NASA Contractor Report 198441

NASA-CR-198441
19960010174

# TADS–A CFD-Based Turbomachinery and Analysis Design System With GUI

## Volume II—User's Manual

R.A. Myers, D.A. Topp, and R.A. Delaney
*Allison Engine Company*
*Indianapolis, Indiana*

December 1995

Prepared for
Lewis Research Center
Under Contract NAS3–25950

National Aeronautics and
Space Administration

# Contents

**1 Summary** **1**

**2 Introduction** **3**

**3 Conventions and Nomenclature** **7**
3.1 Typographic Conventions . . . . . . . . . . . . . . . . . . . . . . 7
3.2 Nomenclature . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
3.3 GUI Conventions . . . . . . . . . . . . . . . . . . . . . . . . . 8
3.3.1 Windows . . . . . . . . . . . . . . . . . . . . . . . . 8
3.3.2 Mouse Buttons . . . . . . . . . . . . . . . . . . . . . 8
3.3.3 Pulldown Lists . . . . . . . . . . . . . . . . . . . . . 8
3.3.4 Toggle Buttons . . . . . . . . . . . . . . . . . . . . . 9
3.3.5 Radio Buttons . . . . . . . . . . . . . . . . . . . . . 9
3.3.6 Push-buttons . . . . . . . . . . . . . . . . . . . . . . 9
3.3.7 Text Boxes . . . . . . . . . . . . . . . . . . . . . . . 9
3.3.8 Action Buttons . . . . . . . . . . . . . . . . . . . . . 9
3.4 File Formats . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.5 File Naming . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

**4 Preparing Input for *TADS*** **13**
4.1 Airfoil Description . . . . . . . . . . . . . . . . . . . . . . . . 13
4.2 Flowpath Description . . . . . . . . . . . . . . . . . . . . . . . 14
4.3 Aerodynamic Data . . . . . . . . . . . . . . . . . . . . . . . . 16

**5 Main Panel** **19**
5.1 Edit Programs Mode . . . . . . . . . . . . . . . . . . . . . . . 21
5.2 Edit Data Mode . . . . . . . . . . . . . . . . . . . . . . . . . . 21

5.3 Edit/Run Mode . . . . . . . . . . . . . . . . . . . . . . . . . 23
5.4 Run Mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
5.5 Edit Machines Mode . . . . . . . . . . . . . . . . . . . . . . 23

**6 Remote Processor Setup Panel 25**

**7 Input Panels 27**
7.1 Standardized Data Input Panels . . . . . . . . . . . . . . . . . 28
7.1.1 *TIGG* Input Panel . . . . . . . . . . . . . . . . . . . . 31
7.1.2 *ADPAC* Input Panel . . . . . . . . . . . . . . . . . . . 33
7.1.3 *GRAPE* Input Panel . . . . . . . . . . . . . . . . . . . 35
7.1.4 *RVCQ3D* Input Panel . . . . . . . . . . . . . . . . . . 35
7.2 *SLICER* Data Input Panel . . . . . . . . . . . . . . . . . . . . 39
7.2.1 **Percent Mass** Slice Mode . . . . . . . . . . . . . . . . . 42
7.2.2 **Percent Span** Slice Mode . . . . . . . . . . . . . . . . . 42
7.2.3 **Percent Area** Slice Mode . . . . . . . . . . . . . . . . . 42
7.2.4 **Inches** Slice Mode . . . . . . . . . . . . . . . . . . . . 43

**8 *TADS* Operating Instructions 45**
8.1 Installing *TADS* . . . . . . . . . . . . . . . . . . . . . . . . 45
8.2 User Setup . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
8.3 Executing *TADS* . . . . . . . . . . . . . . . . . . . . . . . . 47
8.4 Trouble Shooting *TADS* . . . . . . . . . . . . . . . . . . . . 49

**9 On Line Documentation 51**

**A Complete List of Input and Output Files 55**

**B Sample Flowpath Description Input File 59**

**C Sample Aerodynamic Data Input File 61**

**D Sample X Resource File 65**

**E Extracting the Source Files 69**

**F Compiling *TADS* Components 73**

**G Running the Distribution Demonstration Test Case** **75**

# List of Figures

2.1 The coupled throughflow and blade-to-blade analysis is an iterative, multi-step process. . . . . . . . . . . . . . . . . . . . . . 5

4.1 The airfoil shape is defined as a surface in two parameters. . . 15

5.1 *TADS* main panel controls the coupled analysis. . . . . . . . . 20
5.2 Program mode selector controls the GUI's appearance and execution. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
5.3 The component group controls change in appearance and function as the program mode is changed. . . . . . . . . . . . . . 22

6.1 Program modules can be run on remote hosts configured using the Setup Panel. . . . . . . . . . . . . . . . . . . . . . . . . 26

7.1 The error panel will display the valid range. . . . . . . . . . . 30
7.2 *TIGG* input panel controls the axisymmetric grid generation. . 33
7.3 *ADPAC* input panel controls the through-flow analysis. . . . . 34
7.4 *GRAPE* input panel controls the blade-to-blade grid generation. 36
7.5 *RVCQ3D* input panel controls the blade-to-blade analysis. . . 38
7.6 Slicer input panel controls the location of the 2-D analyses. . . 40
7.7 User has the option to manually define slice geometry. . . . . . 41

# List of Tables

2.1 Coupled analysis organization. . . . . . . . . . . . . . . . . . . . 6

4.1 Required input data files. . . . . . . . . . . . . . . . . . . . . . . 13

7.1 Action buttons on standardized input panels control file creation, modification and restoration. . . . . . . . . . . . . . . . . 29

7.2 Action buttons on "Slicer" input panel control file creation, modification, restoration and program execution. . . . . . . . 32

7.3 The *GRAPE* input parameters were modified to enhance grid quality while reducing user effort. . . . . . . . . . . . . . . . . 37

F.1 *TADS* development platform software configuration. . . . . . . 73

# 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.

*TADS* couples a throughflow solver (*ADPAC*) with a quasi-3D blade-to-blade solver (*RVCQ3D*) in an interactive package. Throughflow analysis 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 cases.

*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.

