlet displacement thickness ($\delta_h{}^*$) multiplied by a reference length that is required by the loss model. Default is 0.003 The reference length is hub axial chord in feet.
DSTART	Tip inlet displacement thickness ($\delta_h{}^*$) multiplied by the reference length.
SHAPEH	Shape factor of the hub inlet boundary layer that is required by the loss model. Default is 1.4
SHAPET	Shape factor of the tip inlet boundary layer. Default is also 1.4

Table 9.3: Listing of the *3DLOSS* input parameters.

Input Parameter	*Description*
JCAP	Number of points on the upstream grid boundary. (Undocumented feature in original code).
SLE	Arclength of leading edge region (distance around airfoil between leading edge tangency points).
STE	Arclength of trailing edge region (distance around airfoil between trailing edge tangency points).
XUPFRC	Locates upstream grid boundary as a fraction of a distance. The distance is the average of the axial chord and the airfoil pitch. Replaces **XLEFT**.
XDNFRC	Locates downstream grid boundary as a fraction of a distance. The distance is the average of the axial chord and the airfoil pitch. Replaces **XRIGHT**.
JWAKEX	Exit axial grid stretching trigger. When JWAKEX = 1, points are exponentially clustered near the trailing edge (which keeps the grid density high for large downstream axial extents. JWAKEX = 0 prevents grid skewing for extremely short axial extents (e.g. embedded blade rows in a multistage machine). JWAKEX=1 is the default; if XDNFRC < 0.5, the GUI automatically switches JWAKEX to 0.

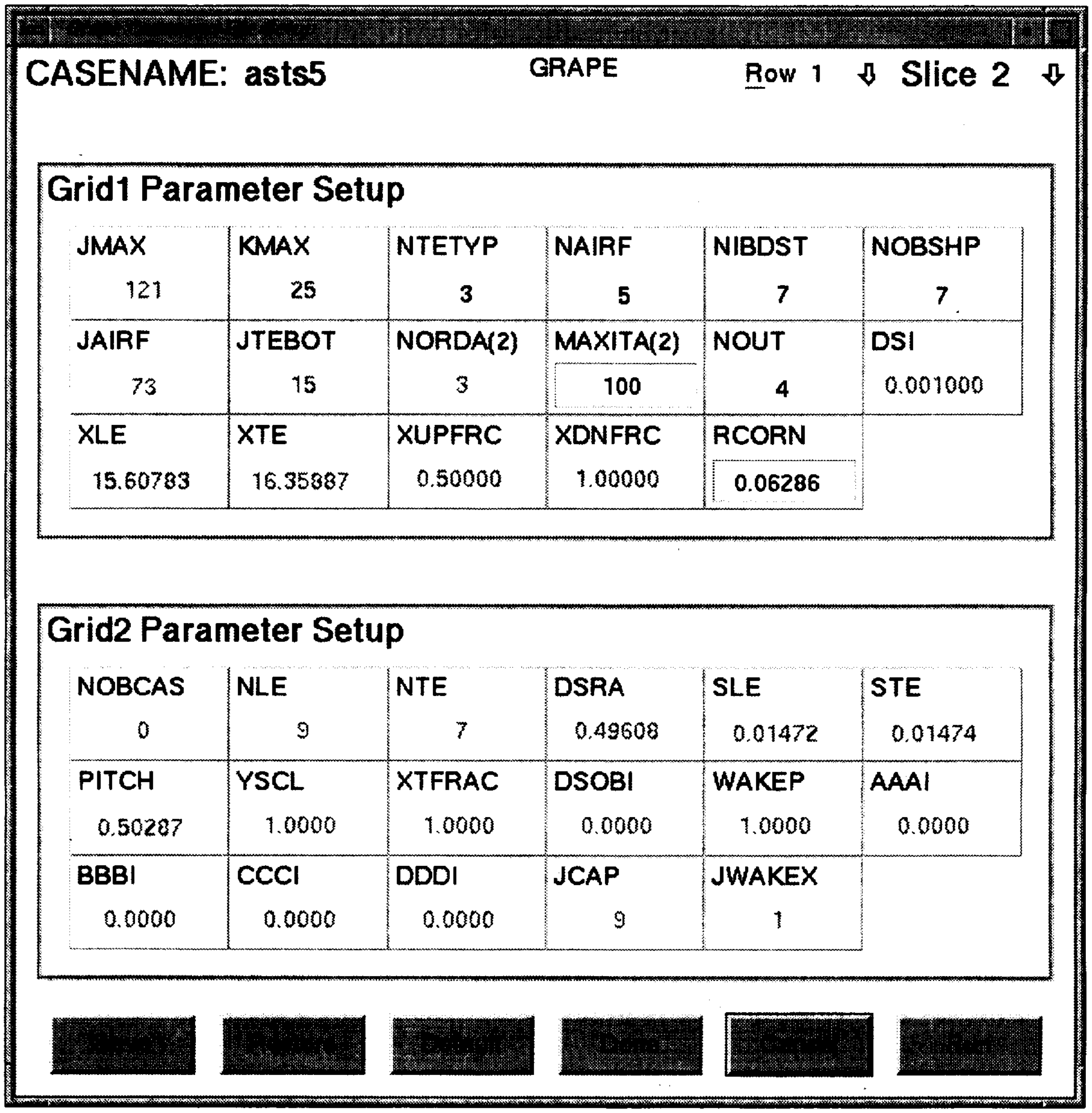
Table 9.4: The *GRAPE* input parameters were added/modified to enhance grid quality while reducing user effort.

9.1.4 *GRAPE* Input Panel

GRAPE is a slice dependent component module. It uses a standardized input panel as described in Section 9.1. Figure 9.5 shows how the controls are grouped to correspond to the namelist structure of the input as described in the *GRAPE* user's manual ([8, 4]). Some modifications were made to *GRAPE* input parameters as noted in Table 9.1.4. Also, the control labels correspond to their namelist counterparts. These features combined with the *GRAPE* documentation provide clear guidance for both experienced and inexperienced *GRAPE* users.

The *GRAPE* input panel will process only the namelist input parameters shown on the input panel. If the user wishes to input *GRAPE* namelist variables not shown on the panel he/she must input the data directly into the ASCII file before *GRAPE* is executed. Once this has been done, the *GRAPE* input panel must not be invoked. If it is, it will rewrite the ASCII namelist file.

Warning: This file, `casename.grape.in` (located in the individual slice subdirectories), is only read once by *TADS* during initialization. However, this file is written out for each slice every time the input panel is exited. The file which is read each time the input panel is displayed is the native binary file `casename.grape.db` in the working directory.



Grid1 Parameter Setup

JMAX 121	KMAX 25	NTETYP 3	NAIRF 5	NIBDST 7	NOBSHP 7
JAIRF 73	JTEBOT 15	NORDA(2) 3	MAXITA(2) 100	NOUT 4	DSI 0.001000
XLE 15.60783	XTE 16.35887	XUPFRC 0.50000	XDNFRC 1.00000	RCORN 0.06286	

Grid2 Parameter Setup

NOBCAS 0	NLE 9	NTE 7	DSRA 0.49608	SLE 0.01472	STE 0.01474
PITCH 0.50287	YSCL 1.0000	XTFRAC 1.0000	DSOBI 0.0000	WAKEP 1.0000	AAAI 0.0000
BBBI 0.0000	CCCI 0.0000	DDDI 0.0000	JCAP 9	JWAKEX 1	

Figure 9.5: *GRAPE* input panel controls the blade-to-blade grid generation.

9.1.5 *GRAPE for B2BADPAC* Input Panel

The *GRAPE for B2BADPAC* is another blade-to-blade grid generation option off of the main panel. It is not actually a different input panel, but it does have one important extra feature beyond *GRAPE* alone. Because *B2BADPAC* requires a full 3-D grid, *GRAPE* for *B2BADPAC* must run a stacking program called *b2badpac* after a full set of individual slice grids had been generated. Consequently, there is no option to generate a grid for single slice in this input panel. Also, if the grid generation for any slice in a given row has failed, the *B2BADPAC* grid cannot be generated and *b2badpac* will terminate abnormally.

The grid generated by *GRAPE* by *B2BADPAC* is a left handed 3-D cartesian mesh. It is written to the proper `casename.row.#` directory in which the subsequent *B2BADPAC* run will be executed.

An important time savings can be realized when running *GRAPE for B2BADPAC* and *B2BADPAC* for multiple *TADS* iterations. Unlike *RVCQ3D* computations where the radius and streamtube thickness for a given slice changes depending on the throughflow solution, the *B2BADPAC* grid never needs to be altered. All of the changes between *TADS* iterations is reflected in the *B2BADPAC* input and boundata files. Hence, once a *B2BADPAC* grid is generated, *GRAPE for B2BADPAC* never needs to be re-run (unless, of course, there is a change in geometry of the blade or the axial extents of the axisymmetric throughflow mesh).

9.1.6 *RVCQ3D* Input Panel

RVCQ3D is another standardized, slice dependent component module. Like *GRAPE*, the controls are grouped to correspond to the namelists structure of the input as described in the *RVCQ3D* documentation ([2]). Figure 9.6 shows how the control labels correspond to their namelist counterparts. Again, these features combined with the *RVCQ3D* documentation provide clear guidance for both experienced and inexperienced *RVCQ3D* users.

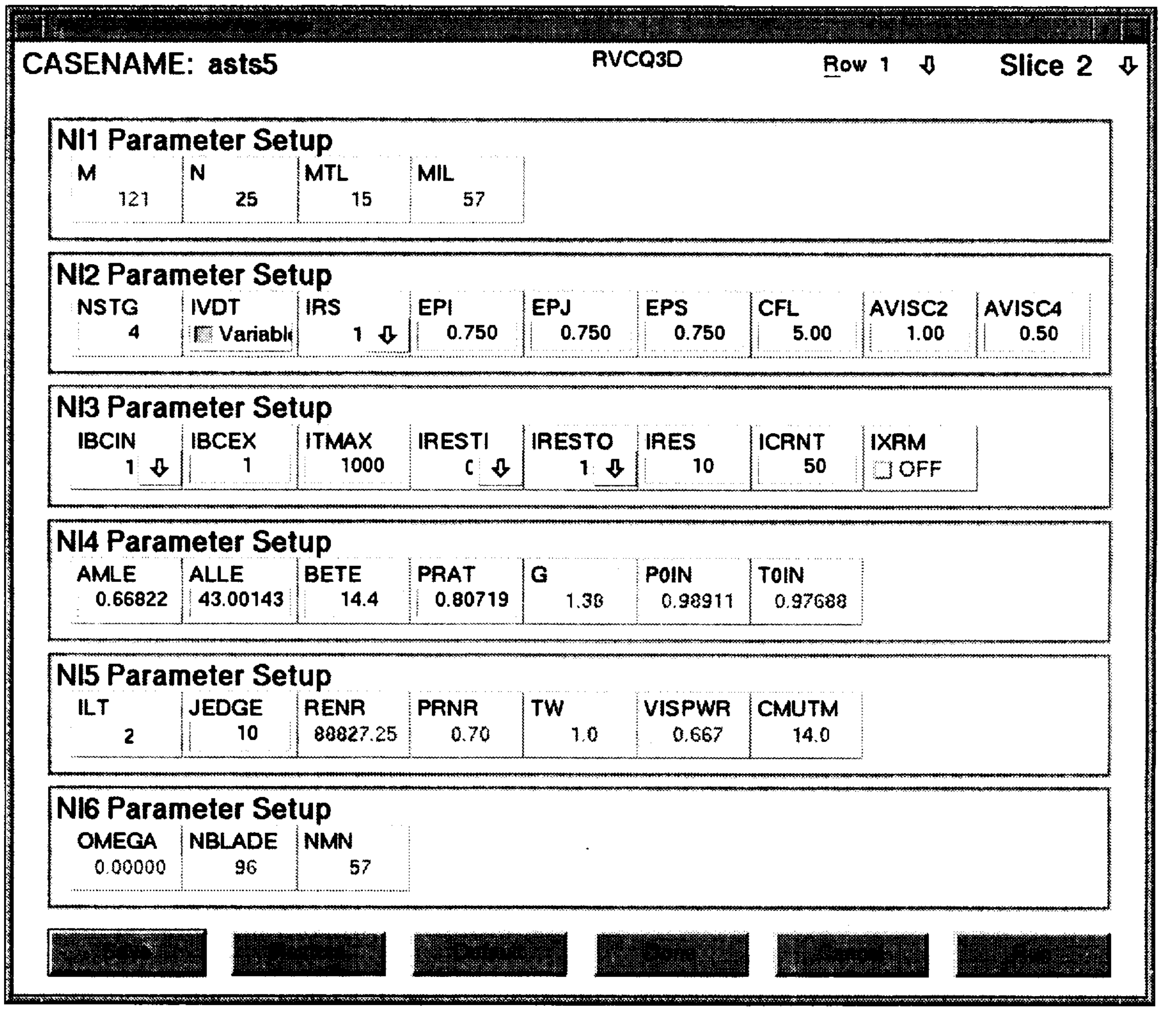
Again, just like *GRAPE*, the *RVCQ3D* input panel will process only the namelist input parameters shown on the input panel. If the user wishes to input *RVCQ3D* namelist variables not shown on the panel he/she must input the data directly into the ASCII file before *RVCQ3D* is executed. Once this has been done, the *RVCQ3D* input panel must not be invoked. If it is, it will rewrite the ASCII namelist file.

Warning: This file, `casename.rvcq3d.in` (located in the individual slice subdirectories), is only read once by *TADS* during initialization. However, this file is written out for each slice every time the input panel is exited. The file which is read each time the input panel is displayed is the native binary file `casename.rvcq3d.db` in the working directory.

9.1.7 *B2BADPAC* Data Input Panel

B2BADPAC is a slice dependent component module. It uses a standardized input panel shown in Figure 9.7. Like the throughflow *ADPAC* input panel, many of the control parameters on the *B2BADPAC* input panel correspond to the input keywords described in the *ADPAC* User's Manual [6].

B2BADPAC uses the grid from the *GRAPE for B2BADPAC* module, which is created by stacking the blade-to-blade grids for a given row. No slip conditions are applied



CASENAME: asts5 RVCQ3D Row 1 Slice 2

NI1 Parameter Setup

M	N	MTL	MIL
121	25	15	57

NI2 Parameter Setup

NSTG	IVDT	IRS	EPI	EPJ	EPS	CFL	AVISC2	AVISC4
4	Variabl	1	0.750	0.750	0.750	5.00	1.00	0.50

NI3 Parameter Setup

IBCIN	IBCEX	ITMAX	IRESTI	IRESTO	IRES	ICRNT	IXRM
1	1	1000	C	1	10	50	OFF

NI4 Parameter Setup

AMLE	ALLE	BETE	PRAT	G	P0IN	T0IN
0.66822	43.00143	14.4	0.80719	1.38	0.98911	0.97688

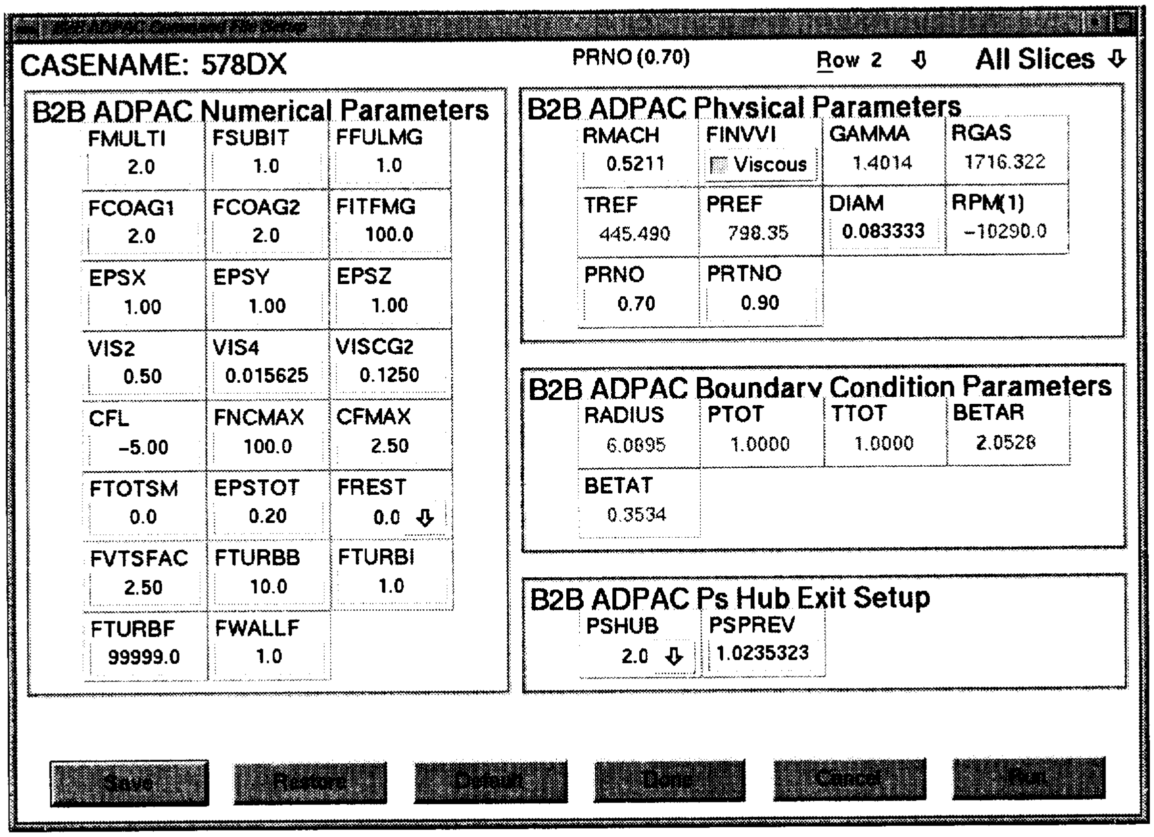
NI5 Parameter Setup

ILT	JEDGE	RENR	PRNR	TW	VISPWR	CMUTM
2	10	88827.25	0.70	1.0	0.667	14.0

NI6 Parameter Setup

OMEGA	NBLADE	NMN
0.00000	96	57

Figure 9.6: *RVCQ3D* input panel controls the blade-to-blade analysis.