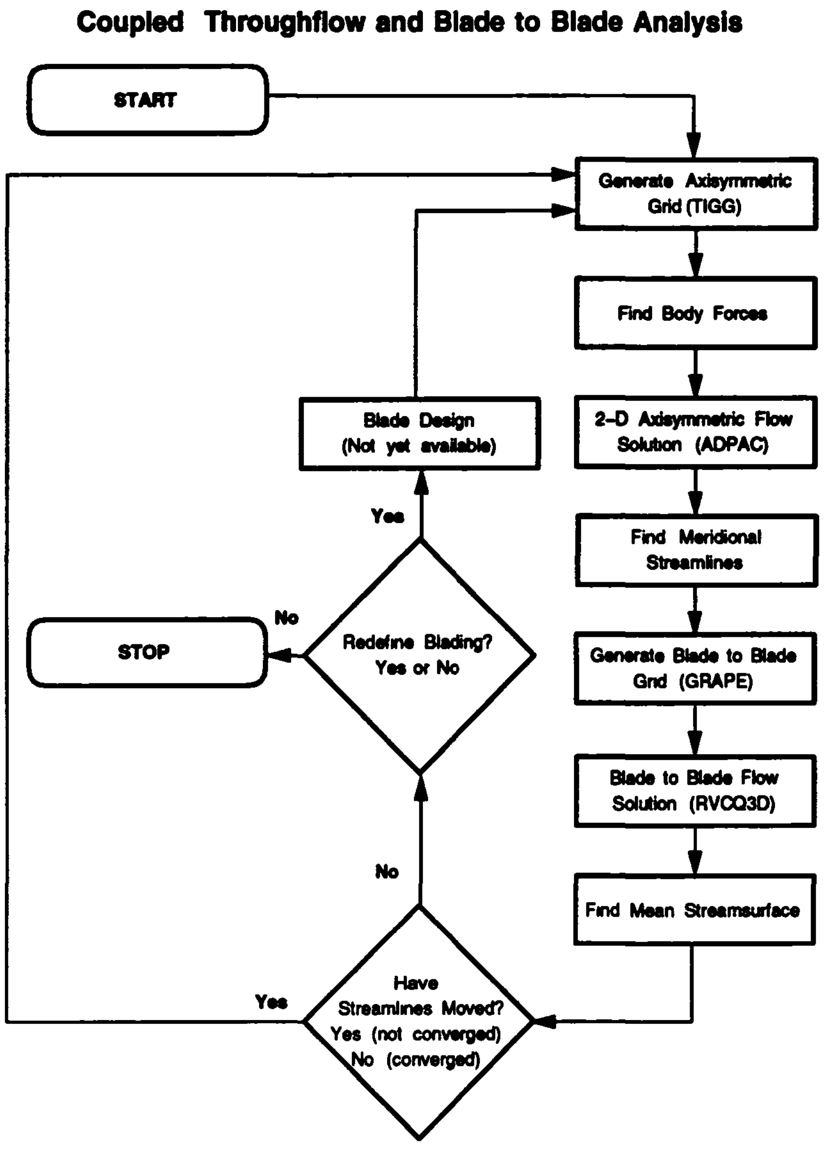

Coupled Throughflow and Blade to Blade Analysis
START
Generate Axisymmetric Grid (TIGG)
Find Body Forces
2-D Axisymmetric Flow Solution (ADPAC)
Find Meridional Streamlines
Generate Blade to Blade Grid (GRAPE)
Blade to Blade Flow Solution (RVCQ3D)
Find Mean Streamsurface
Have Streamlines Moved? Yes (not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

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* | | |
|---|---|---|---|---|
| Axisymmetric Grid Generation | *TIGG*<br>*BATCH TIGG* | intigg<br>intigg | tiggc3d<br>tiggc3d | |
| Throughflow Calculation | *ADPAC*<br>*VIADAC* | adpacbc<br>n/a | bodyf | adpac |
| Slicer | *SLICER* | radsl | slicer | |
| Blade-to-Blade Grid Generation | *GRAPE* | grape | | |
| Blade-to-Blade Calculation | *RVCQ3D*<br>*PGASC*<br>*TSONIC* | fixrvc<br>n/a<br>n/a | rvccq3d | |
| Mean Streamline Finder | *MEANSL* | restack | meansl | |

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
- Message panel
- Remote processor setup panel
- Standardized data input panels

  Slice independent panels

  Slice dependent panels
- Specialized data input panels

# 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, [ ].
- Keypresses 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, acronymns 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 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.

### 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 Ref.[8]. 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 slice 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, most files are 3-D, whole, multiple grid files, in accordance with the definitions in Ref. [7], 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-to-blade section to be analyzed. Within the subdirectory, 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 four things as input: a casename, a Cartesian description of the airfoil, a description of the meridional flowpath, and aerodynamic data. The minimum file set required for *TADS* to execute 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 around the airfoil to form a closed surface, when viewed from above. 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 descrip-

Table 4.1: Required input data files.

| *Name* | *Format* | *Description* |
|---|---|---|
| `casename.tdsblad` | *PLOT3D* (SDB) | airfoil geometry |
| `casename.tdspath` | ASCII | flowpath geometry |
| `casename.tdsaro` | ASCII | aerodynamic data |

tion with the machine axis aligned with the $X$ direction. The file is a 3-D, whole, multiple grid file, in accordance with the definitions in Ref. [7], 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.

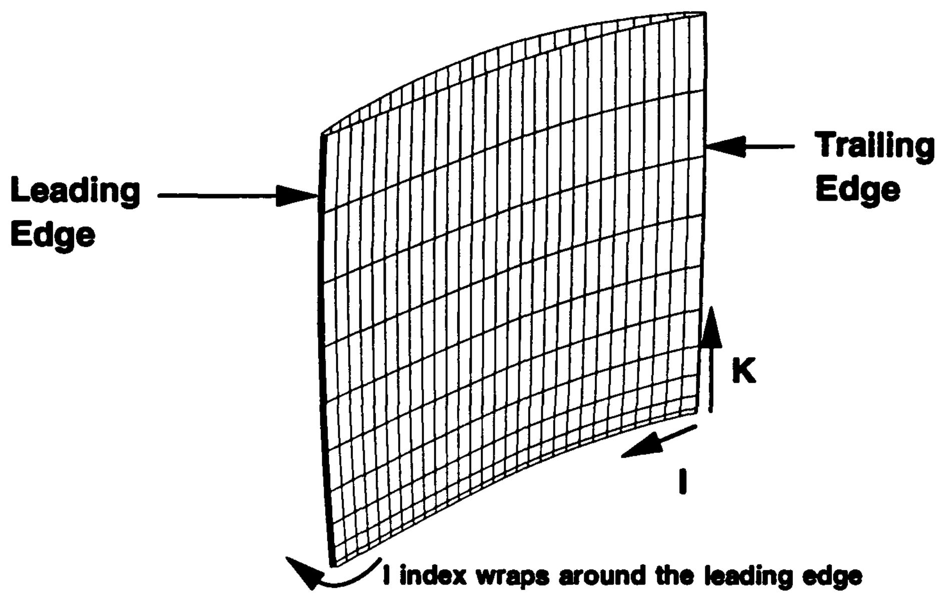
## 4.2 Flowpath Description

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

**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.

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 ($\gamma$) 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. 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. 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.