CASENAME: 578DX PRNO (0.70) Row 2 All Slices

B2B ADPAC Numerical Parameters

FMULTI 2.0	FSUBIT 1.0	FFULMG 1.0
FCOAG1 2.0	FCOAG2 2.0	FITFMG 100.0
EPSX 1.00	EPSY 1.00	EPSZ 1.00
VIS2 0.50	VIS4 0.015625	VISCG2 0.1250
CFL -5.00	FNCMAX 100.0	CFMAX 2.50
FTOTSM 0.0	EPSTOT 0.20	FREST 0.0
FVTSFAC 2.50	FTURBB 10.0	FTURBI 1.0
FTURBF 99999.0	FWALLF 1.0	

B2B ADPAC Physical Parameters

RMACH 0.5211	FINVVI ☐ Viscous	GAMMA 1.4014	RGAS 1716.322
TREF 445.490	PREF 798.35	DIAM 0.083333	RPM(1) -10290.0
PRNO 0.70	PRTNO 0.90		

B2B ADPAC Boundary Condition Parameters

RADIUS 6.0895	PTOT 1.0000	TTOT 1.0000	BETAR 2.0528
BETAT 0.3534			

B2B ADPAC Ps Hub Exit Setup

PSHUB 2.0	PSPREV 1.0235323

Figure 9.7: *B2BADPAC* input panel is the second option for controlling the blade-to-blade analysis.

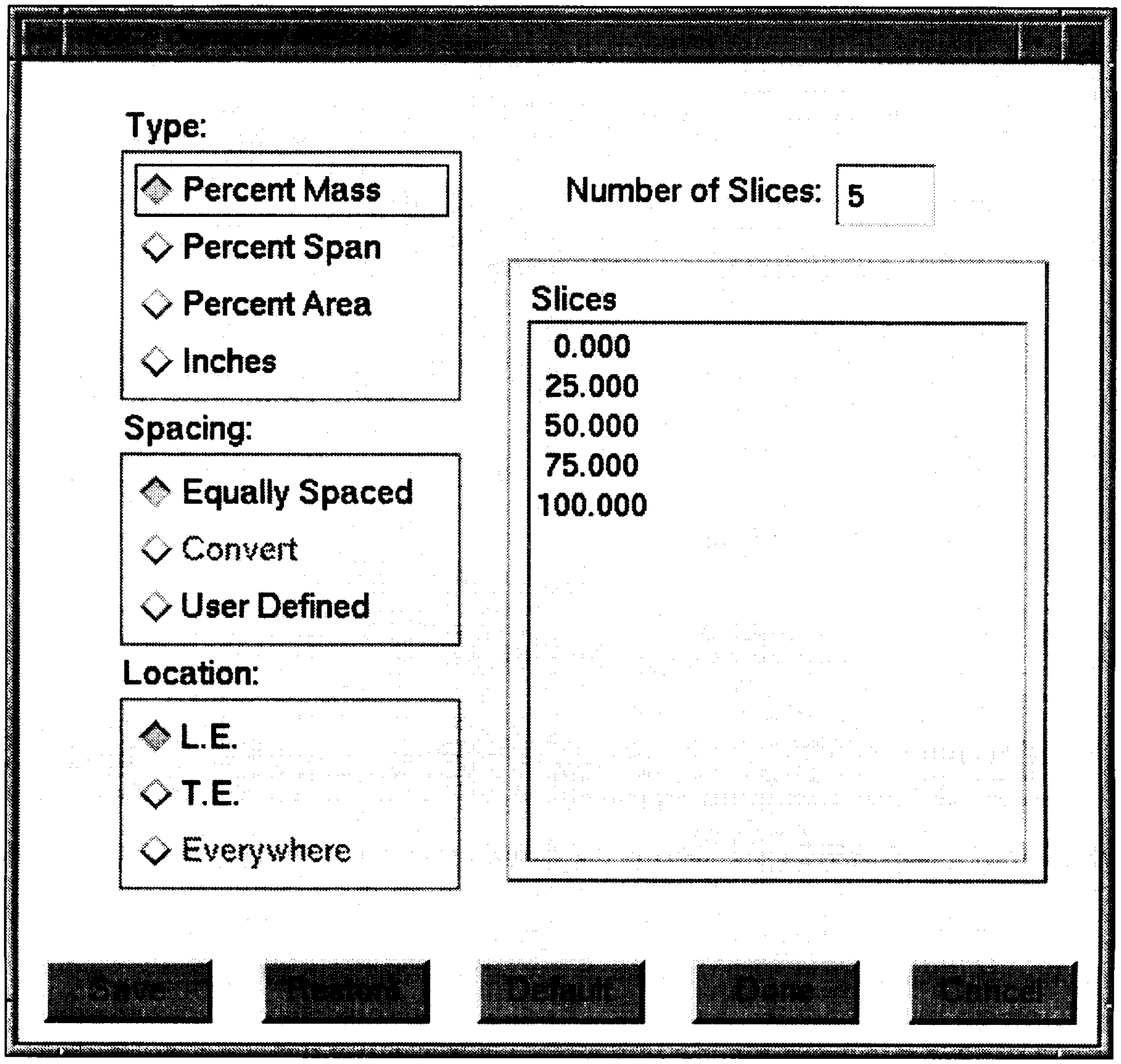
on the blade surfaces, while slip conditions are enforced on the endwalls.

When the *B2BADPAC* is invoked it reads in the slice geometry and aerodynamic information files. It uses these files to generate the boundary conditions shown in the Boundary Conditions group listings. The boundary condition parameter names correspond to the dummy text headers used in an **INLETT** type of boundary condition specification of an *ADPAC* `casename.boundata` file. This list is the only slice dependent portion of the *B2BADPAC* input panel.

Again, just like *GRAPE*, the *B2BADPAC* input panel will process only the input parameters shown on the input panel. If the user wishes to input *B2BADPAC* variables not shown on the panel he/she must input the data directly into the `casename.input` and `casename.boundata` files before *B2BADPAC* is executed. Once this has been done, the *B2BADPAC* input panel must not be invoked. If it is, it will rewrite the *ADPAC* ASCII input files.

9.2 SLICER Data Input Panel

The blade-to-blade analysis is performed along streamlines in the meridional plane as found by the throughflow analysis. This requires that the meridional streamlines be located in the throughflow solution, and that the airfoil be sliced along these streamlines. *TADS* uses two separate programs to accomplish this purpose: *radsl* and *slicer*. This



Type:
[✓ Percent Mass] Percent Span Percent Area Inches

Spacing:
[✓ Equally Spaced] Convert User Defined

Location:
[✓ L.E.] T.E. Everywhere

Number of Slices: 5

Slices
0.000
25.000
50.000
75.000
100.000

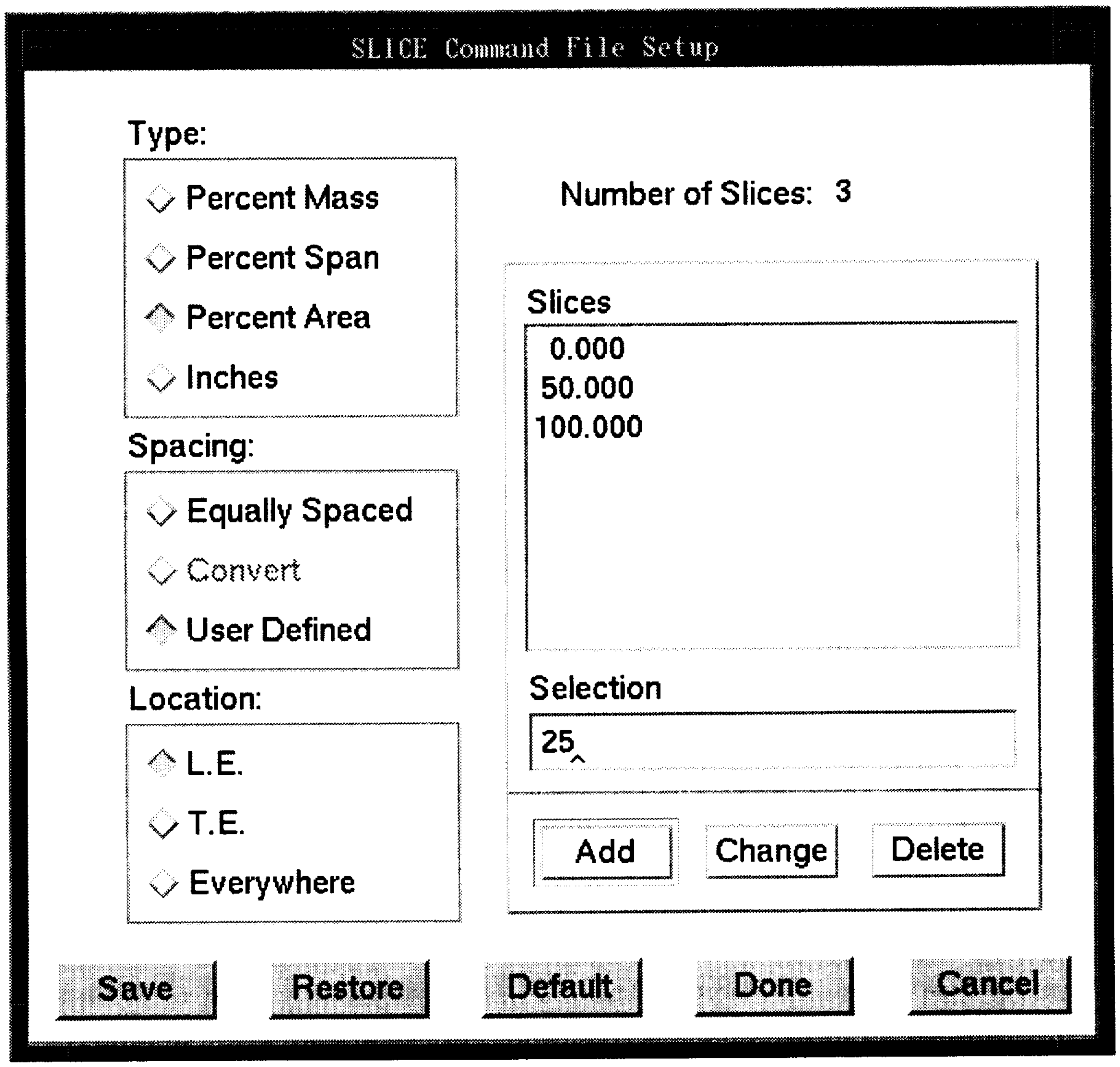
Figure 9.8: Slicer input panel controls the location of the 2-D analyses.

combination of programs is referenced as *SLICER* throughout this manual.

The input panel for *SLICER* is specialized (non-standard), primarily due to the non-scalar nature of the slice data. Another factor influencing the layout is the complex interactions between the individual controls.

Figure 9.8 shows the layout of this panel. To the left are three groups of radio buttons. These control the type of slices, the spacing of slices and the location of slices. The listbox to the right displays the values at which the slices will be made (calculated or specified depending on type/spacing/location).

When the spacing indicator is set to **Equally Spaced**, slice values are calculated based on the number of slices entered into the **Number of Slices** textbox. For this mode, the first slice is always the hub (0.0%) and the last slice is always the tip (100.0%), with the remaining slices being equally distributed. The **Number of Slices** textbox is only active when **Equally Spaced** is selected and becomes a label for other spacing modes. Currently, the maximum number of slices is set to eleven and the minimum


SLICE Command File Setup

Type: Percent Mass / Percent Span / [✓ Percent Area] / Inches

Spacing: Equally Spaced / Convert / [✓ User Defined]

Location: [✓ L.E.] / T.E. / Everywhere

Number of Slices: 3

Slices
0.000
50.000
100.000

Selection
25

Figure 9.9: User has the option to manually define slice geometry.

allowed is three.

When the spacing indicator is set to **User Defined**, the user has total control over the slice spacing and is allowed to add, delete or modify slices from a selection panel (see Figure 9.9). The user has the responsibility to insure that the values are valid and in the correct units (percent or inches). Here again, the user is limited to between three and eleven slices.

The **Convert** option under **Spacing** is not always available to the user. Under specific conditions this option is disabled to prevent confusion and inconsistent input. The most common of these conditions is if the axisymmetric grid file `casename.mesh` is not found (has not been generated). The **Convert** option, when available, will allow the conversion from one type to another and/or from one location to another. Note, converting from one "Percent" type to another has no visible effect on the displayed slice values, but will result in geometrically different slices.

Several combinations of type/spacing/location are disallowed by *TADS*. The conditions and reasons surrounding these situations are explained below. The clearest way to explain the type/spacing/location interactions is to describe how each combination is handled by the program.

Warning: In the multistage environment, the user must take extreme care when specifying different slice configurations. Using a different slice setting on adjacent rows can create discontinuities in the streamlines. This is usually not a problem in *TADS* because each row is handled separately. However in the large multi-row computations where the execution time for a full TADS iteration can be a prohibitive, any situation that hampers convergence is unwelcome. Therefore, it is recommended that a consistent slice definition be used on all blade rows.

9.2.1 Percent Mass Slice Mode

When the type indicator is **Percent Mass**, the generated slices are along streamlines at the values displayed in the listbox. These values are percentages of the total mass flow in the passage at either the leading or trailing edge station (**L.E.** and **T.E.**, respectively), whichever is specified by the location indicator. [3] The **Convert** option is not available in the **Percent Mass** mode. Conversely, the **Percent Mass** mode is not available when the **Convert** option is active.

9.2.2 Percent Span Slice Mode

When the type indicator is **Percent Span**, the generated slices are along streamlines which intersect the flowpath at the specified location [3] (i.e. leading or trailing edge) at the indicated percent span.

9.2.3 Percent Area Slice Mode

When the type indicator is **Percent Area**, the generated slices are along streamlines which intersect the flowpath at the specified location [3] at the indicated percent area. However, if the location is **Everywhere**, then the slices are not along streamlines, but are made at constant area locations along the flowpath's axial stations. This scenario would produce slices which are at the same percent area at both the leading and trailing edge locations (whereas streamlines always follow a constant percent mass).

9.2.4 Inches Slice Mode

When the type indicator is **Inches**, the generated slices are along streamlines which intersect the flowpath at the specified location [3] at the indicated distance (in inches) from the engine center line.

[3]The **Everywhere** location is only allowed for **Percent Area** slices and is not available when the **Percent Mass**, **Percent Span** or **Inches** type is selected.

Chapter 10

TADS Operating Instructions

This chapter contains some basic operating instructions for *TADS*. These instructions include general information covering source code compilation, resource configuration, program execution and trouble shooting. This chapter assumes that the *TADS* distribution has been extracted and placed in the "install" directory (`TADS.02`). Instructions for extracting the distribution can be found in Appendix I.

TADS was developed on a SGI Personal Iris operating under IRIX 4.0 (X11Rev.4). [1] It has been demonstrated on several different platforms including a SGI Indigo 2 and an IBM RS6000. The *TADS* system module was written primarily in C, but contains some FORTRAN 77 subroutines. These subroutines are input subroutines which have been stripped from the original program modules and modified for use in *TADS*.

10.1 Installing TADS

TADS has a UNIX compatible **make** facility for source code compilation. The `Makefile` which governs the compilation process is necessarily machine-dependent and complex. An installation shell script (*install_TADS*) is also provided to facilitate a proper installation. This script will prompt the installer for the necessary information needed to **make** *TADS* on his/her system "automatically".

To begin installation, it is first necessary to enter the install directory (`TADS.02`) with the command:

cd *"Path to* `TADS.02` *directory"*/`TADS.02`

The automated installation is performed by issuing the command:

install_TADS

A complete installation of *TADS* and its associated modules will require approximately 50 megabytes of disk space. This estimate assumes *TADS* is only being installed for a single platform. After installation is complete, the user may remove the object files by issuing the command:

[1] *TADS* has been ported to IRIX 5.3 (X11Rev.6).

cleanup_TADS

A description of the installing *TADS* manually can be found in Appendix J.

10.2 User Setup

Before *TADS* can be run, several operations must be performed to configure the user's system. First, the user's search path must be modified to include the path to *run_tads*. Secondly, *TADS* requires the environment variable `TADSDIR` be defined. Also, the user must provide an X resources file.

To add *run_tads* to the execution search path, enter the command:

setenv `PATH $PATH:` *"Path to* `TADS.02` *directory"*/`TADS.02 /apl`

The user must set the `TADSDIR` environment variable to the absolute path of the *TADS* install directory (`TADS.02`) with the command:

setenv `TADSDIR` *"Path to* `TADS.02` *directory"*/`TADS.02` .

TADS looks for X resources in a file named `.tads.rc`. This file must be in the user's home directory. To symbolically link a "standardized" resource file to the user's home directory execute the following command from the install directory:

install_user

The user can customize the *TADS* GUI environment by replacing the symbolic link with a user customized version of the resource file. Customizable features include fonts, colors, sizes and positions of some but not all GUI components. A sample resource file is found in Appendix H.

Executing the *install_user* command will also check to see if `TADSDIR` has been defined and will display the current value for the user to verify. Therefore, the `TADSDIR` environment variable should be set prior to running *install_user*.

10.3 Executing TADS

After *TADS* has been installed and configured it is recommended that the sample cases provided with the standard distribution be tested to verify proper installation. A discussion of the demonstration test case included with the distribution is given in Appendix K.

Once the *TADS* installation has been verified, the user is ready to run. Before running *TADS* , the user must place the required input files (see Table 4.1) in the working (current) directory. To execute *TADS* enter:

run_tads `-case casename[-help]`

run_tads is a shell script used to execute *tads*. If the user has not added the `TADSDIR /apl` directory to the execution search path, it can be copied or directly linked to the working directory as follows:

```
ln -s TADSDIR /apl/run_tads .
```

run_tads was designed to provide optional help on command line options and to provide preliminary input and configuration verification. The script determines the "local" machine type in order run the correct executable. Currently, there are no valid options (other than `-help`).