# 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 three 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.

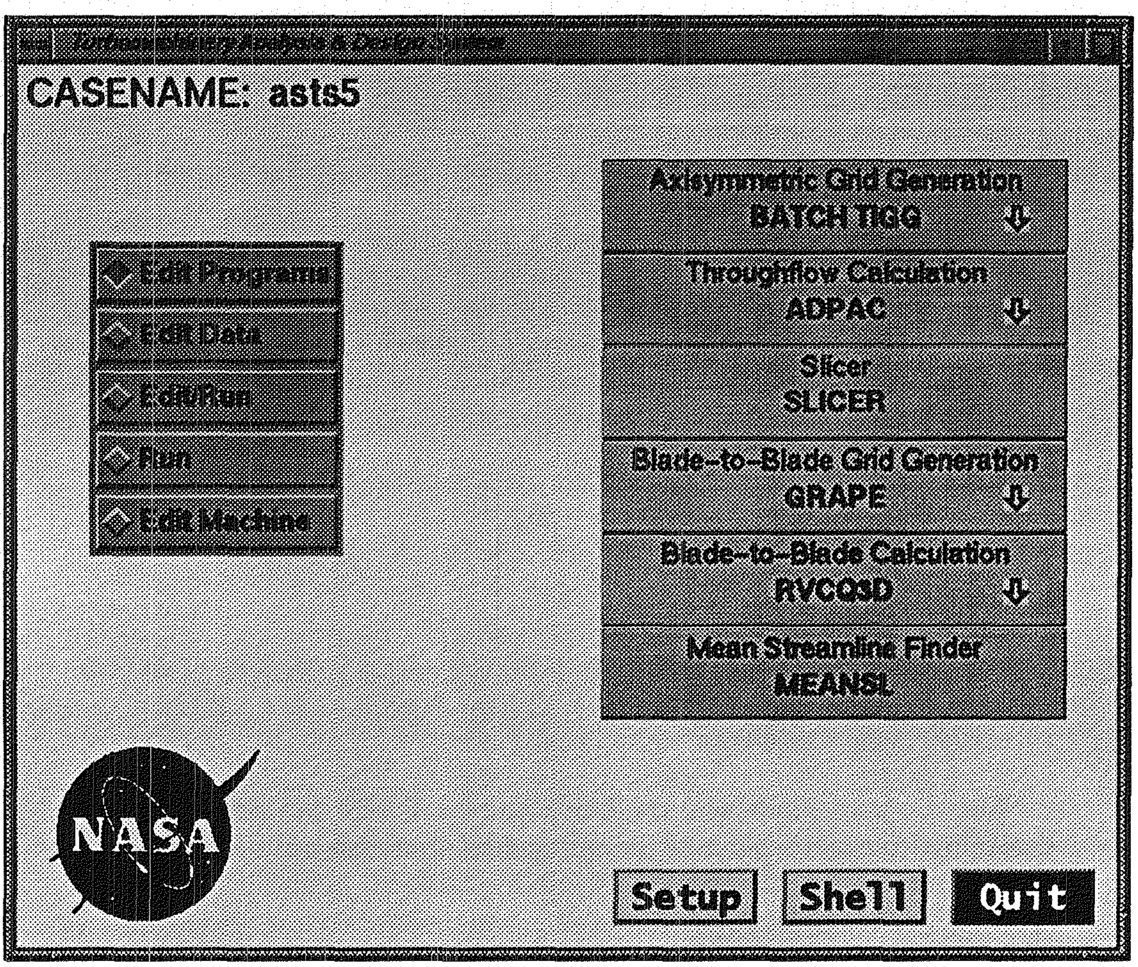


Figure 5.1: *TADS* main panel controls the coupled analysis.

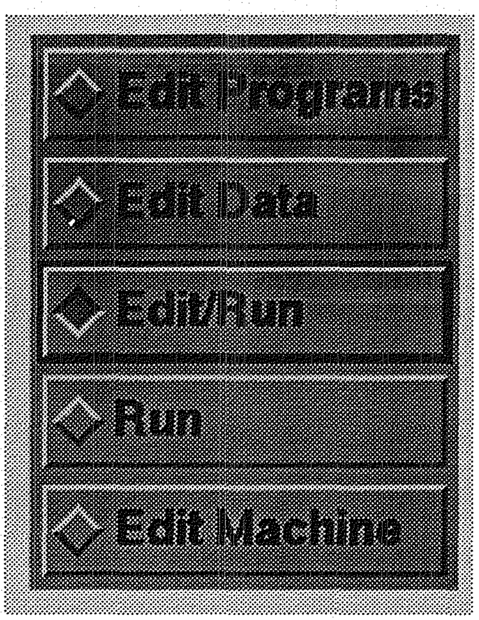


Figure 5.2: Program mode selector controls the GUI's appearance and execution.

## 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.

## 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,

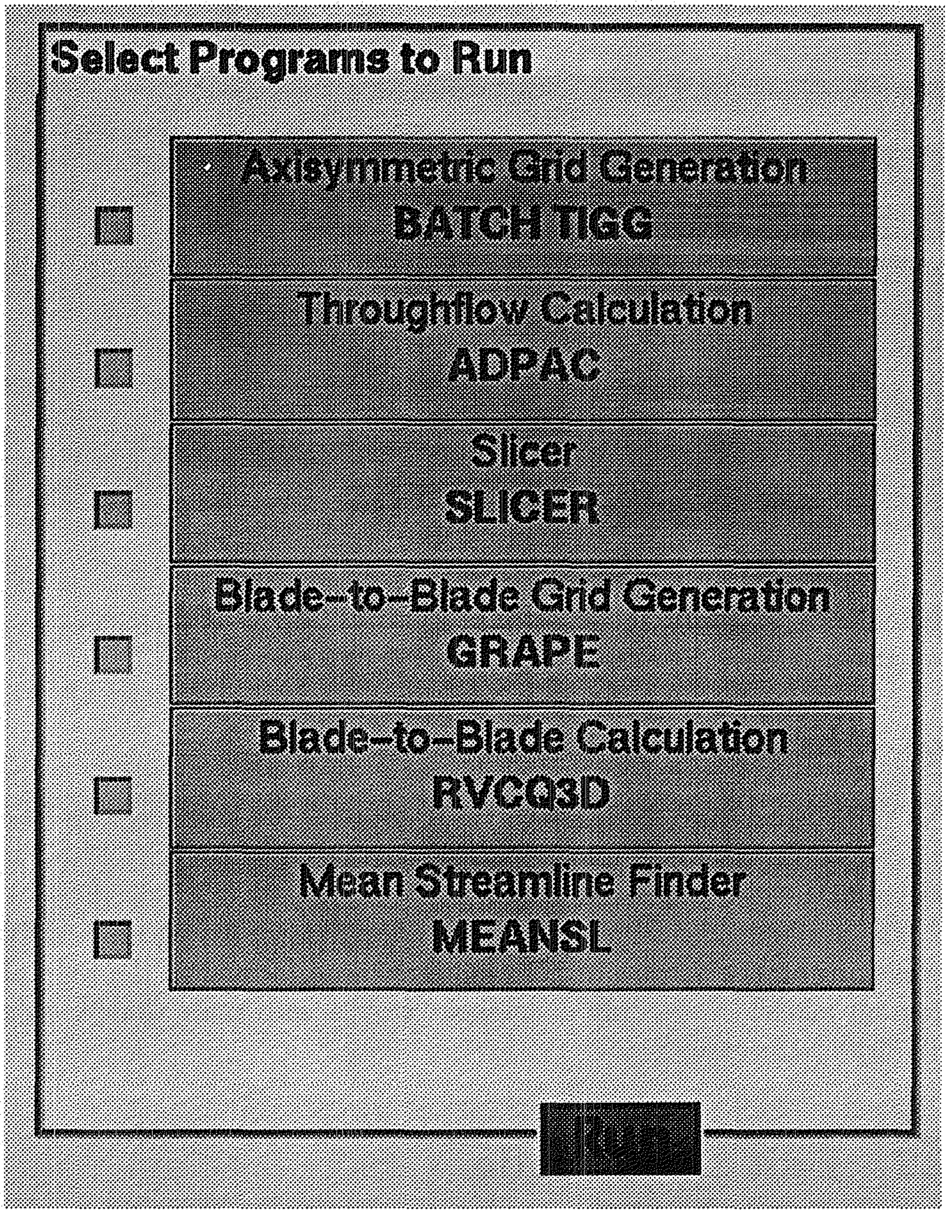


Figure 5.3: The component group controls change in appearance and function as the program mode is changed.