After an initialization delay the main panel will appear. A smaller panel will also appear and present text informing the user of program activity. This smaller window is the message panel and may be moved and/or resized by the user. Note, closing the message panel will not close the main panel and will not force *TADS* to terminate.

From the main panel the user has control over many operations. The order in which these operations are performed will vary from user to user, and from design to design. The scenario presented in this section is an attempt to reproduce a "typical" solution.

Typically, the user will first select the **Setup** action button to configure *TADS* for remote execution of component modules. Remember, the working directory must be NFS mounted on the remote machines for *TADS* to work properly. Once all the desired machines have been configured click **Done**. If all programs are to be executed "locally", this configuration step may be bypassed.

A logical second step is to select the **Edit Programs** mode. This allows the user to choose which programs *TADS* will use for the coupled analysis. The programs are selected from the pulldown lists located next to the component group controls (see Figure 5.1).

After selecting program modules, the user needs to inform *TADS* where the modules should run (e.g. on which remote machine). This is done by entering the **Edit Machines** mode and clicking on the desired choice from the appropriate pulldown menu. Be sure that any modules requiring GL graphics are run locally (local machine must have GL capability). GL programs may be run remotely, only if both the remote and local machines are SGI.

At this point, the user will normally enter the **Edit/Run** mode and proceed to execute the component modules one at a time until a complete analysis has been performed. Alternatively, the user may enter the **Edit Data** mode and attempt to setup all the input data before any programs are executed. While this approach can work, it is not the recommended method. The **Edit Data** mode is best reserved for fine tuning a solution after the initial pass has been completed. The exception to this rule is when running multi-slice modules. The **Edit Data** mode allows the user to execute a single slice analysis, whereas the **Edit/Run** mode will attempt to run the program module for all slices. For example, it is often desirable to generate a 2-D blade-to-blade grid for a single slice in order to evaluate the quality of the grid.

Once a single iteration has been completed, the user may switch to **Run** mode and continue the coupled analysis without any further user interaction. Each time **Run** is selected, *TADS* will execute another iteration. In many cases run during development, only one iteration was required to achieve an acceptable solution.

At any time, the user may exit (**Quit**) *TADS* from the main panel. If the user restarts *TADS* with the same casename, the previous configuration will be remembered (including active modes and programs). A *TADS* restart is transparent to the component modules as long as the related files are not altered between executions. Note, the previous configuration will be lost if the configuration file (`casename.configure`) is deleted before *TADS* is rerun.

10.4 Trouble Shooting TADS

If any problems are encountered at execution time check the following list for some possible solutions. The list is by no means complete, but is intended to deal with the problems most commonly encountered during development. The list is structured from the least to most aggressive, so try the items at the top of the list before moving on to more drastic measures. Any problems encountered in the independent program modules should be addressed by the appropriate author(s) and/or their documentation.

- Check to be sure the required input files are present (Table 4.1).
- Check to be sure *run_tads* is in search path.
- Check to be sure the DISPLAY environment variable has been defined.
- Check to be sure the `TADSDIR` environment variable has been defined.
- Check to be sure `.tads.rc` exists in the user's home directory.
- Remove all .db files from working directory (**`rm casename.*.db`**).
- Remove configuration file from working directory (**`rm casename.configure`**).
- Remove all multi-slice input files from slice subdirectories.
- Remove all non-required input files from working directory.

10.5 On Line Documentation

This user's manual, along with the final report, have been provided in an on-line format with this release of *TADS*. The documentation is in HTML (HyperText Markup Language) format and may be viewed with any HTML viewer/browser. "Hypertext" is text/graphics with pointers (links) to other text/graphics. It allows the user to access more information about a particular subject by "clicking" on it. Some of the more popular browsers are:

- NCSA Mosaic
- Netscape
- tkWWW
- Emacs (w3 mode)

A toplevel HTML file (referred to as a homepage) can be found in the `$TADS.02 /html` directory under the name `TADS.homepage.html`. As an example, to start NCSA Mosaic with this homepage the user would enter:

```
xmosaic -home $TADS.02 /html/TADS.homepage.html
```

References

[1] G.G. Adkins, L.H. Smith, "Spanwise Mixing in Axial-Flow Turbomachines," ASME Journal of Engr. for Power, Vol 104, pp 97-110.

[2] Chima, R., "Explicit Multigrid Algorithm for Quasi-Three-Dimensional Viscous Flows in Turbomachinery," Journal of Propulsion and Power, Vol. 3 No. 5, 1987.

[3] Chima, R., Turkel, E. Schaffer, S., "Comparison of Three Explicit Multigrid Methods for the Euler and Navier-Stokes Equations," NASA TM88878, Jan., 1987.

[4] Chima, R., "Revised GRAPE Code Input for Cascades," NASA Lewis, June, 1990.

[5] Damle, S., Dang, T., Reddy, D., "Throughflow Method for Turbomachines Applicable for all Flow Regimes," ASME Paper No. 95-GT-295.

[6] Hall, E., Topp, D., and Delaney, R., "Task 7 - ADPAC User's Manual" NASA CR195472, 1995.

[7] Miller, D., "TIGGERC - Turbomachinery Interactive Grid Generator for 2-D Grid Applications and Users Guide," NASA TM106586, 1994

[8] Sorenson, R., "A Computer Program to Generate Two-Dimensional Grids About Airfoils and Other Shapes by Use of Poisson's Equation," NASA TM81198, 1980.

[9] Turner, P., "ACE/gr: A XY plotting package for workstations or X-terminals running X", Copyright 1991-1994 Paul J Turner.

[10] Walatka, P., Buning, P., Pierce, L, Elson, P., "PLOT3D User's Manual, Version 3.6" NASA TM101067, 1990.

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

Appendix A

Complete List of Input and Output Files

The current working directory contains files and subdirectories. The subdirectories contain files associated with multi-slice modules.

The files in the current directory are listed below:

Name	*Format*	*Description*
`casename.adpac.input`	ASCII	*ADPAC* standard input file.
`casename.adpac.out`	ASCII	*ADPAC* standard output file.
`casename.bf.1`	*PLOT3D*	*ADPAC* 2-D blockage/body force file for block #1.
`casename.boundata`	ASCII	*ADPAC* block boundary definition file.
`casename.configure`	ASCII	Configuration information (machine name, manufacturer, executable path and data path) used to submit jobs to remote machines. It is created and maintained by the "Remote Processor Configuration" panel.
`casename.converge`	ASCII	*ADPAC* solution residual convergence history file.
`casename.forces`	ASCII	*ADPAC* output containing resultant forces and momentum on body.

continued on next page

continued from previous page

Name	*Format*	*Description*
`casename.grape.db`	binary	Database file used to manipulate *GRAPE* input data for all slices.
`casename.meansl`	*PLOT3D*	*MEANSL* output file containing mean stream surface.
`casename.mesh`	*PLOT3D*	*ADPAC* mesh file (*PLOT3D* compatible).
`casename.p3dabs`	*PLOT3D*	*ADPAC PLOT3D* output file (absolute flow).
`casename.pshub`	ASCII	Static pressure data file used to store *ADPAC* back pressure values between *adpacbc* and the *ADPAC* input GUI panel.
`casename.restart.new`	*PLOT3D*	New *ADPAC* restart file (output by *ADPAC*).
`casename.restart.old`	*PLOT3D*	*ADPAC* restart file (used as input for *ADPAC* restart runs).
`casename.rvcq3d.db`	binary	Database file used to manipulate *RVCQ3D* input data for all slices.
`casename.rvt.1`	ASCII	*ADPAC* 2-D rV_θ file for block #1
`casename.rvtdesign`	ASCII	aerodynamic design file for *PREDESIGN*
`casename.slcaro`	ASCII	*SLICER* output file containing aerodynamic information for meridional streamline interpolation from `casename.tdsaro`.
`casename.slice_data`	ASCII	*SLICER* input file containing slice location, type and spacing information.
`casename.stkq`	*PLOT3D*	*RESTACK* output *PLOT3D* "Q" file of stacked 2-D solutions.
`casename.stkx`	*PLOT3D*	*RESTACK* output *PLOT3D* "X" file of stacked 2-D solutions.

continued on next page

continued from previous page

Name	*Format*	*Description*
`casename.tdsaro`	ASCII	Aerodynamic information at the airfoil leading and trailing edges and the airfoil tangency point indices. A sample file can be found in Appendix tdsaro.
`casename.tdsasl`	*PLOT3D*	*RADSL* output *PLOT3D* "X" file of meridional streamlines.
`casename.tdsasq`	*PLOT3D*	*RADSL* output *PLOT3D* "Q" file of meridional streamlines.
`casename.tdsaxi`	ASCII	*intigg* input file containing axisymmetric grid parameters.
`casename.tdsblad`	*PLOT3D*	3-D Cartesian airfoil surface defined by two parameters, one clockwise around the airfoil, and the other along the span.
`casename.tdsbsl`	*PLOT3D*	*SLICER* output file containing airfoil sliced along meridional streamlines.
`casename.tdspath`	ASCII	Meridional flowpath definition, consisting of two line in the *(X,R)* plane. A sample file can be found in Appendix tdspath.
`casename.tiggin`	ASCII	*TIGG* input file
`casename.tpl.1`	ASCII	*ADPAC* total pressure loss coefficient data file for block #1
`tds_casename`	ASCII	Text file which contains current case name - this file is used by the Fortran programs to construct file names
`casename.row.#`	Directory	Subdirectory name where # is a row number.
`casename.sl.#`	Directory	Subdirectory name under each `casename.row.#` directory where # is a slice number.

The files found in a representative slice subdirectory are listed below:

Name	*Format*	*Description*
`casename.grape.in`	ASCII	*GRAPE* namelist input file
`casename.grape.out`	ASCII	*GRAPE* output file
`casename.rvcq3d.in`	ASCII	*RVCQ3D* namelist input file
`casename.rvcq3d.out`	ASCII	*RVCQ3D* output file
`grid.bin`	*PLOT3D*	*GRAPE* 2-D, single grid, SDB binary output file used for *PLOT3D* post-processing and as input to *RVCQ3D*
`restout.bin`	*PLOT3D*	*RVCQ3D* 2-D, relative, single grid, SDB binary output file used for *PLOT3D* post-processing and *RVCQ3D* restarting

Appendix B

Sample Flowpath Description Input File

```
Flowpath data: Hub profile, ihub followed by x,r pairs
         9
    0.1450100040E+02    0.7461999893E+01
    0.1467599964E+02    0.7465000153E+01
    0.1543200016E+02    0.7474999905E+01
    0.1560700035E+02    0.7478000164E+01
    0.1636400032E+02    0.7482999802E+01
    0.1654400063E+02    0.7485000134E+01
    0.1712100029E+02    0.7489999771E+01
    0.1730100060E+02    0.7491000175E+01
    0.1796599960E+02    0.7499000072E+01
Flowpath data: Tip profile, itip followed by x,r pairs
         9
    0.1452400017E+02    0.8392000198E+01
    0.1471399975E+02    0.8376000404E+01
    0.1538700008E+02    0.8321000099E+01
    0.1559300041E+02    0.8305999756E+01
    0.1637500000E+02    0.8250000000E+01
    0.1656500053E+02    0.8237999916E+01
    0.1709600067E+02    0.8206000328E+01
    0.1729800034E+02    0.8194000244E+01
    0.1796699905E+02    0.8159999847E+01
```

Appendix C

Sample Aerodynamic Data Input File

```
Aerodynamic Information File
Total number of blade rows
        1
Blade Row  1
------------
Machine Type:
        =0  axial machine; =1  centrifugal compressor; =2  radial turbine
        0
Number of slices for which there is aerodynamic information
       11
```

Leading Edge Data

Radius	Total Pressure	Total Temperature	Axial Location
7.500540	155.481003	1089.040039	15.606600
7.510270	155.369003	1087.050049	15.606400
7.536040	155.097000	1082.189941	15.606000
7.577550	154.707993	1076.079956	15.605300
7.634360	154.272995	1069.250000	15.604300
7.706000	153.865997	1063.020020	15.603100
7.792050	153.557007	1058.569946	15.601600
7.892210	153.412994	1059.229980	15.599800
8.006500	153.533005	1068.270020	15.597900
8.135540	154.059998	1091.260010	15.595600
8.281160	155.212006	1140.939941	15.593100

Radius	Axial Mach Number	Tangential Mach	Radial Mach
7.500540	0.525909	0.526155	-0.002270
7.510270	0.526044	0.524035	-0.002562
7.536040	0.526363	0.518744	-0.003331
7.577550	0.526477	0.511143	-0.004553
7.634360	0.526305	0.502242	-0.006201
7.706000	0.525663	0.493186	-0.008242
7.792050	0.524440	0.485145	-0.010649
7.892210	0.522145	0.479530	-0.013377
8.006500	0.518735	0.478035	-0.016400
8.135540	0.513141	0.483239	-0.019663
8.281160	0.503329	0.499569	-0.023093

Trailing Edge Data

Radius	Static Pressure	Total Temperature	Axial Location
7.507200	126.592361	1089.040039	16.364599
7.516120	126.589859	1087.050049	16.364799
7.539790	126.579399	1082.189941	16.365101
7.577980	126.571007	1076.079956	16.365601
7.630370	126.558014	1069.250000	16.366301
7.696580	126.549057	1063.020020	16.367201
7.776210	126.549652	1058.569946	16.368299
7.868980	126.590836	1059.229980	16.369499
7.974810	126.666481	1068.270020	16.371000
8.094000	126.811035	1091.260010	16.372601
8.227530	126.980896	1140.939941	16.374399

Radius	Axial Mach Number	Tangential Mach	Radial Mach
7.507200	0.502212	0.134567	-0.001697
7.516120	0.502219	0.134569	-0.002021
7.539790	0.502324	0.134597	-0.002871
7.577980	0.502365	0.134608	-0.004222
7.630370	0.502441	0.134629	-0.006038
7.696580	0.502487	0.134641	-0.008295
7.776210	0.502454	0.134632	-0.010967
7.868980	0.501993	0.134509	-0.014056
7.974810	0.501220	0.134301	-0.017607
8.094000	0.499827	0.133928	-0.021717
8.227530	0.498471	0.133565	-0.026673

```
 Thermodynamic Information
    Gamma        Gas Constant
 1.376945           53.345001
 Physical Properties
Wheel RPM       Tip Clearance   Number of Blades
 0.000000           0.000000           96.000000
 Tangency Points
     itnsl      itnst      itnpt      itnpl
         4         33         40         69
 Blade Row  2
 ------------
 Machine Type:
         =0  axial machine; =1  centrifugal compressor; =2  radial turbine
         0
 Number of slices for which there is aerodynamic information
        11
 etc....
```

Appendix D

Sample rV_θ Design File

Please note: Currently, *TADS* only reads in the first two numbers after the keyword for block identification purposes. The remaining items have been written in an effort to mimic the *ADPAC* boundary data file (in anticipation of any future design mode work in *ADPAC*).

```
RVTHETA   1  1  I  I  M  M  L  L   0  0  1  17   1   2   1  17   1   2  rVt1
NDATA
          11
      RAD          RVTHETA (ft**2/s)   AXIAL LOCATION
     5.962250  282.012878    -5.590770
     6.583190  293.013306    -5.693230
     7.189030  306.572113    -5.793210
     7.781120  321.585937    -5.890920
     8.361050  337.451874    -5.986620
     8.930420  353.716888    -6.080570
     9.490850  370.017914    -6.173040
    10.044000  386.021118    -6.264310
    10.591500  401.398865    -6.354650
    11.135000  415.830658    -6.444340
    11.676300  429.092072    -6.533660
```

```
NDATA
        11
     RAD         RVTHETA (ft**2/s)   AXIAL LOCATION
    6.106590  289.621857   -4.298120
    6.714930  312.959625   -4.261340
    7.295110  337.111603   -4.226270
    7.852560  359.662750   -4.192570
    8.393350  381.564880   -4.159870
    8.920380  403.251801   -4.128010
    9.435910  424.641724   -4.096840
    9.941870  445.735657   -4.066250
   10.441300  463.130463   -4.036050
   10.940100  477.473236   -4.005900
   11.455300  465.235321   -3.974740
AXRVTDIST    1  1   K  K  M  M  I  J  2  2   0  0  1  17   0  0  1  17  ax dist
NDATA
        11
     RAD         POWER
    5.962250    1.000000
    6.583190    1.000000
    7.189030    1.000000
    7.781120    1.000000
    8.361050    1.000000
    8.930420    1.000000
    9.490850    1.000000
   10.044000    1.000000
   10.591500    1.000000
   11.135000    1.000000
   11.676300    1.000000
```

Appendix E

rV_θ File Description

The *ADPAC* rV_θ file is a data file containing aerodynamic turning information for a 2-D axisymmetric design mode calculation. Each rV_θ file name has a index extension that indicates which block the data refers to. As a result, the naming procedure for the rV_θ file is more like an *ADPAC* body force file than a mesh or restart file.