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 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 run on a local GL machine or on two SGI machines.

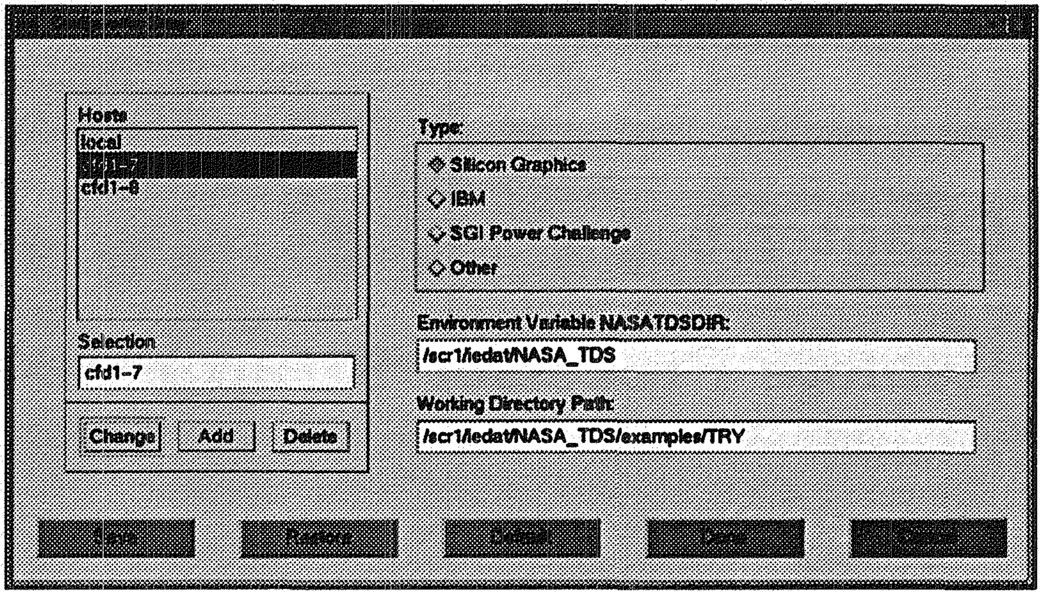
# Chapter 6

# 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



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.

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 7.1.

# Chapter 7

# 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).

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 7.1 for all input panels.

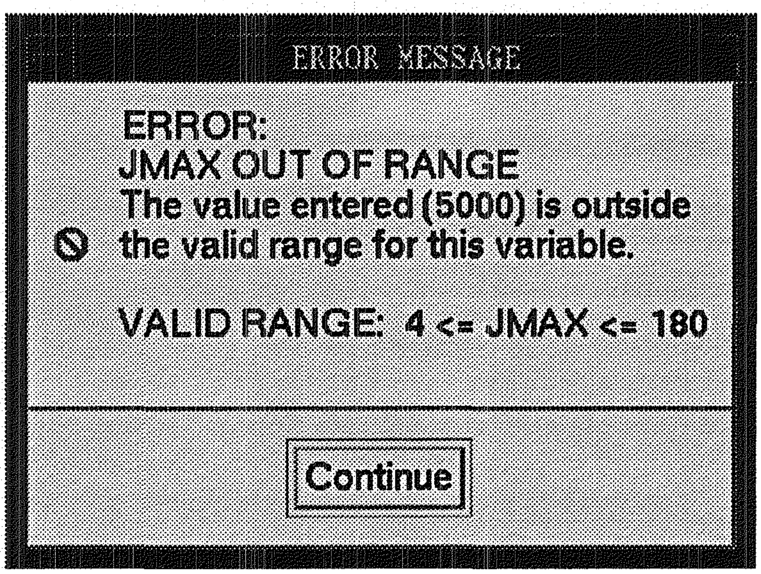
## 7.1 Standardized Data Input Panels

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 two distinct subclasses: slice dependent and slice independent. The input panels of these subclasses appear almost identical, except for some additional controls on the slice dependent input panels. The additional controls are described in greater detail later in this section.

For both slice dependent and independent 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.

Table 7.1: 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. |


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 7.1: The error panel will display the valid range.

If a numeric entry fails an active bounds checker, an error panel appears (see Figure 7.1 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.

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 7.2.

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. 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 only and allows the user to debug input data without having to run solutions on all slices. The button is only active for individual slices and only 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 locked out for individual slices. This also prevents the user from inadvertently changing slice independent data for a single slice.

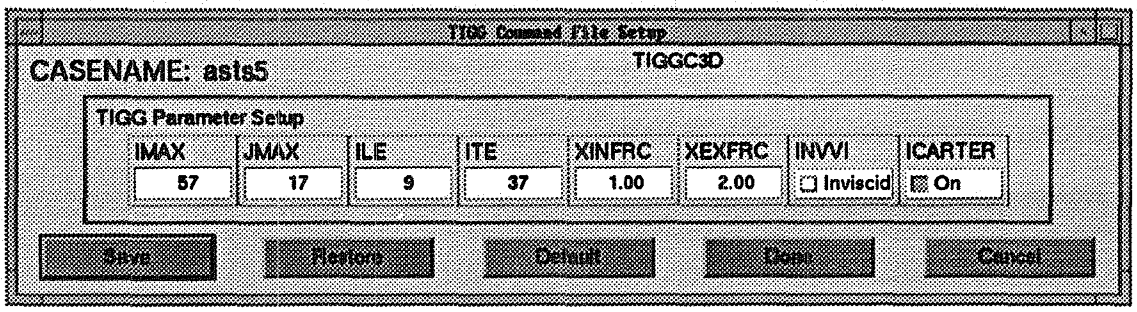
### 7.1.1 *TIGG* Input Panel

*TIGG* is a slice independent component module. It uses a standardized input panel as described in section 7.1. A representative screen image of it's GUI input panel is shown if Figure 7.2. The controls correspond to input described in the *TIGG* user's manual (Ref.[5]). This input panel is used for both *TIGG* and *BATCH TIGG*[2] component modules. This is possible because the input for both is identical.

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

Table 7.2: 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 7.1). |
| **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

TIGG Parameter Setup

| IMAX | JMAX | ILE | ITE | XINFRC | XEXFRC | INVVI | ICARTER |
|---|---|---|---|---|---|---|---|
| 57 | 17 | 9 | 37 | 1.00 | 2.00 | ☐ Inviscid | ▨ On |

Figure 7.2: *TIGG* input panel controls the axisymmetric grid generation.