In order to understand the body force file format, a representative FORTRAN coding example to read in a body force file is given below for comparison.

rV_θ File Format FORTRAN Coding Example

```
OPEN(IRVT,FILE=RVTFILE(N),STATUS='OLD')
READ(IRVT,'(I10)') NG
READ(IRVT,'(3I10)') IMX,JMX,KL
READ(IRVT,'(4E15.8)') DUMMY,DUMMY,DUMMY,DUMMY
READ(IRVT,'(5E15.8)') ((RVTIMP(I,J),I=1,IMX),J=1,JMX)
CLOSE(IRVT)
```

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

IRVT	FORTRAN logical unit number for rV_θ file input
NG	Number of blocks in rV_θ file (must be 1)
IMX	Mesh size+1 in the i coordinate direction
JMX	Mesh size+1 in the j coordinate direction
KL	Mesh size in the k coordinate direction
RVTIMP	Imposed rV_θ distribution. Product of cell centered radius and tangential velocity

Appendix F

Sample Total Pressure Loss Data Input File

```
Aerodynamic Loss Information File
Loss type (1=TPL comp, 2=TPL turb 3=deltaPt/Pt1)
        1
Number of radial location for loss values at trailing edge
       11
            Radius          Loss Value      Axial Location
          7.507200           0.077144           16.364599
          7.516120           0.075055           16.364799
          7.539790           0.069843           16.365101
          7.577980           0.062353           16.365601
          7.630370           0.053803           16.366301
          7.696580           0.045599           16.367201
          7.776210           0.039261           16.368299
          7.868980           0.036490           16.369499
          7.974810           0.039603           16.371000
          8.094000           0.051804           16.372601
          8.227530           0.077228           16.374399
Starting and Ending Loss Indices (0 defaults to Leading and Trailing Edge)
        0         0
```

Appendix G

New ADPAC Input Parameters

DFACTC(*NUM*)

(Default Value = 0.0)

```
DFACTC(1) = 0.0
```

The **DFACTC** keyword is used to set the diffusion factor curve number used in the calculation of the total pressure loss coefficient. The specified diffusion factor curve, blade geometry, and current flow variables are used to update the magnitude of the total pressure loss coefficient applied along streamlines in a solution. The value of *NUM* specifies the mesh block on which the diffusion factor is applied. Currently, only one diffusion factor has been hard-coded into *ADPAC* (corresponding to **DFACTC(1) = 1.0**). Any remaining diffusion factor relations are proprietary and therefore must be supplied by the user. A value of 0.0 means that the diffusion factor relations will not be used and total pressure loss coefficient values will remain constant as specified in the file set by the **TPLFILE(*NUM*)** keyword.

FBFMETH

(Default Value = 0.0)

```
FBFMETH = 0.0
```

The **FBFMETH** keyword determines which calculation method is used to update blade body force during an 2-D axisymmetric run. The default value, 0.0, enables the "direct method" for calculating body forces, while the alternate value, 1.0, uses the "relaxation method" for updating body forces. The direct method has the benefits of extremely fast convergence and a rigorous theoretical basis for it implementation into the 2-D axisymmetric solver. Because of numerical issues, however, it is not applicable to all cases. The relaxation method is a slower converging and less elegant approach to updating body forces, but it is applicable to all cases. Because of these considerations, this key is often the first item to examine when diagnosing a failed or poorly converging solution.

NOTE: When using the throughflow loss model (activated by keys **FLOSSB, FLOSSE**, and **FLOSSI**), **FBFMETH MUST BE** set to 1.0.

FBFRLX

(Default Value = 1.0)

```
FBFRLX = 1.0
```

The **FBFRLX** keyword controls the amount of relaxation in the updating of the body forces in the "relaxation method" (**FBFMETH** = 1.0). During each iteration in the calculation, the updated body force is the sum of the old body force and the product of **FBFRLX** and the difference of the current and desired V_θ (the desired V_θ is based on the mesh θ). While a value of 1.0 is sufficient for most cases, lower values (0.1-0.5) are often required for supersonic flows and cases with large turning and significant flow gradients.

FCAMBB

(Default Value = 99998.0)

```
FCAMBB = 99998.0
```

The *FCAMBB* keyword assigns a trigger which determines the number of iterations before the mesh geometry (the blade mean camber line) is altered in order to match the imposed rV_{theta} values. This trigger is only relevant if **FDESIGN** = 2.0. *ADPAC* sets this keyword's default high so that camber changes are not introduced inadvertently. A good starting value for *FCAMBB* is between 5.0 and 20.0. This interval gives the solution time to set up reasonably good initial body force values before changes in the mesh are made.

FCAMBE

(Default Value = 99999.0)

```
FCAMBE = 99999.0
```

The *FCAMBE* keyword assigns a trigger which determines the number of iterations before the mesh geometry (the blade mean camber line) ceases to be altered. This keyword is only relevant if **FDESIGN** = 2.0. This trigger was added so that the user can eliminate the computational expense of updating the camber when the updating is producing essentially no benefit to the quality of the solution. This can often happen when the change in geometry creates a minor change in the flow. This flow change then prompts a geometry change back to the previous mesh shape. This pattern then repeats itself indefinitely in a manner similar to the cyclical nature of a shed vortex. While this cyclical response is rarely unstable, it can cause poor convergence. If the problem of cyclical convergence is not encountered, *FCAMBE* can generally be kept at its large, default value.

FCAMBI

(Default Value = 1.0)

```
FCAMBI = 1.0
```

The *FCAMBE* keyword assigns a trigger which determines the number of iterations between changes in the mesh geometry (the blade mean camber line). This keyword is only relevant if **FDESIGN** = 2.0. For best results, this value should be kept below 5.0. When using the relaxation method (*FBFMETH* = 1.0), however, a large interval (*FCAMBE* = 10.0-15.0) between mesh changes is often necessary. A large iteration interval is needed because the relaxation method requires several iterations to update the body force for the current mesh shape before a new one is generated. If *FCAMBE* is too low, there is a risk that body force updating cannot "catch up" with the changing geometry and the solution will become unstable.

FDESRLX

(Default Value = 1.0)

```
FDESRLX = 1.0
```

The **FDESRLX** keyword determines the amount of relaxation when changing the mesh geometry (the blade mean camber line). The scheme for altering the mesh geometry is a relaxation method and as such it can update the blade camber too quickly, leading to solution instability. During early development of the design mode, it was found that a hard-coded value of 1.0 for **FDESRLX** did not provide sufficient under relaxation for every case. This was particularly true if the starting mesh was a poor initial guess. For design mode solutions with poor convergence, a **FDESRLX** value of 0.1-0.5 can often alleviate problems.

FLOSSB

(Default Value = 99998.0)

```
FLOSSB = 99998.0
```

The *FLOSSB* keyword assigns a trigger which determines the number of iterations before the throughflow loss model is activated. The default value for *FLOSSB* is set to be very large so that the loss model is not activated inadvertently. When the loss model is being used, this value should be set to reasonably high value (e.g. ≥ 10.0) so that the solution has developed adequately enough to ensure stable operation. In order for the loss model to function properly, a total pressure loss coefficient file must be present or a diffusion factor curve must be specified.

FLOSSE

(Default Value = 99999.0)

```
FLOSSE = 99999.0
```

The *FLOSSE* keyword assigns a trigger which determines the number of iterations before the throughflow loss model is deactivated. This trigger was added so that the user can eliminate the computational expense of the throughflow loss model evaluations when those evaluations are producing essentially no benefit to the quality of the solution. Generally, this value can be kept at its large, default value

FLOSSI

(Default Value = 1.0)

```
FLOSSI = 1.0
```

The *FLOSSI* keyword assigns a trigger which determines the number of iterations between throughflow loss model evaluations. For best results, this value should be 1.0 which implies that the source terms for the throughflow loss model are updated every iteration. However, this value can be increased to reduce CPU overhead by reevaluating the source terms for the throughflow loss model every *FLOSSI* iterations (at the possible expense of irregular convergence).

RVTFILE(*NUM*)

(Default Value = default_file_name)

```
RVTFILE(1) = casename.rvt.1
```

The **RVTFILE** keyword value determines the name of the file used to read in the rV_θ distribution for the *ADPAC* 2-D axisymmetric design mode. The value of *NUM* in the keyword definition specifies the mesh block that rV_θ distribution applies to.

TPLFILE(*NUM*)

(Default Value = default_file_name)

```
TPLFILE(1) = casename.tpl.1
```

The **TPLFILE** keyword value determines the name of the file used to read in the desired total pressure loss coefficient values for the throughflow loss model. The value of *NUM* in the keyword definition specifies the mesh block that the radial profile of total pressure loss coefficients applies to.

WBF(*NUM*)

(Default Value = 0.0)

```
WBF(1) = 0.0
```

The **WBF** keyword value is a trigger that determines whether or not to write out the final body forces at the end of a 2-D axisymmetric run. The value of *NUM* in the keyword definition specifies the mesh block for which a body force file will be written. A default value of 0.0 means that no body force file will be written, while a value of 1.0 triggers the writing of body forces at the last iteration of a computation.

Although intended as more a means of representing a converged 3-D flow with a 2-D solution and body forces, this keyword (with a value of 1.0) provides body forces values which can be used to restart the solution. Although 2-D solutions are generally so short that they do not require restarting, recent experience with large multistage solutions indicates that having restart capability can be important.