### 7.1.2 *ADPAC* Input Panel

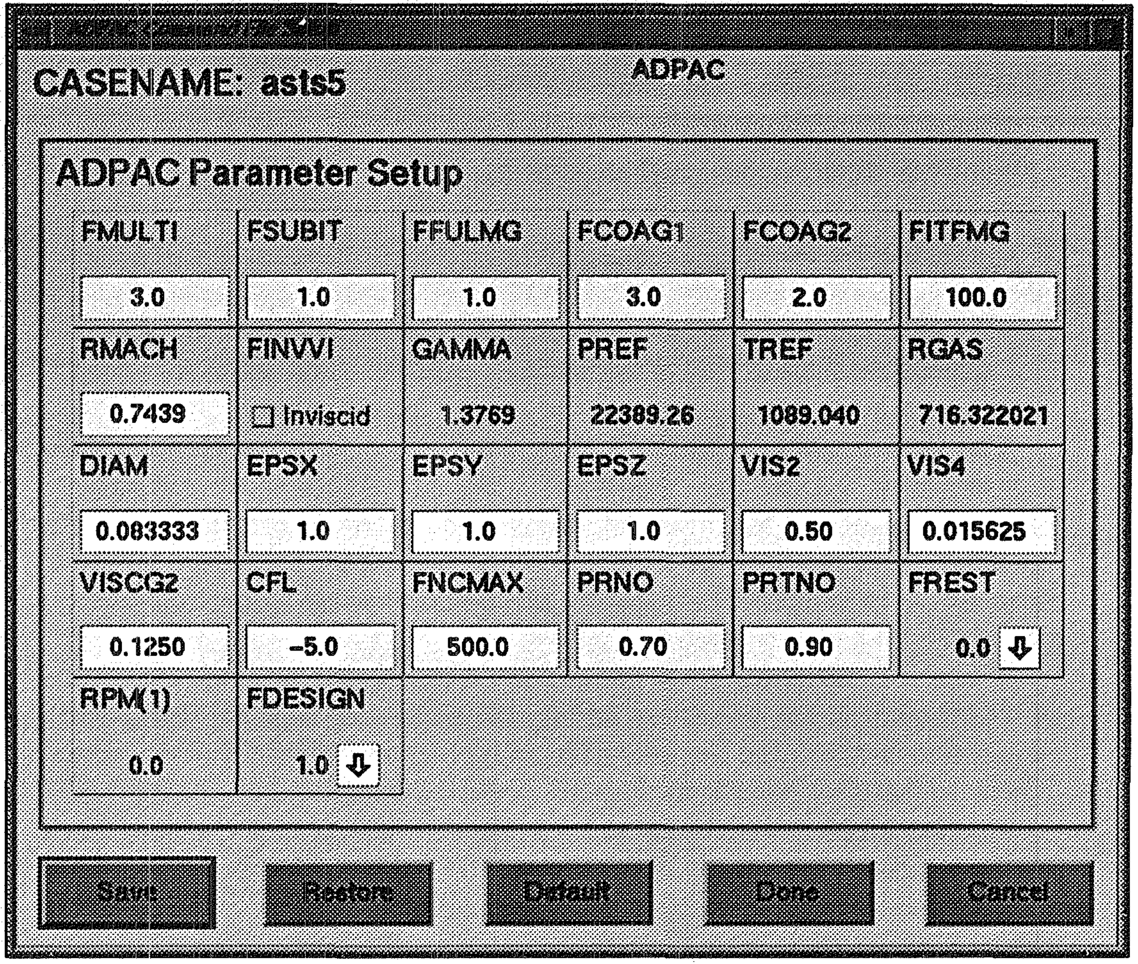
*ADPAC* is a standardized, slice independent component module. A representative screen shot is shown if Figure 7.3. The control labels correspond to the input keywords described in the *ADPAC* user's manual (Ref.[4]).

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*.

**Warning:** *TADS* will only recognize the *ADPAC* input parameters shown on the input panel and in the given order (left to right and top to bottom). Similarly, it will only write out these same inputs to the *ADPAC* input file (`casename.adpac.input`) in a fixed order. These conditions limit the user's ability to specify *ADPAC* input data prior to execution. The user is free to use any legal *ADPAC* input format as long as *TADS* is not initialized or the *ADPAC* input panel is not invoked from within *TADS*. Either of these


CASENAME: asts5

ADPAC

ADPAC Parameter Setup

| | | | | | |
|---|---|---|---|---|---|
| FMULTI | FSUBIT | FFULMG | FCOAG1 | FCOAG2 | FITFMG |
| 3.0 | 1.0 | 1.0 | 3.0 | 2.0 | 100.0 |
| RMACH | FINVVI | GAMMA | PREF | TREF | RGAS |
| 0.7439 | ☐ Inviscid | 1.3769 | 22389.26 | 1089.040 | 716.322021 |
| DIAM | EPSX | EPSY | EPSZ | VIS2 | VIS4 |
| 0.083333 | 1.0 | 1.0 | 1.0 | 0.50 | 0.015625 |
| VISCG2 | CFL | FNCMAX | PRNO | PRTNO | FREST |
| 0.1250 | -5.0 | 500.0 | 0.70 | 0.90 | 0.0 ⇩ |
| RPM(1) | FDESIGN | | | | |
| 0.0 | 1.0 ⇩ | | | | |

Figure 7.3: *ADPAC* input panel controls the through-flow analysis.

conditions will cause *TADS* to reread/rewrite the *ADPAC* input file.

### 7.1.3 *GRAPE* Input Panel

*GRAPE* is a slice dependent component module. It uses a standardized input panel as described in Section 7.1. Figure 7.4 shows how the controls are grouped to correspond to the namelist structure of the input as described in the *GRAPE* user's manual (Ref.[6] and Ref.[3]). Some modifications were made to *GRAPE* input parameters as noted in Table 7.1.3. 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.

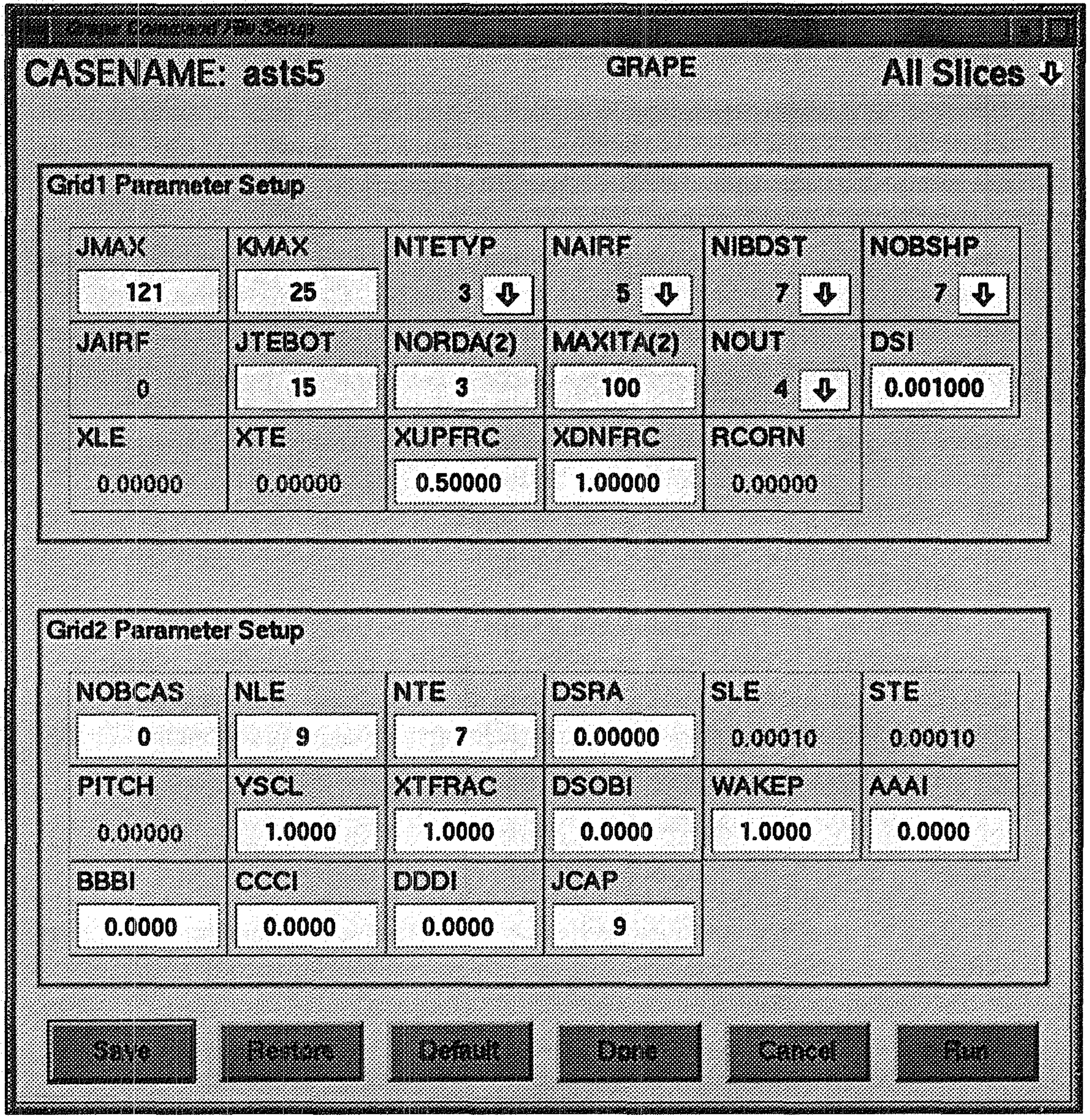
The *GRAPE* input panel will process only the namelist input parameters shown on the input panel. If the user wishes to input *GRAPE* namelist variables not shown on the panel he/she must input the data directly into the ASCII file before *GRAPE* is executed. Once this has been done, the *GRAPE* input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

### 7.1.4 *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 (Ref.[1]). Figure 7.5 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



Grape Command File Setup

CASENAME: asts5 GRAPE All Slices ⇩

Grid1 Parameter Setup

| | | | | | |
|---|---|---|---|---|---|
| JMAX | KMAX | NTETYP | NAIRF | NIBDST | NOBSHP |
| 121 | 25 | 3 ⇩ | 5 ⇩ | 7 ⇩ | 7 ⇩ |
| JAIRF | JTEBOT | NORDA(2) | MAXITA(2) | NOUT | DSI |
| 0 | 15 | 3 | 100 | 4 ⇩ | 0.001000 |
| XLE | XTE | XUPFRC | XDNFRC | RCORN | |
| 0.00000 | 0.00000 | 0.50000 | 1.00000 | 0.00000 | |

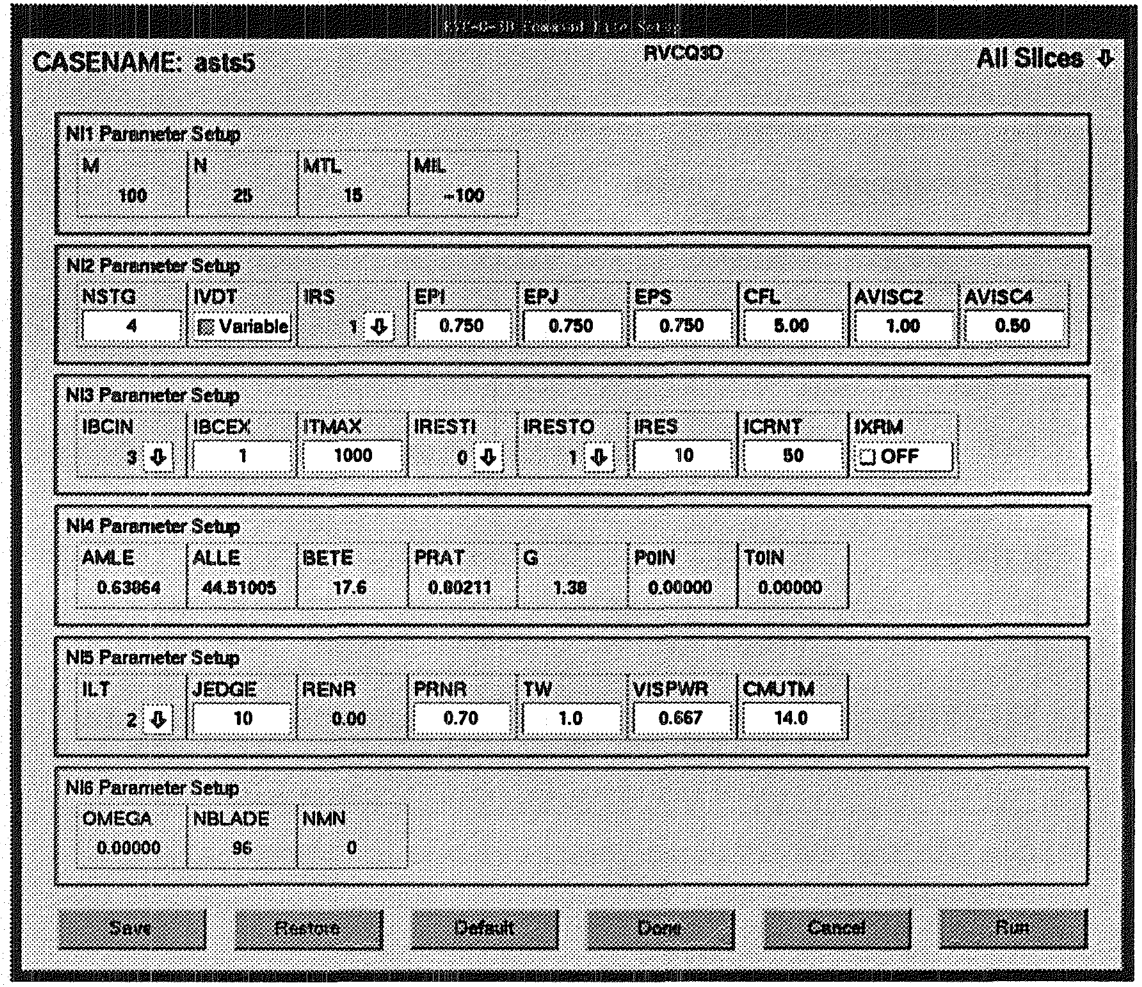
Grid2 Parameter Setup

| | | | | | |
|---|---|---|---|---|---|
| NOBCAS | NLE | NTE | DSRA | SLE | STE |
| 0 | 9 | 7 | 0.00000 | 0.00010 | 0.00010 |
| PITCH | YSCL | XTFRAC | DSOBI | WAKEP | AAAI |
| 0.00000 | 1.0000 | 1.0000 | 0.0000 | 1.0000 | 0.0000 |
| BBBI | CCCI | DDDI | JCAP | | |
| 0.0000 | 0.0000 | 0.0000 | 9 | | |

Figure 7.4: *GRAPE* input panel controls the blade-to-blade grid generation.

| 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**. |

Table 7.3: The *GRAPE* input parameters were modified to enhance grid quality while reducing user effort.



CASENAME: asts5 RVCQ3D All Slices ⇩

NI1 Parameter Setup

| M | N | MTL | MIL |
|---|---|---|---|
| 100 | 25 | 15 | -100 |

NI2 Parameter Setup

| NSTG | IVDT | IRS | EPI | EPJ | EPS | CFL | AVISC2 | AVISC4 |
|---|---|---|---|---|---|---|---|---|
| 4 | ▨ Variable | 1 ⇩ | 0.750 | 0.750 | 0.750 | 5.00 | 1.00 | 0.50 |

NI3 Parameter Setup

| IBCIN | IBCEX | ITMAX | IRESTI | IRESTO | IRES | ICRNT | IXRM |
|---|---|---|---|---|---|---|---|
| 3 ⇩ | 1 | 1000 | 0 ⇩ | 1 ⇩ | 10 | 50 | ☐ OFF |

NI4 Parameter Setup

| AMLE | ALLE | BETE | PRAT | G | P0IN | T0IN |
|---|---|---|---|---|---|---|
| 0.63864 | 44.51005 | 17.6 | 0.80211 | 1.38 | 0.00000 | 0.00000 |

NI5 Parameter Setup

| ILT | JEDGE | RENR | PRNR | TW | VISPWR | CMUTM |
|---|---|---|---|---|---|---|
| 2 ⇩ | 10 | 0.00 | 0.70 | 1.0 | 0.667 | 14.0 |

NI6 Parameter Setup

| OMEGA | NBLADE | NMN |
|---|---|---|
| 0.00000 | 96 | 0 |

Figure 7.5: *RVCQ3D* input panel controls the blade-to-blade analysis.

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.

## 7.2 *SLICER* Data Input Panel

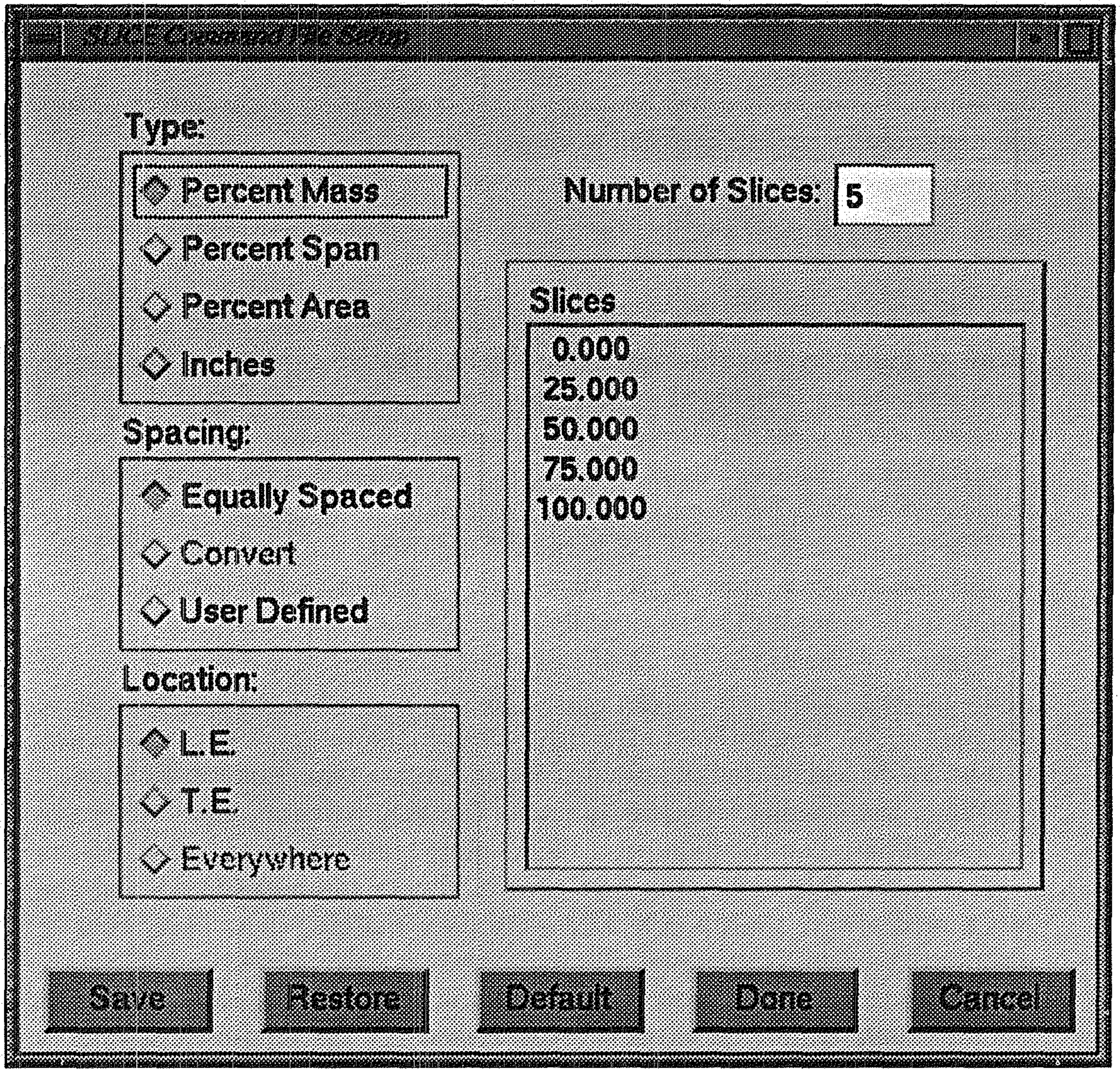
The blade-to-blade analysis is performed along streamlines in the meridional plane as found by the throughflow analysis. This requires that the meridional streamlines be located in the throughflow solution, and that the airfoil be sliced along these streamlines. *TADS* uses two separate programs to accomplish this purpose: *radsl* and *slicer.* This 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 7.6 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 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 7.7). 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.


SLICE Command File Setup

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

Spacing: [✓ Equally Spaced] Convert User Defined

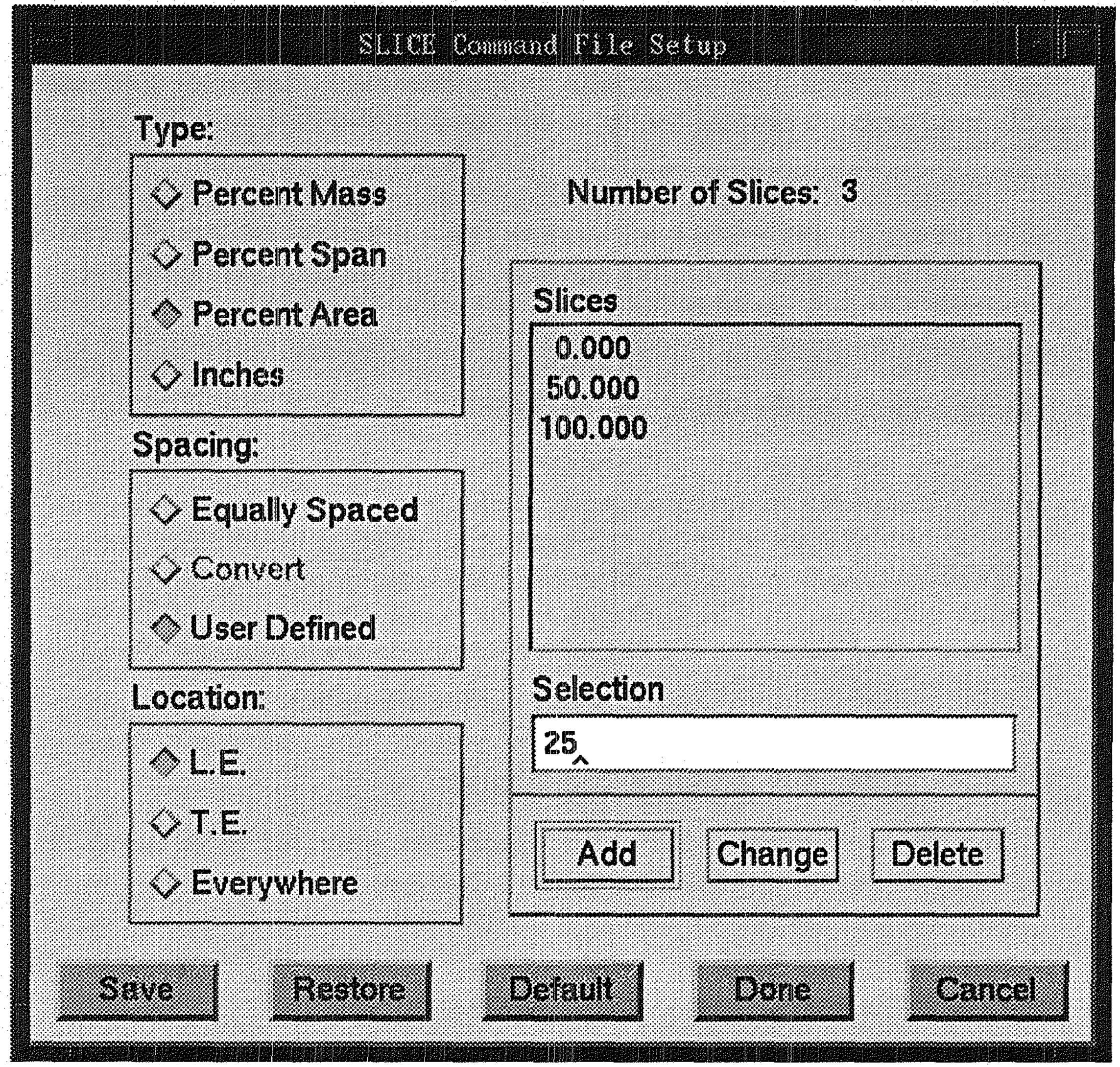
Location: [✓ L.E.] T.E. Everywhere

Number of Slices: 5

Slices

```
  0.000
 25.000
 50.000
 75.000
100.000
```

Figure 7.6: Slicer input panel controls the location of the 2-D analyses.


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 7.7: User has the option to manually define slice geometry.

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.

### 7.2.1 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.

### 7.2.2 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.

### 7.2.3 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

[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.

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).

### 7.2.4 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.

# Chapter 8

# 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.01`). Instructions for extracting the distribution can be found in Appendix E.

*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*.

## 8.1 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".

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

To begin installation, it is first necessary to enter the install directory (`TADS.01`) with the command:

**cd** *"Path to* `TADS.01` *directory"*/`TADS.01`

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:

*cleanup_TADS*

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

## 8.2 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.01` *directory"*/`TADS.01`

The user must set the `TADSDIR` environment variable to the absolute path of the *TADS* install directory (`TADS.01`) with the command:

**setenv** `TADSDIR` *"Path to* `TADS.01` *directory"*/`TADS.01.`

*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 D.

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*.

## 8.3 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 G.

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*. It 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.

## 8.4 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.

# Chapter 9

# 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.01/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.01/html/TADS.homepage.html
```

# Bibliography

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

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

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

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

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

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

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

[8] 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.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.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. |
| `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. |

*continued on next page*

*continued from previous page*

| *Name* | *Format* | *Description* |
|---|---|---|
| **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 |
| **tds_casename** | ASCII | Text file which contains current case name – this file is used by the fortran programs to construct file names |
| **casename.sl.#** | Directory | Subdirectory name 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 ouput file used for *PLOT3D* post-processing and as input to *RVCQ3D* |
| `restout.bin` | *PLOT3D* | *RVCQ3D* 2-D, relative, single grid, SDB binary ouput 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
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
```

# Appendix D

# 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 E

# 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.01.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.01.tar.Z
```

This operation essentially replaces the compressed file `TADS.01.tar.Z` with an uncompressed file `TADS.01.tar` .

The next step is to extract the individual files and directories from the `TADS.01.tar` file. Before this is done, the user must put the `TADS.01.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.01.tar
```

(Note, on some systems **tar xvf** TADS.01.tar may be sufficient.)

Execution of the UNIX list command **ls** will verify that the TADS.01directory has been created. The **tar** command will have created a top level directory named TADS.01in the current directory. The TADS.01directory 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.01.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   TOOLS/         apl/
cleanup_TADS*  csdb/          examples/      gui/
guilib/        html/          install_TADS*  install_user*
misc/          modules/       sdb/
```

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. |
| 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 | Directroy 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. |
| **`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 F

# 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 F.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

Table F.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

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.

# Appendix G

# Running the Distribution Demonstration Test Case

After *TADS* has been properly installed and configured (see Section 8.1 or Appendix F), 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_S5/    TRY/
```

Both of these directories contain identical required input files (see Table 4.1) for AST Stator 5. The **AST_S5** 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 his/her results to the results in the **TRY** directory.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No 0704-0188*

| | | |
|---|---|---|
| 1 AGENCY USE ONLY (*Leave blank*) | 2 REPORT DATE<br>December 1995 | 3 REPORT TYPE AND DATES COVERED<br>Final Contractor Report |

| | |
|---|---|
| 4 TITLE AND SUBTITLE<br>TADS–A CFD-Based Turbomachinery and Analysis Design System With GUI<br>Volume II—Users's Manual | 5 FUNDING NUMBERS<br>WU-505-62-10<br>C-NAS3-25950 |
| 6 AUTHOR(S)<br>R A Myers, D A Topp, and R A Delaney | |
| 7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Allison Engine Company<br>PO Box 420<br>Indianapolis, Indiana 46206–0420 | 8 PERFORMING ORGANIZATION REPORT NUMBER<br>E-10059 |
| 9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>National Aeronautics and Space Administration<br>Lewis Research Center<br>Cleveland, Ohio 44135-3191 | 10 SPONSORING/MONITORING AGENCY REPORT NUMBER<br>NASA CR-198441 |

11 SUPPLEMENTARY NOTES

Project Manager, Kestutis C Civinskas, Propulsion Systems Division, NASA Lewis Research Center, organization code 2760, (216) 433–3944

| | |
|---|---|
| 12a DISTRIBUTION/AVAILABILITY STATEMENT<br>Unclassified - Unlimited<br>Subject Category 07<br>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 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 *TADS* couples a throughflow solver (*ADPAC*) with a quasi-3D blade-to-blade solver (*RVCQ3D*) in an interactive package Throughflow analysis 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<br>Computational fluid dynamics, Design, Turbomachinery | 15 NUMBER OF PAGES<br>85 |
| | 16 PRICE CODE<br>A05 |

| | | | |
|---|---|---|---|
| 17 SECURITY CLASSIFICATION OF REPORT<br>Unclassified | 18 SECURITY CLASSIFICATION OF THIS PAGE<br>Unclassified | 19 SECURITY CLASSIFICATION OF ABSTRACT<br>Unclassified | 20 LIMITATION OF ABSTRACT |

**End of Document**