Appendix H

Sample X Resource File

```
##
! ***********  TADS IBM RS6000 Resource file ************

*shell.height:                                   600
*shell.width:                                    650
*background:                                   beige

*case_title.shadowThickness:                       0
*title_bar.shadowThickness:                        0
*title_bar.Alignment:        alignment_beginning
*con_status_bar.Alignment:   alignment_beginning

*menu.background:                         light grey
*menu.height:                                     40
*menu.width:                                     100
*menu_group.shadowThickness:                       0

*dec_frame.shadowThickness:                        5
*dec_btn.shadowThickness:                          3
*dec_btn.background:                            grey
*dec_lbl.background:                      light grey
*lbl_frame.shadowThickness:                        3
*lbl_frame.background:                    light grey

*pushb_quit.background:                          red
*pushb_quit.height:                               40
*pushb_quit.width:                               100

*pushb_csh.background:                        yellow
*pushb_csh.height:                                40
*pushb_csh.width:                                100

*pushb_run.background:                         green
*pushb_run.*bottomShadowColor:                 black
```

```
*pushb_run.*topShadowColor:                 black

*radio_box.x:                                  50
*radio_box.y:                                 120
*radio_box.shadowThickness:                     3
*radio_box.background:                       gray
*radio_btn.shadowThickness:                     2

*list_tb.Alignment:           alignment_beginning
*list_label.Alignment:        alignment_beginning
*list_pd.shadowThickness:                       3
*list_cell.shadowThickness:                     1

*input_button.background:                 skyblue
*input_bb.shadowThickness:                      0
*input_button.height:                          40
*input_button.shadowThickness:                  3

*slc_frame.shadowThickness:                     3
*slice_pd.shadowThickness:                      1
*slice_pd.Alignment:             alignment_center
*cascade_label.Alignment:        alignment_center

! *************** fonts **********************
*fontList:                      helvR14
*case_title.fontList:           helvB18
*title_bar.fontList:            helvB18
*menu.fontList:                 helvR14
*dec_form_lbl.fontList:         helvB14
*dec_tgl.fontList:              helvB14
*dec_btn.fontList:              helvB14
*dec_lbl.fontList:              helvB14
*pushb_quit.fontList:           helvB18
*pushb_csh.fontList:            helvB18
*pushb_run.fontList:            helvB18
*radio_btn.fontList:            helvB14
*list_tb.fontList:              helvR14
*list_label.fontList:           helvR14
*list_pd.fontList:              helvR14
*list_box.fontList:             helvR14
*input_button.fontList:         helvR14
*slc_list_box.fontList:         helvR14
*slc_frame_label.fontList:      helvB14
*slice_pd.fontList:             helvB18
*slice_label.fontList:          helvB18
```

Appendix I

Extracting the Source Files

This appendix describes the commands necessary to extract the source code and demo files from the *TADS* standard distribution.

The standard *TADS* 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, but not including installation and configuration. The command sequences listed below should work on most systems employing the UNIX operating system.

The *TADS* programs are distributed as a compressed **tar** file named

```
TADS.02.tar.Z
```

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 *TADS* programs is to **uncompress** the **tar** file with the command:

```
uncompress TADS.02.tar.Z
```

This operation essentially replaces the compressed file `TADS.02.tar.Z` with an uncompressed file `TADS.02.tar` .

The next step is to extract the individual files and directories from the `TADS.02.tar` file. Before this is done, the user must put the `TADS.02.tar` file in a suitable location. Once the **tar** file is properly placed, the *TADS* distribution may be extracted with the command:

```
tar xvof TADS.02.tar
```

(Note, on some systems **tar xvf** `TADS.02.tar` may be sufficient.)

Execution of the UNIX list command **ls** will verify that the `TADS.02` directory has been created. The **tar** command will have created a top level directory named `TADS.02` in the current directory. The `TADS.02` directory is referred to as the install directory.

The **uncompress** and **tar** steps can be combined in a single operation on most UNIX systems by issuing the command

```
zcat TADS.02.tar.Z | tar xvf -
```

This combined operation conserves overall disk space requirements during the extraction process.

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

```
.tads.rc.aix   .tads.rc.sgi
Post/          csdb/          guilib/        modules/
PreDesign/     doc/           install_TADS*  plotlib/
TOOLS/         examples/      install_user*  sdb/
apl/           gui/           misc/
```

A description of each of these listings is given below:

`.tads.rc.aix`	X resource file for IBM RS6000 workstations.
`.tads.rc.sgi`	X resource file for Silicon Graphics workstations.
`Post`	Directory containing the sources for the *TADS* post processor.
`PreDesign`	Directory containing the sources for the *TADS* design mode pre-processor.
`TOOLS`	Directory containing utility programs and scripts used for development and installation.
`apl`	Directory containing shell scripts and symbolic links to *TADS* component module executables.
cleanup_TADS	Shell script to remove all the object files created when *TADS* is installed.
`csdb`	Directory containing the Allison developed C version of the SDB library.
`examples`	Directory containing demonstration test cases.
`gui`	Directory containing the source for the *TADS* GUI.
`guilib`	Directory containing the source for the *TADS* GUI library routines.
`plotlib`	Directory containing the sources for the *POST* and *PREDESIGN* plotting library routines.
`html`	Directory containing the HTML versions of this manual and the final report.
install_TADS	Shell script to install the *TADS* GUI and all of the associated component modules.
install_user	Shell script to link X resource file into users home directory.
`misc`	Directory containing development programs not required by *TADS* , but developed under the contract.
`modules`	Directory containing the source code for *TADS* component modules.
`sdb`	Directory containing the NASA developed SDB library.

Appendix J

Compiling TADS Components

This appendix describes the commands necessary to compile the GUI and it's associated modules for the *TADS* standard distribution.

The command sequences 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 the development platform described in Table J.1. Alternate commands will be listed when a significant machine dependence exists.

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 the GUI and its associated library and also for each of the *TADS* component modules. 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. 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 appropriate directory (for example **cd** `$TADSDIR`/gui). It is now possible to compile the module by issuing the command:

make

Compilation options are available by typing **make help**. For example, on an IBM RS6000 workstation, the command **make aix** is the appropriate command.

Table J.1: *TADS* development platform software configuration.

- IRIX Operating System, Revision 4.0.1
- SGI Fortran 77, 3.10
- SGI Ansi C, 3.10
- Motif Development System, 4.0.5
- X11 Rev.4

Appendix K

Running the Distribution Demonstration Test Case

After *TADS* has been properly installed and configured (see Section 10.1 or Appendix J), it is possible to run the demonstration test case provided with the standard distribution. It is recommended that the sample case be tested to verify proper compilation and extraction of the *TADS* distribution.

In order to run the demonstration case, it is necessary to begin in the **`examples`** directory. This directory is located in the install directory and is entered by issuing the command:

```
cd $TADSDIR/examples
```

After entering the **`examples`** directory, the **ls** command will indicate that the following subdirectories (and possibly others) are available:

```
AST/      578DX/   Purdue_Turbine/   Rotor37/    Rotor67/
```

There are several test cases in each one of the these directories. The general format for each case is two subdirectories as follows:

```
Required_Files/       Try/
```

Both subdirectories contain identical required input files (see Table 7.4) for a given case. The **`Required_Files`** directory contains only these required input files. The **`Try`** directory contains these input files and all the files present after one pass through the *TADS* system. Having both a before and after case allows the user to compare results from a *TADS* run.

WARNING: The user should **NOT** run from the **`Try`** directory because the configuration and database files are specific to the machine and directory structure that the case was originally run on. Also, the user should **NOT** run from the **`Required_Files`** directory. An uncorrupted directory of required files should be retained so that an original *TADS* starting point is always available.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE May 1999	3. REPORT TYPE AND DATES COVERED Final Contractor Report

4. TITLE AND SUBTITLE

TADS—A CFD-Based Turbomachinery Analysis and Design System With GUI
Version 2.0—User's Manual

5. FUNDING NUMBERS

WU-538-03-11-00
NAS3-27394, Task 10

6. AUTHOR(S)

M.J. Koiro, R.A. Myers, and R.A. Delaney

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Allison Engine Division of Pratt & Whitney
2001 South Tibbs
Indianapolis, Indiana 46206

8. PERFORMING ORGANIZATION REPORT NUMBER

E-11117

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135-3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—1999-206604

11. SUPPLEMENTARY NOTES

Project Manager, C.J. Miller, NASA Glenn Research Center, organization code 5940, (216) 433-6179.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 07 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The primary objective of this study was the development of a Computational Fluid Dynamics (CFD) based turbomachinery airfoil analysis and design system, controlled by a Graphical User Interface (GUI) . The computer codes resulting from this effort are referred to as TADS (Turbomachinery Analysis and Design System). This document is intended to serve as a User's Manual for the computer programs which comprise the TADS system, developed under Task 18 of NASA Contract NAS3-27350, ADPAC System Coupling to Blade Analysis & Design System GUI and Task 10 of NASA Contract NAS3-27394, ADPAC System Coupling to Blade Analysis & Design System GUI, Phase II—Loss, Design and, Multi-stage Analysis. TADS couples a throughflow solver (ADPAC) with a quasi-3D blade-to-blade solver (RVCQ3D) in an interactive package. Throughflow analysis and design capability was developed in ADPAC through the addition of blade force and blockage terms to the governing equations. A GUI was developed to simplify user input and automate the many tasks required to perform turbomachinery analysis and design. The coupling of the various programs was done in such a way that alternative solvers or grid generators could be easily incorporated into the TADS framework. Results of aerodynamic calculations using the TADS system are presented for a highly loaded fan, a compressor stator, a low speed turbine blade and a transonic turbine vane.

14. SUBJECT TERMS

Turbomachinery; Compressor; Turbine; Computational fluid dynamics

15. NUMBER OF PAGES
99

16. PRICE CODE
A05

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT