NASA Contractor Report 4349

TranAir: A Full-Potential, Solution-Adaptive, Rectangular Grid Code for Predicting Subsonic, Transonic, and Supersonic Flows About Arbitrary Configurations

User's Manual

F. T. Johnson, S. S. Samant, M. B. Bieterman,
R. G. Melvin, D. P. Young, J. E. Bussoletti,
and C. L. Hilmes
Boeing Military Airplane Company
Seattle, Washington

Prepared for
Ames Research Center
under Contract NAS2-12513

National Aeronautics and
Space Administration

Office of Management

Scientific and Technical
Information Program

1992

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1– INTRODUCTION

TranAir is a system of computer programs developed to analyze compressible flow over arbitrary complex configurations at subsonic, transonic or supersonic freestream Mach numbers. TranAir solves the non-linear, full potential equation subject to a variety of boundary conditions modeling wakes, inlets, exhausts, porous walls, and impermeable surfaces. TranAir can also model regions of different total temperature and pressure (material properties) in the flow field.

In the numerical method implemented in TranAir, the configuration geometry and the flow volume are discretized independently of each other. The flow field is divided into a locally refined rectangular grid which is generated *internally* by the code. The surface boundary is divided into networks of panels where separate boundary conditions can be specified. The output flow quantities of interest are returned at panel corner points and panel center points.

TranAir can solve problems with up to 70,000 panels and up to approximately one million finite elements on a machine with 8MW main memory and 200MW SSD[1] (CRAY X-MP or Y-MP) or 128 MW main memory (CRAY 2). The CPU time required for a run with 300,000 finite elements at transonic conditions is approximately 3000 seconds.

1–1 OVERVIEW OF MANUAL

To help you orient and/or locate information, a description of each section of this manual is provided below:

[1]CRAY, CRAY Y-MP, SSD, and UNICOS are registered trademarks and CRAY X-MP, and CRAY-2 are trademarks of Cray Research, Inc.

Table 1.1: Overview of TranAir User's Manual

SECTION	DESCRIPTION
2. Description of TranAir	Method, organization, data communication used.
3. Overview of an Aerodynamic Analysis	Outlines the complete performance of a TranAir analysis.
4. Program Input	Describes inputs required to run the program. The descriptions are divided into sections corresponding to TranAir input data blocks.
5. Program Output	Describes some program output.
A. Running TranAir on CRAY-Y-MP	Gives script for running TranAir.
B. TranAir Graphics Utility (TGRAF)	Support program to examine field grids and properties. Recently added surface properties.
C. Boundary Layer Output Files	Output files for limited boundary layer analysis (New Capability).

1–2 HINTS FOR USING THIS MANUAL

Some helpful hints for using this manual are:

- Many sections start off with key information for the entire section. Spending a little time with the front of a section may help in understanding some of the particular details. This is particularly true for the program input (section 4.0).
- Most significant input program options and explanations are placed first in a description.
- Many of the input options and conventions are the same as the original TranAir program (reference 1).
- Some of the input parameters have been eliminated when they have no meaning for the current version of TranAir.

Please note that here are some new boundary conditions in this version of TranAir.

1–3 TranAir CAPABILITY

To get some idea of what TranAir can do, a brief summary of the three-dimensional steady flow capability is outlined below:

Problem Definition

TranAir predicts non-linear (and linear) full potential, subsonic, transonic, and supersonic irrotational flow properties about arbitrary 3-D lifting configurations. A wide variety of aerodynamic boundary conditions are available. Regions with different total pressure and temperature can be represented with user-specified boundaries between the regions.

Computed Flow Properties

- Surface flow properties
  - Isentropic pressure coefficient
  - Velocity components, nondimensional
  - Local Mach number
  - Potential
- 3-D surface pressure forces and moments (FX, FY, FZ, MX, MY, MZ)
  - Per panel column
  - Per network (an array of panels)
  - Accumulation of all previous networks

- Configuration forces and moments (CL, CD, CY, FX, FY, FZ, MX, MY, MZ)
  - Input configuration
  - Full configuration (with or without plane(s) of symmetry)
- Sectional pressure forces and moments along a user-specified plane (CFX, CFY, CFZ, CMX, CMY, CMZ, CLC, CDC, CNC, CMC, CLCCHORD/CREF, CUTLEN)
  - Per network
  - Total
  - Pressures along cut sections
- Flow-field properties for sample networks of points
  - Velocity components, nondimensional
  - Isentropic pressure coefficient
  - Potential
  - Local Mach number
  - Coupled Boundary layer applied to selected wing-like surfaces

Program Operational Features

- Solves one set of flow and boundary conditions in a computer run
- May solve for unsymmetric flow
- Reduces geometric input and computer cost for configurations with one or two planes of flow symmetry
- Restarts to continue a solution
- Restarts for a new solution (angle of attack, Mach number, etc.)
- Restarts to generate new or revised post-solution results
- Solution adaptive gridding to adjust field grid within user-specified range
- Grid sequencing solution to refine field grid as specified by user

Output Files

- Printout
- Field grids and flow properties
- Surface flow properties at panel network corner points
- Sectional cut pressures versus X, Y, Z, X/C
- Sectional forces and moments summary
- Configuration paneling after network edge abutment has been made
- PLOT3D for display in surface flow properties data.

What TranAir Cannot Do

- Predict flow dominated by viscous effects
- Predict flow dominated by STRONG (non-isentropic) transonic flow effects
- Predict flow with propeller slipstream swirl
- Will not automatically determine wake shapes

2– DESCRIPTION OF TranAir

This chapter provides a brief description of the TranAir method. The program organization and data communication are also described.

2–1 NUMERICAL METHOD

2–1.1 Boundary Value Problem

The TranAir code incorporates a numerical method to solve the full potential equation:

$$\vec{\nabla} \cdot \rho \vec{\nabla} \Phi = 0. \tag{2.1}$$

The density ρ and pressure p are given by the following isentropic formulas:

$$\rho = \rho_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{q^2}{q_\infty^2}) \right]^{\frac{1}{\gamma - 1}}, \tag{2.2}$$

$$p = p_\infty \left[1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{q^2}{q_\infty^2}) \right]^{\frac{\gamma}{\gamma - 1}}. \tag{2.3}$$

Here, Φ is the total velocity potential to be determined, $q = \| \vec{\nabla} \Phi \|_2$ is the speed given by the magnitude of the gradient of the potential, M is the Mach number, γ is the ratio of specific heats, and subscript ∞ denotes a value at a far upstream location (freestream value).

The boundary conditions include a far field condition that the perturbation potential tends to zero when moving away from the configuration boundary. The normal mass flux boundary condition may be specified to simulate impermeability or transpiration. It is also possible to impose a Dirichlet condition (for example, on an engine exhaust surface) where tangential flow can be prohibited by specifying potential to be a constant. The boundary conditions on wakes represent conservation of mass and normal momentum across a wake.

A modification of the above formulation allows the simulation of flows involving regions of differing total temperature and pressure. Potential flow exists in each separate region as long as total temperature and pressure are constant in each region. Hence, to model such regions, pressure and density in those regions have to be redefined in the following way:

$$\rho = \rho_\infty \frac{r_p}{r_T}\left[1 + \frac{\gamma-1}{2}M_\infty^2(1 - \frac{q^2}{q_\infty^2 r_T})\right]^{\frac{1}{\gamma-1}}, \tag{2.4}$$

$$p = p_\infty r_p\left[1 + \frac{\gamma-1}{2}M_\infty^2(1 - \frac{q^2}{q_\infty^2 r_T})\right]^{\frac{\gamma}{\gamma-1}}. \tag{2.5}$$

Here, r_p and r_T are the ratios of the total pressure and total temperature in the region to their corresponding freestream values.

The continuous problem posed by equation (2.1) is not amenable to analytic solution for any reasonable boundary conditions. A numerical method must be used in which the continuous problem is represented (*discretized*) by a set of nonlinear equations which are *solved* using an appropriate technique. Discretization and solution techniques used in TranAir are briefly described.

2–1.2 Discretization

The boundary value problem (stated in equations (2.1) through (2.5)) is defined over all of space. However, in TranAir, the computations are restricted to a finite subset of that domain. This is achieved by using sources for the Prandtl Glauert equation (which is a linearized form of the full potential equation) as the dependent variables instead of the potentials Φ on a uniform grid. Typically, the sources tend to decay much faster than the potential when moving away from the boundary. The computational region is restricted to that part of space where the sources are significant.

Once the computational region is established, a uniform *global grid* is constructed in that region. This global grid is further refined hierarchically (i.e., each box that is refined gives rise to eight similar boxes), producing the final computational grid (see figure 2.1). The rectangular elements penetrate the boundary; i.e., they do not conform to the boundary. Construction of such a grid is automated within the code and no surface conforming grid generation is required.

Discrete unknowns representing the solution are defined at the nodal points of this grid. Additional unknowns to account for nonlocal phenomena, such as wakes, are also defined at various points on the configuration boundary. Equation (2.1) is then approximated over the small volume elements in the grid (using a finite element method) to produce a system of non-linear equations. If the elemental volume includes any part of the boundary, then the non-linear equation(s) associated with that volume include the relevant boundary conditions. Generating finite element operators for purely rectangular cells away from the boundary is trivial. Near the boundary, the grid elements are cut by the boundary into general polyhedral shapes.

2–1.3 Solution Technique

The discretization described above yields a large, non-linear, non-symmetric, and often poorly conditioned system of equations. Newton's method is used to solve this system with each linearized problem being solved by the preconditioned GMRES method (see the Theory Document).

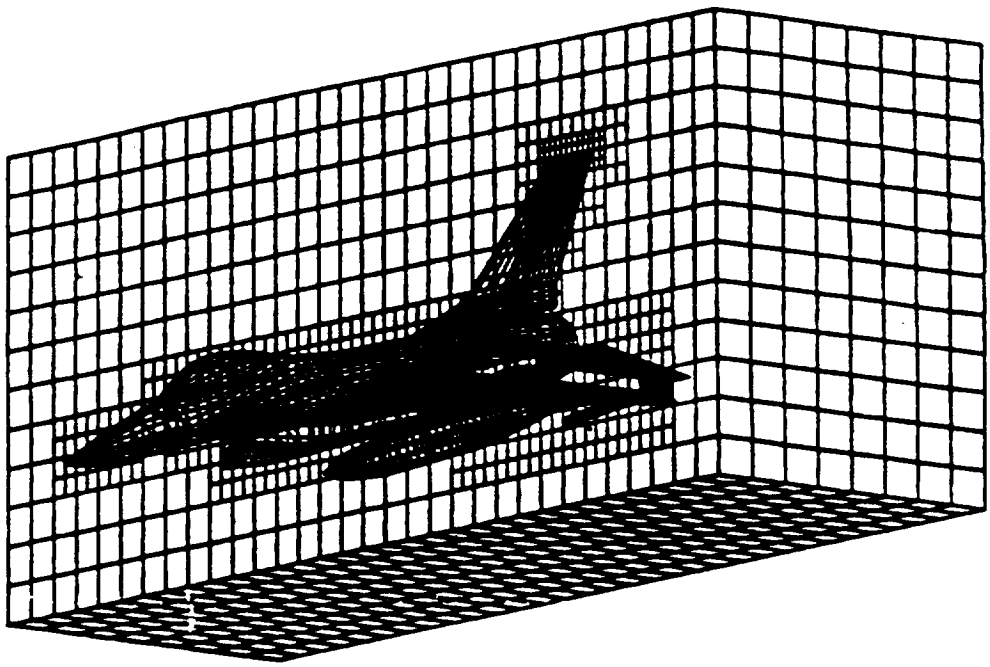


Figure 2.1: Refined Grid Used in TranAir

The code constructs a sequence of grids which are used to solve the problem. The coarsest grid in the sequence comes first. The discretization is generated on this grid and solved with Newton's method. The resulting solution is interpolated to the next (finer) grid and used as an initial guess in using Newton's method on this grid. This process is repeated until the solution is obtained on the finest grid in the sequence.

The sequence of grids can be constructed in two ways. In the *grid sequencing* mode, the final grid is specified by the user in the input deck and the code automatically derefines this grid to generate the coarser grids in the sequence. In the *solution adaptive* mode, the coarsest grid is the same coarsest grid from the sequence above, but successive finer grids are generated internally by the code by performing grid refinements (and derefinements) based on estimates of the local error. In the current implementation of solution adaptive gridding the local errors are based on changes in velocity magnitude from element to element, scaled by an "interest factor" defined by the user to emphasize or de-emphasize the errors.

2–1.4 Boundary Layer Coupling

Viscous effects in transonic flow are approximated by TranAir through invocation of boundary layer coupling. The boundary layer coupling implemented in TranAir is a pseudo-3D approach based on the infinite wing (with sweep and taper) approximation in boundary layer theory. This approximation may be applied to any lifting surfaces in the configuration but it may only be accurate for large aspect ratio wings.

The solution of the boundary layer equations is driven by the pressure field on the configuration surface, as computed at each Newton step in the solution of the inviscid flow equations. The boundary layer solution provides a set of transpiration boundary conditions for the next Newton step of the inviscid solver. The transpiration conditions are damped by taking linear combinations of the current transpiration with that of the previous boundary layer solution using a coefficient defined by the user. The boundary layer equations themselves are pseudo-3D in the sense that they assume spanwise symmetry of the pressure field and solve for a 2D boundary layer on an assumed streamline on the lifting surface. The streamline is defined to be normal to both leading and trailing edges of the lifting surface and is assumed to be a conical arc in between. Cuts in the lifting surface are defined by the user, usually at constant span locations. The surface pressures on these cuts are interpolated to a boundary layer grid defined by the conical arc streamline. After solving the boundary layer equations, the solution is re-scaled (accounting for proper Reynolds number variation with varying chord lengths) and interpolated back to the cuts to define transpirations at network corner points.

Although the boundary layer modeling is somewhat limited by these assumptions, there are a number of user-controllable input options that provide a wider degree of flexibility in approximating viscous effects than the simple nature of the swept-tapered wing approximation might lead one to think. For example, the user specifies both leading edge and trailing edge sweeps. These do not need to correspond to the physical planform of the wing. If the user specified values approximate the streamlines on the surface of the configuration, a more accurate answer may be obtained than by using

the physical planform sweeps. Thus, a feasible modeling approach might be to solve a inviscid problem and trace streamlines on the surface to provide better estimates of the appropriateness of the infinite swept tapered wing approximation and to provide more accurate estimates for the leading and trailing edge sweep parameters.

2–2 PROGRAM ORGANIZATION

TranAir is organized into four programs which are usually run sequentially. These programs are the **Input Processor**, the **Solver**, and the **Output Processor** and a **Binary Converter** program to produce graphical input to the TranAir Graphics Program run on a Silicon Graphics IRIS workstation (see Appendix B). A schematic of the program and data communication is shown in figure 2.2. These modules are described briefly.

2–2.1 Input Processor

The **Input Processor** reads the input data and processes the geometry. Specifically, the following tasks are performed:

- Input data is read in and interpreted for errors.
- Network geometry in the form of corner points (or shape parameters to generate corner points inside the code) is read.
- Certain network abutments are forced, if desired. All network abutments are identified. Special wake parameter locations are determined.
- Consistency of various regions with differing total pressure and temperature is ensured by checking the connectivity of regions separated by the boundary.
- Extent of the computational region is determined for the input geometry (if it is not directly specified).
- Network corner point data is written on a file called MSPTS. This file contains geometry data after the program has forced abutments or preprocessed the geometry using some of the internal preprocessors.
- Data needed by the solver is written on a file (called fort.7 during execution).

2–2.2 Solver

The **Solver** performs the following tasks:

1. Generates the finest predetermined grid based on geometry and user specified controls.
2. Extracts a sequence of grids from the finest grid by derefining everywhere by one level.
3. For each grid, solves the problem using the following steps:
   - 3.1 Generates Boundary Operators.
   - 3.2 Generates Green's function (Poisson solver preconditioner) for the current global grid.
   - 3.3 Obtains initial solution for the current grid.
   - 3.4 Solves the discrete equations using Newton's method.
     - 3.4.1 Generates and decomposes the Jacobian, if needed.
     - 3.4.2 Solves the linear problem via GMRES using the following steps:

       Computes residuals.

       Combines the sparse solver and Poisson solver preconditioners.
     - 3.4.3 Computes a non-linear update.
     - 3.4.3 If not converged returns to 3.4.1.
   - 3.5 If solution adaptive option is chosen, then:
     - 3.5.1 Computes local error estimates.
     - 3.5.2 Computes the next grid in the sequence.
   - 3.6 If not on the final grid, returns to 3.1.
4. Extracts aerodynamic output at panel corner points on the configuration surface.

2–2.3 Output Processor

The **Output Processor** performs the following tasks:

- Reformats the aerodynamic output extracted in the Solver and prints it. A file (called fort.10 during execution) with this information is also generated for plotting purposes. Two files, AELP3DQ and AELP3DG, for use in PLOT3D are also generated.
- Prepares a summary of the configuration forces and moments. A file (during execution, fort.12) with force and moments summary is also generated for plotting purposes.
- Extracts the sectional properties from the network pressure data. Two files (fort.11 and fort.14), with sectional properties are also generated for plotting purposes.

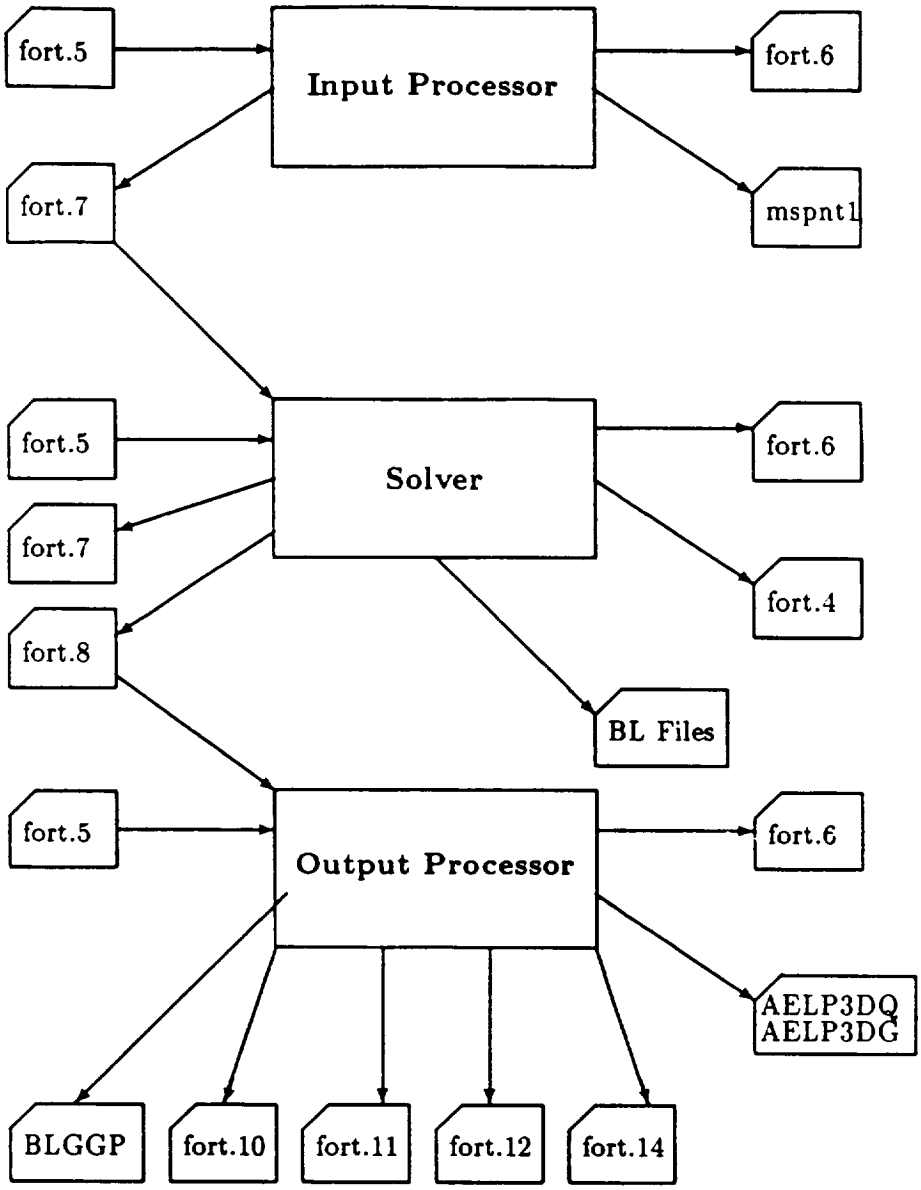


Figure 2.2: TranAir Code Modules and Generic Data Communication Chart

3– OVERVIEW OF AN AERODYNAMIC ANALYSIS

3–1 INTRODUCTION

A successful aerodynamic analysis consists of selecting an appropriate computational model to be used, preparing the input data and job control deck, making a computer run, and finally, interpreting the results.

Selection of the Computational Tool

The first step is to select the appropriate computational tool implementing a given model (Prandtl-Glauert, Full-Potential, Euler, Boundary Layer, Navier-Stokes, etc). The propriety of choosing one approach over another does not always depend upon which model best resembles the physics but rather upon which model is most effective in practice. Tools that can reliably handle complex geometries may not always be available for the most advanced representation of the physics.

Generally, the following questions must be addressed for each of the possible computational tools or approaches that are available.

- What specific aerodynamic information is sought?
- What configuration components are significant in the flow representation? The components that do not have a significant influence on the analysis should be omitted if possible. For example, tails can often be omitted from configurations when the answers are sought for the wing.
- Are there specific flow phenomena that have to be resolved, or can they be approximated by an appropriate mathematical model for which reliable computational tools exist? For example, exhaust plumes usually have significantly different stagnation properties than the freestream and the most appropriate formulation may be the Euler equations. However, in most situations these plumes can be approximated by a region with constant (but different than freestream) total pressure and/or temperature, separated from the outer flow through a boundary. If this is the case, both the inside and outside flow can be modeled using the full-potential approach, each with a different total temperature and pressure.

- To what degree has the computational tool been validated for problems similar to the one you wish to solve?
- What computer costs are expected?
- How much time and effort is required to prepare the input data and interpret the results?

Making a Computer Run

The next step is to make a computer run using an appropriate tool. A successful computer run involves the preparation of input and the execution of the jobs on the appropriate computer. A successful solution run should be free of errors in the geometric modeling and program errors uncovered by the run.

The program may have coding errors or algorithm errors that are uncovered only through consistent and exhaustive application of the code to a variety of cases. This is especially true for large computer programs. It usually takes a significant amount time to uncover and fix such errors. Modeling errors, on the other hand, can be avoided. Diagnostic data to help the user identify many of these modeling errors is generally provided for simpler cases. In more complicated cases, it is necessary to seek help from more experienced users.

Once a successful run has been made, the final step is the analysis of the computed results.

3–2 OUTLINE OF A TranAir RUN

TranAir has been used in a wide variety of aerodynamics analyses [3] through [19]. This section outlines a successful solution run using TranAir.

3–2.1 Defining Configuration Surfaces

In order to solve the underlying boundary value problem in TranAir, it is necessary to define the configuration surface in networks of surface corner points. A large portion of the input data to be specified to the code concerns the geometry of the configuration. In practically all situations of significance, the surface geometry must be defined by a geometry system. In order to define the configuration geometry, the ability to extract surface points from the database of the geometry system is crucial.

Preparing Network Layout

Once the geometry definition is ready, the next step is to divide the configuration surface lofts into topologically simple subsurfaces. A subsurface is called a network and is represented by a rectangularly indexed array of surface points. The network of surface corner points also forms networks of panels which may be regarded as the locally flat approximations to the configuration boundary used by the program in imposing the specified boundary conditions. The issues involved in this process

are briefly discussed below. (Geometry and boundary condition specification are described in more detail in section 4–6.)

TranAir allows considerable flexibility in defining the networks. Specifically, the inter-network connectivity may span over full or partial network edges, and the surface point density of the neighboring networks need not be the same. The order in which the networks are input is completely arbitrary.

Network boundaries should always occur at slope discontinuities of the configuration. Beyond this, the primary factors determining the network layout are the logical breaks in the configuration following the global topology of the configuration. The secondary factor determining the network layout is the boundary condition (see section 3–2.2). TranAir allows only one boundary condition type per network and, hence, the division of the configuration surface into the networks also depends on the boundary conditions. There are several rules to be followed in preparing network layouts.

- Network boundaries should occur at logical topological changes in the configuration.
- Network boundaries must occur at significant discontinuities in slopes of configuration surfaces (or modeling accuracy will suffer, since the slopes are smoothed within a network).
- Different boundary condition types can only be specified on separate networks.

To facilitate postprocessing, the following additional rules should also be followed.

- Networks should be defined consistently to produce outward surface normals which point into the flow (see section 4–6).
- The number of networks should be as small as possible.
- To keep account of the networks, appropriate network layouts should be drawn.

A network layout for a typical configuration (wing/body) is shown in figure 3.1.

Extracting Surface Points

Based on the network layout, surface points must be extracted from the appropriate lofts and ordered into networks. During this task, it is also necessary to decide on the proper surface point density and continuity across network edges. The primary purpose of discretizing the configuration is to provide an accurate representation of the boundary so that discrete operators involving the boundary conditions can be built correctly. The density of the surface points is also used in controlling the density of the volume grid in the vicinity of the boundary. It is generally safe to assume that a denser representation of the boundary should be used where the surface curvature is high or where greater output resolution is desired. Since the cost of a TranAir run is relatively insensitive to the number of surface points (or panels), it is recommended that surfaces be represented more densely than in typical panel method applications.

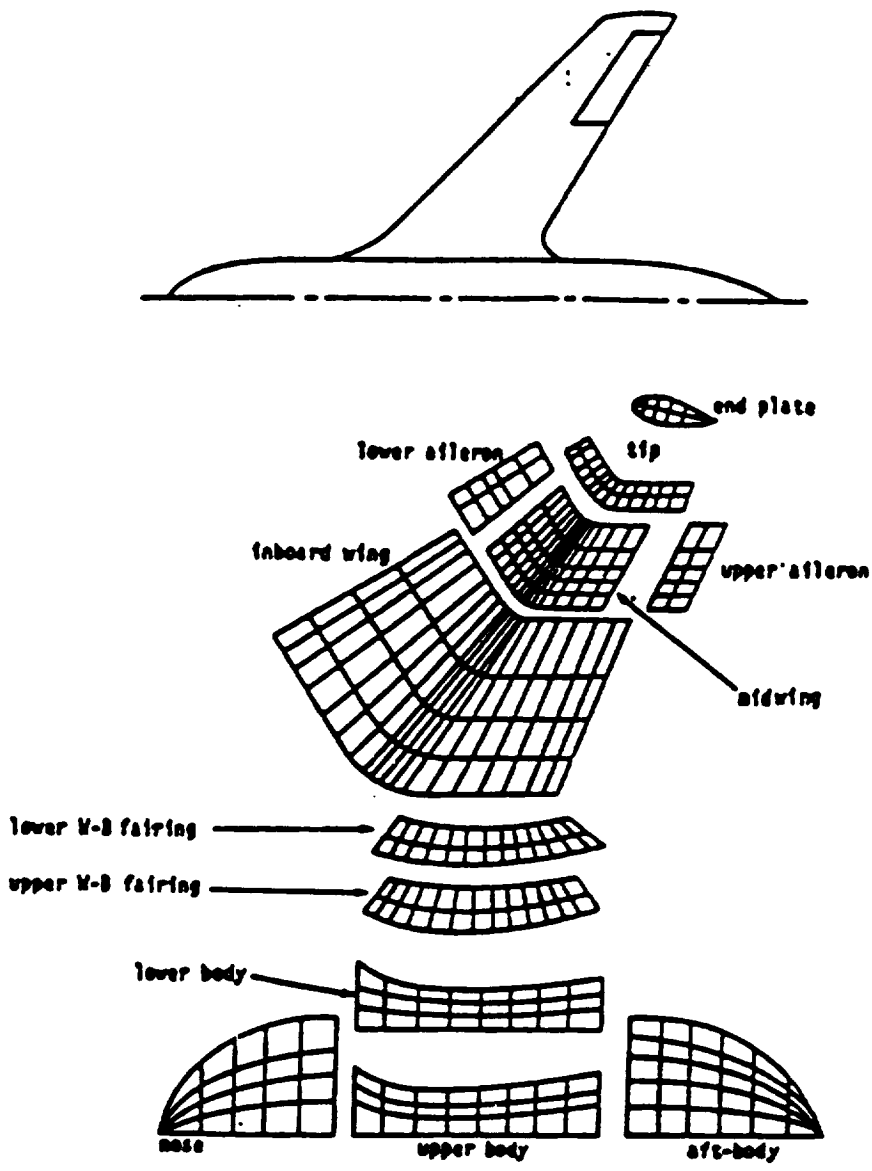


Figure 3.1: Network Layout for a Typical Wing/Body Configuration

3–2.2 Selecting Boundary Conditions

Based on the configuration and the definition of the flow problem, the user must assign boundary conditions to each network in the configuration. These are assigned by specifying one or more input parameters associated with each network or group of networks. For commonly used boundary conditions, a single integer may select from a wide variety of boundary conditions. Some of the more commonly utilized boundary conditions are illustrated in figure 3.2. These boundary conditions are used to simulate the following:

- Impermeable surfaces
- Wakes
- Exits and Bases
- Inlets
- Thin Lifting Surfaces
- Wind Tunnel Walls

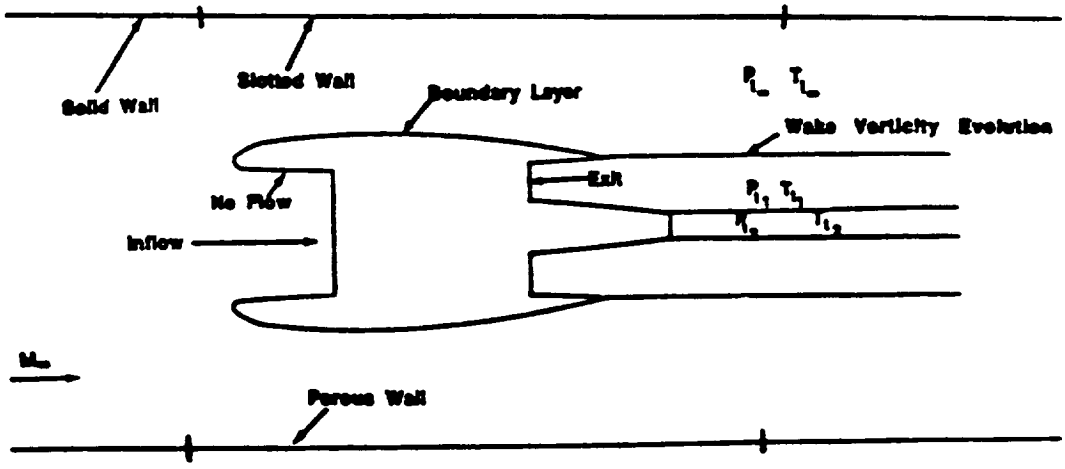


Figure 3.2: Typical Boundary Conditions used in TranAir

3–2.3 Establishing Grid Generation Controlling Parameters

In order to obtain the numerical solution to the nonlinear, full-potential equation, a volume grid must be generated. The grid is rectangular, *non-surface conforming*, and is locally refined. Unlike most CFD codes, TranAir generates the grid *internally*. There are certain issues that must be understood regarding a TranAir grid so that the user can control it to the user's advantage.

The grid in TranAir need only extend far enough to satisfy these two criteria. First, the grid must enclose the entire configuration. Second, if the freestream is subsonic, it should also enclose any supersonic bubbles.

For supersonic freestream problems, the grid needs to extend far enough away from the configuration in the y and z directions that Mach lines emanating from the most forward portions of the configuration do not reflect back on to a rearward portion of the configuration. (For supersonic freestream conditions, TranAir does not yet employ a far field boundary condition, and the Mach lines simply reflect from the tops and sides of the computational grid.)

Grid refinement performed by the code is controlled in one of two ways. First, near the boundary, the grid is refined based on a criterion that makes the size of the grid box "close" to the size of a panel in the vicinity of that box. This means the program will typically make the grid finer where the paneling density is higher. Second, the limits on the physical size of the box can be directly specified. These limits are specified as distributions using a relatively small number of parameters.

TranAir uses a sequence of successively refined grids. If the **grid sequencing** option is chosen then this sequence is constructed using all the specifications provided to construct the grid. If the **solution adaptive gridding** option is chosen then certain additional input is desirable in helping the program construct the sequence of grids.

3–2.4 Controlling the Iterative Solution Process

The user may provide input to guide the iterative solution process on each grid. These inputs include limits on the total number of iterations the code may perform to achieve convergence, the number of digits of reduction required to achieve convergence, and some additional parameters that may be used to increase solution efficiency (drop tolerance and viscous damping coefficient).

3–2.5 Executing the Code

Once the input data is prepared, the TranAir code can be executed in one of three modes.

In the first mode, the user can perform a *data check* wherein the program input is interpreted and network edge abutments are forced and identified. In this type of data check, the following steps can be performed.

- Ensure that all the input has been read in without errors.

- Ensure that all surface upper normals and associated material properties are consistent.

- Ensure that all abutments have been properly made. If any abutments are forced, ensure that the program has moved the correct set of points and that the distortion produced during this process is acceptable (or at least understood).

In the second mode, a type of data check can be made in which only the grid is constructed so that the grid can be examined before launching the full solution. This data check is only meaningful if the grid sequencing option is exercised.

In the third mode, the code can be executed to obtain the full solution.

There are several ways to restart a TranAir solution.

- An iterative solution may be continued to improve convergence.

- An iterative solution may be continued to obtain solution at a slightly perturbed flow condition (although this may be of questionable utility if solution adaptive gridding has been used to obtain the first solution).

- An iterative solution may be continued to obtain a solution on a perturbed grid.

- The results from an existing solution may be post processed.

Note that if a boundary layer model is included in the configuration, the case may not be restarted.

3–2.6 Selecting Output Options

In the course of obtaining a complete solution, the code produces a large number of files. The files are for restart, printout and data for processing on a graphics workstation. By default, the code will produce the surface aerodynamic properties at network corner points. Additional output (such as the sectional properties, forces and moments summaries, and aerodynamic data at panel center points) may also be obtained by selecting appropriate controlling parameters.

4– PROGRAM INPUT

4–1 INTRODUCTION

All the input required to define 3D steady flow analysis for TranAir are given in this section. The major program modules and the different input data files and contents are summarized below. Because of the many program options which are available and the large number of ways they may be employed, the input data has been grouped together into data blocks. An index is provided to help access the data blocks. An example input is given for a simple wing-body configuration. The conventions used in the layout of the data are summarized and should be reviewed before starting to assemble the inputs. The major portion of this section contains descriptions of the data blocks. In particular, it contains input card images, input data descriptions, default values, detailed explanations, and some examples.

To run a *datacheck* or a *complete solution* requires that the input data be supplied to the TranAir input module (`fdinp`). This complete set of data is distributed as required to the other TranAir modules.[1] To add flexibility in assembling data, input may be spread over several files. Representative possible input files and their contents are listed below. In the following table, input for a particular module is denoted by the letters `i`,`s`, or `o` followed by the extension `.inp`; the string `$CASE` may be thought of as a shell variable under UNICOS which is used to distinguish input for one run from another.

[1]For all of the modules, user-input appears as `stdin` which normally maps to `fort.5` under UNICOS. For UNICOS scripts running TranAir, standard input should be re-directed to the user-provided input file for each module. In the case that no user-provided input exists for a particular module, it is recommended that input be redirected to `/dev/null`.

4–1.1

File	Contents
`$CASE.i.inp`	All inputs.
or	
`$CASE.i.inp`	All inputs except for `$CASE.poi` data.
`$CASE.poi`	Configuration definition (described with x, y, z) and boundary conditions (all **$POInt** data).
or	
`$CASE.i.inp`	All inputs except for provided in the files below.
`$CASE.poi`	Configuration definition (described with x, y, z) and boundary conditions (all **$POInt** data).
User-specified file	Any file of user-specified data (see the description of the keyword **$FIL** in this section).

To *restart* an existing run for revisions to portions of the original input affecting convergence parameters or freestream conditions requires restarting the TranAir *solver module* (`fdsol`). The input file and possible contents are listed below.

File	Contents
`$CASE.s.inp`	All previously input data for a complete solution with appropriate modifications.
or	
`$CASE.s.inp`	Only inputs for the solver module with appropriate modifications.

To *restart* an existing run for revisions to any output process requires restarting the TranAir *output module* (`fdout`). The input file and possible contents are listed below.

File	Content
`$CASE.o.inp`	All previously input data for a complete solution with appropriate modifications and/or additions.
or	
`$CASE.o.inp`	Only inputs for the output module with appropriate modifications and/or additions.

The data blocks accepted by `fdsol` and `fdout` are summarized in table 4.2.

4–1.1 Data Blocks

So that the experienced user may rapidly access the different data blocks, an index has been provided in table 4.1. The data blocks are identified by a **$** sign and three letters (e.g., **$POI** for points data block input). This is the input used in the program to identify a data block. The data blocks have been organized into six major categories reflecting major program features. Input data is required from some of the first four categories. The last two categories are optional and depend on the problem.

To help identify the input data, several styles are used in the input descriptions.

- Capital Letters - keywords associated with a data block (e.g., **$POI** for points).
- Italics - parameter names of numerical data.
- Standard Text - comment lines which may be included or excluded for input.

Each input line has a label and number associated with the block of data and the line order within the block. Comment lines do not have any line labels.

For each group of data blocks, the input data is first explained in generic terms. This is followed by an illustrative example of the data block card images as they would appear in the input dataset. Next, the description of the parameters and suggested values and/or defaults are given. Any special notes are provided at the end.

Default Parameters

TranAir sets defaults for many input parameters. There are data blocks that can be omitted if it is desired that the program should use default values for the parameters defined via those data blocks. However, if any data blocks are provided as input, then all the parameters in that data block should be provided as input. This caution is provided primarily since the FORTRAN read statements interpret blank spaces as zero values for the parameters in the list and not their default values. Most inputs will include data using the **$TIT**, **$MAC**, **$ANG**, **$POI**, **$BOX**, **$TOL**, **$ITE**, **$REF**, **$PLO**, and **$END** keywords. Other commonly used keywords are **$EAT**, **$PEA**, **$SYM** and, for grid control and solution adaptive gridding, **$LBO** and **$ADA**.

4–1.2 Sample Input: Wing-Body Configuration

To help explain the format of the input data, a simple wing-body analysis TranAir input is used as an example (see figure 4.1 and figure 4.2). This case represents a coarsely paneled wing-body with a simple field grid. This short checkout case is used for validating TranAir. This has been set up with all the input parameter names given as comments above the input value. Recommended parameter values have been marked with an * sign after the parameter name. Many of the marked values have been established after numerous studies to determine their value.

Keyword	Page	Section	Information Specified by the Data Block.
BASIC			
$TIT	46	4-2	Title and name.
$DAT	46	4-2	Controls program execution and miscellaneous options.
$FIL	46	4-2	Alternate file name for a portion of the input data.
$END	46	4-2	End of input data.
FLOW PROPERTIES			
$SYM	50	4-3	Symmetry in the problem.
$MAC	50	4-3	Mach number.
$ANG	50	4-3	Angle of attack.
$SID	50	4-3	Sideslip angle.
$MAT	54	4-4.1	Total pressure and temperature for different regions.
$THE	57	4-4.2	Associate material properties to networks.
SOLUTION CONTROL			
$ITE	60	4-5.1	Solution iteration control.
$BOX	63	4-5.2	Computational domain and global grid.
$TOL	67	4-5.3	Grid refinement controls for the entire computational domain.
$LBO	67	4-5.3	Grid refinement controls for the "specified hexahedral regions."
$SBO	67	4-5.3	Grid refinement controls for the "specified regions of interest."
$NRE	67	4-5.3	Grid refinement control keyed to networks.
$ADA	78	4-5.5	Solution adaptive grid controls.
$BOU	83	4-5.6	Boundary layer analysis controls.
CONFIGURATION DEFINITION & BOUNDARY CONDITIONS			
$POI	98	4-6.2	Corner points.
$TRA	120	4-6.4	Trailing wake networks (program generated).
$QUA	122	4-6.5	Quadrilateral networks (program generated).
$CIR	124	4-6.6	Circular shaped networks (program generated).
$REA	127	4-6.7	Rotate, scale, or translate an existing network.
ABUTMENTS			
$PEA	133	4-7.1	Abutment forcing.
$EAT	138	4-7.2	Tolerance distance for abutment identification.

Table 4.1: Functional List of Keywords

Keyword	Page	Section	Information Specified by the Data Block.
OPTIONAL OUTPUT			
$PRI	141	4-8.1	Printout control.
$REF	143	4-8.2	Reference lengths and areas for force and moment coefficients.
$FOR	146	4-8.3	Force and moments "summary" controls.
$SEC	149	4-8.4	Sectional properties.
$SUR	156	4-8.5	Surface properties file.
$FIE	160	4-8.6	Field properties file.

Table 4.1: Functional List of Keywords (continued)

Keyword	Processed by:
$MAC	**fdsol**
$ANG	**fdsol**
$ITE	**fdsol**
$SID	**fdsol**
$REF	**fdout**
$FOR	**fdout**
$SEC	**fdout**
$SUR	**fdout**
$FIE	**fdout**

Table 4.2: Keywords Recognized by **fdsol** and **fdout**

The input data shown as two files (**swb.i.inp** and **swb.poi**) illustrates a practical way to work with the inputs for a medium or large configuration. By placing the **$POI** data in a separate file, the remainder of the inputs (now much shorter) can be edited, printed, and saved with greater convenience.

```
$TITle                                                                  tit1
SIMPLE WING-BODY (87 PANELS), MACH NO. = 0.60, ALPHA = 4.0 DEG.         tit2
TWO CYCLE ADAPTIVE FIELD GRID WITH SOME FOCUS ON THE WING.              tit3
! SAARIS  865-6150  M/S 7H-94
!  * after input parameter denotes recommended value.
$DATacheck                                                              dat1
! ndtchk = 0.-solution, 1.-stop after initial grid, 2.-stop after abutment
=ndtchk   imspt*    ilnl*     ipcut*
0.        0.        0.        0.                                        dat2
! *** Flow Properties ***
$SYMmetry, planes of flow                                               sym1
=xzpln    yzpln
1.        0.                                                            sym2
$MACh number                                                            mac1
=amach    linr*     amfgas*   fsvmi*    upwnd*
.6        0.                            1.                            mac2
$ANGle of attack                                                        ang1
=alpha
4.                                                                      ang2
! *** Solution Control ***
$ITEration                                                              ite1
=niter*   tol*      nvis*      dropt*    nsrch*    njac*
500.      7.        0.         .001      60.       2.                  ite2
$BOX                                                                    box1
=xi       xf        nx
-20.      120.      28.                                                 box2
=yi       yf        ny
0.        45.       9.                                                  box3
=zi       zf        nz
-20.      25.       9.                                                  box4
$TOLerance for global grid                                              tol1
=adpfac   panfac    dxmin     dxmax     epsf*
1.        1.        2.        7.        1.51                          tol2
```

Figure 4.1: Sample Input Data for `swb.i.inp`

```
$LBO - Special region of Interest                                    lbo1
=adpfac   panfac    dxmin     dxmax     lbnam
1.        1.        1.        7.        wing                         lbo2
=x1       y1        z1        reldx1
20.       9.        -10.      1.                                     lbo3
=x2       y2        z2        reldx2
71.       9.        -10.      1.                                     lbo4
=x3       y3        z3        reldx3
34.       28.75     -10.      1.                                     lbo5
=x4       y4        z4        reldx4
84.       28.75     10.       1.                                     lbo6
=x5       y5        z5        reldx5
20.       9.        10.       1.                                     lbo7
=x6       y6        z6        reldx6
71.       9.        10.       1.                                     lbo8
=x7       y7        z7        reldx7
34.       28.75     10.       1.                                     lbo9
=x8       y8        z8        reldx8
84.       28.75     10.       1.                                     lbo10
$ADAptive grid controls                                              ada1
=ncycle
2.                                                                   ada2
=maxgrd   nbxtgt    ratio1*   ratio2*   fracrf*   fracdf*
5.        6000.     0.40      0.80      0.20      0.40               ada3
=numlbo
0.                                                                   ada4
=maxgrd   nbxtgt    ratio1*   ratio2*   fracrf*   fracdf*
1.        6000.     0.40      0.80      0.20      0.40               ada3
=numlbo
0.                                                                   ada4
! *** Configuration Definition and Boundary Conditions ***
$FILe for the $POInts data                                           fil1
swb.poi                                                              fil2
$TRAiling wakes from wing and body                                   tra1
=kn
3.                                                                   tra2
=kt
18.                                                                  tra3
=inat     insd      xwake     twake                    netname
WINGA     1.        140.      .0                       WINGWK    tra4
BODYL     3.        140.      .0                       BODYLWK   tra4
BODYU     3.        140.      .0                       BODYUWK   tra4
```

Figure 4.1: Sample Input Data for `swb.i.inp` (continued)

```
! *** Abutments ***
$PEA - partial of full network edge abutment                          pea1
=nfpa     iopfor
2.        0.                                                          pea2
=nne      peatol
2.        .005                                                        pea3
=nn       en        epinit    eplast
WINGA     4.        6.        1.                                      pea4
BODYL     2.        5.        9.                                      pea4
2.                                                                    pea3
WINGA     4.        6.        11.                                     pea4
BODYU     4.        7.        3.                                      pea4
$EAT - liberalized abutments                                          eat1
=epsgeo   igeois    igeout*   nwxref*   triint*
.005      0.        0.        0.        0.                          eat2
! *** Optional Output ***
$PRIntout options                                                     pri1
=iprpan*
0.                                                                    pri2
=iprmus   iprcen    iprcrn    iprfm0
1.0       .0        1.        2.                                      pri3
$REFerences for accumulated forces and moments                        ref1
=xref     yref      zref
46.       0.        0.                                              ref2
=sref     bref      cref      dref
2400.     60.       40.       90.                                     ref3
$SECtional properties                                                 sec1
=numgrp
1.                                                                    sec2
*CUT and reference printout for sectional properties                  sec9
=optcrd   optmrp    iprtnf    iprtpp*   isecpr*   ixyzop    reflen
1.        1.        0.        0.                            30.       sec10
=numcut
2.                                                                    sec11
=xc       yc        zc        xcn       ycn       zcn
.0        15.       .0        .0        1.        .0                sec12
=chrd     refrac
          .25                                                         sec13
.0        25.       .0        .0        1.        .0                sec12
          .25                                                         sec13
$END of TranAir inputs
```

Figure 4.1: Sample Input Data for `swb.i.inp` (continued)

```
$POInts - wing-body impermeable surface boundary conditions                poi1
=kn
4.                                                                         poi2
=kt
1.                                                                         poi3
=nm        nn                                                   netname
11.        3.                                                   WINGA     poi4
=X(1,1)    Y(1,1)     Z(1,1)     X(1,2)     Y(1,2)     Z(1,2)
   69.4737    9.2105     0.0000    63.7818    9.5807     0.7831           poi5
   53.7705    9.7741     1.9747    37.2370    9.4677     1.9614           :
   30.2755    9.1527     0.8173    29.4086    9.1127    -0.0025           :
   30.2800    9.1529    -0.8192    37.2409    9.4679    -1.9617
   53.7722    9.7741    -1.9746    63.7823    9.5807    -0.7830
   69.4737    9.2105     0.0000
   76.6660   20.0000     0.0000    70.8460   20.1781     0.7831
   60.7679   20.2711     1.9747    44.3403   20.1238     1.9614
   37.4878   19.9722     0.8173    36.6347   19.9530    -0.0025
   37.4922   19.9723    -0.8192    44.3441   20.1238    -1.9617
   60.7695   20.2711    -1.9746    70.8465   20.1781    -0.7830
   76.6660   20.0000     0.0000
   83.3330   30.0000     0.0000    77.3943   30.0000     0.7831
   67.2541   30.0000     1.9747    50.9248   30.0000     1.9614
   44.1733   30.0000     0.8173    43.3330   30.0000    -0.0025
   44.1776   30.0000    -0.8192    50.9286   30.0000    -1.9617
   67.2557   30.0000    -1.9746    77.3948   30.0000    -0.7830
   83.3330   30.0000     0.0000
       6.        2.                                               WINGTA    poi4
   83.3330   30.0000     0.0000    77.3943   30.0000     0.7831           poi5
   67.2541   30.0000     1.9747    50.9248   30.0000     1.9614           :
   44.1733   30.0000     0.8173    43.3330   30.0000    -0.0025           :
   83.3330   30.0000     0.0000    77.3948   30.0000    -0.7830
   67.2557   30.0000    -1.9746    50.9286   30.0000    -1.9617
   44.1776   30.0000    -0.8192    43.3330   30.0000    -0.0025
      11.        3.                                               BODYL     poi4
    0.0000    0.0000     0.0000     3.0000    0.0000    -3.4147           poi5
   10.0000    0.0000    -6.0000    20.0000    0.0000    -8.0000           :
   29.4086    0.0000    -9.1127    37.2409    0.0000    -9.6690           :
   53.7722    0.0000    -9.9715    63.7823    0.0000    -9.6126
   69.4737    0.0000    -9.2105    80.0000    0.0000    -8.0000
   90.0000    0.0000    -6.0000
    0.0000    0.0000     0.0000     3.0000    2.4142    -2.4149
   10.0000    4.2421    -4.2432    20.0000    5.6561    -5.6576
   29.4086    6.4428    -6.4445    37.2409    6.8363    -6.8377
   53.7722    7.0506    -7.0513    63.7823    6.7970    -6.7973
   69.4737    6.5128    -6.5128    80.0000    5.6569    -5.6569
   90.0000    4.2427    -4.2427
```

Figure 4.2: Sample Input Data for **swb.poi**

```
   0.0000     0.0000     0.0000     3.0000     3.4147    -0.0009
  10.0000     6.0000    -0.0016    20.0000     8.0000    -0.0022
  29.4086     9.1127    -0.0025    37.2409     9.4679    -1.9617
  53.7722     9.7741    -1.9746    63.7823     9.5807    -0.7830
  69.4737     9.2105     0.0000    80.0000     8.0000     0.0000
  90.0000     6.0000     0.0000
      11.         3.                                              BODYU    poi4
   0.0000     0.0000     0.0000     3.0000     3.4147    -0.0009          poi5
  10.0000     6.0000    -0.0016    20.0000     8.0000    -0.0022          :
  29.4086     9.1127    -0.0025    37.2370     9.4677     1.9614          :
  53.7705     9.7741     1.9747    63.7818     9.5807     0.7831
  69.4737     9.2105     0.0000    80.0000     8.0000     0.0000
  90.0000     6.0000     0.0000
!For brevity, columns 2 and 3 have been omitted.
$POInts - bodybase with base boundary condition                       poi1
=kn
1.                                                                    poi2
=kt
5.                                                                    poi3
=nm        nn                                                         netname
5.         2.                                                         BODYB    poi4
=X(1,1)    Y(1,1)     Z(1,1)     X(1,2)     Y(1,2)     Z(1,2)
   90.0000     0.0000     6.0000    90.0000     4.2427     4.2427          poi5
   90.0000     6.0000     0.0000    90.0000     4.2427    -4.2427          :
   90.0000     0.0000    -6.0000                                           :
   90.0000     0.0000     0.0000    90.0000     0.0000     0.0000
   90.0000     0.0000     0.0000    90.0000     0.0000     0.0000
   90.0000     0.0000     0.0000
$POInts - body to wing carry over wake                                poi1
1.                                                                    poi2
20.                                                                   poi3
4.         2.                                                         AWBW     poi4
   69.4737     9.2105     0.0000    80.0000     8.0000     0.0000          poi5
   90.0000     6.0000     0.0000   140.0000     6.0000     0.0000          :
   69.4737     9.2105     0.0000    80.0000     9.2105     0.0000          :
   90.0000     9.2105     0.0000   140.0000     9.2105     0.0000
$POInts - Sample flow properties                                      poi1
1.                                                                    poi2
6.                                                                    poi3
2.         3.                                                         poi4
31.6       20.        0.         38.3       30.        0.              poi5
71.6       20.        -.01       88.3       30.        -.01            :
71.6       20.        +.01       88.3       30.        +.01            :
```

Figure 4.2: Sample Input Data for swb.poi (continued)

4–1.3 Input Conventions

In preparing input data, it is useful to know the basic conventions used for the input data. The following are the *important* input conventions.

Dimensional Units

All coordinate numbers and related dimensional data are defined in the same units. Thus, if a configuration is defined in inches, the input reference area must be in square inches.

Density is normalized to freestream, thus making ρ_∞=1.0 and velocities are referenced to the freestream velocity. The user may define the freestream velocity, but it is assumed to be unity if no user input has been provided to change its value.

Note that these units differ somewhat from other conventional definitions (such as in the program PLOT3D, where freestream velocity is normalized to the freestream Mach number). For purposes of communication with other codes (in particular PLOT3D), the output files for PLOT3D, AELP3DG, and AELP3DQ have been rescaled to PLOT3D conventions.

Number Format

All numbers are assumed to be decimal numbers, represented in (6E10.0) format. This format includes the format (6F10.0) and is used to specify most data. It uses a maximum of six numbers per line, with ten spaces for each number. Any exceptions to this format are noted when they apply.

Data Blocks

All input data are defined in data blocks, some of which are optional. A data block represents a complete set of data. Because of the amount of data to be defined, some data blocks contain sub-blocks. If a sub-block is input, all its internal data must be defined.

Order of Data Blocks

Input data blocks between the first (**$TITle**) and the end (**$END**) of the data file may be defined in any order. The recommended order for the data blocks is the order in which they are documented. If this order is retained, the blocks can be easily checked against the documentation. In the course of processing the input files, the TranAir input processor makes several passes over the input data, extracting in each pass keywords which can be processed and leaving for the next pass those keywords which may have a dependency on some other set of keywords. Thus the user need not be concerned with the order of the keyword blocks in the input file. However, within each keyword block, the data order must be as described in the following subsections.

Special Column 1 Symbols

Five symbols are used in *column 1* to identify particular input data features:

Symbol	Description
$	Marks the beginning of a data block.
*	Identifies the start of a data sub-block (block of data within a data block) in the configuration forces and moments summary, in the sectional properties input sections, and wind tunnel wall.
=	Introduces a comment line; generally used for defining symbols above a line of input, or for any comment that is useful to the program user.
!	Introduces a comment line; it is the same as the "=" sign.
# D L	Used to duplicate "D" times the following "L" lines; D and L are input as free-field data separated by a space.

Comments are useful in defining input parameters and qualifying input data.

4–2 BASIC ($TIT, $FIL, $DAT, $END)

This basic group of input keywords provides control for some miscellaneous but important functions when running a TranAir case. This includes:

- **$TIT**le - run title.
- **$DAT**acheck - controls program execution and miscellaneous options.
- **$FIL**e - controls the inclusion of additional files into `$CASE.i.inp`.
- **$END** - indicator of last input line for the `$CASE.i.inp` file.

Some of the features and parameters of these data blocks are described below:

$DATacheck

- *ndtchk* - Allows the user to stop TranAir at various stages of the solution process. These stages of program execution are:
  - *Data Interpretation and Abutment Analysis* (*ndtchk*=2) - In the Input Processor, the program checks the readability of the inputs. It also searches for network edge abutments and prints out a detailed summary of the result. The user should look for erroneous abutments, incomplete abutments, excessive movement of network panel edges from matching panels, etc. It is recommended that the abutment analysis be performed on all runs analyses which contain new or modified geometry.
  - *Grid* (*ndtchk*=1) - In the Solver module, the program reads the output from the input processor and generates a grid. This grid should be viewed to ensure that the special regions of interest that have been specified have provided the desired grid. This step makes sense in a run in which the sequence of grids is prescribed (grid sequencing). This step is also recommended for cases with a complicated geometry.
  - *Solution* (*ndtchk*=0) - In this case, the program completes the solution and provides all the output.
- *imspt* - A flag indicating whether the configuration geometry, after being modified by the abutment processing, should be written to an output file (`MSPTS`).
- *ilnl* - A flag indicating whether the surface solution file should be written twice; with the first copy containing the linear solution and the second containing the non-linear solution. (This feature is obsolete and it is recommended that it not be used.)
- *ipcut* - A flag indicating whether tests should be made to discover if some portion of the configuration cuts some other portion, other than along abutment lines. It is highly recommended that this option be turned on when performing a datacheck with new configurations.

- *ngstop* - A flag indicating that regardless of how may solution adaptive grids or cycles have bee specified, stop the solution process when the *ngstop*-th grid has been completed.

$FILe

- *fname* - Name of a file containing additional input data.

As a convenience to help the user manage the input data file (`$CASE.i.inp`), an option has been introduced to insert files of data into the input data file. This option can be used at any point in the input process to insert an additional data file (`$CASE.ext`). It is the responsibliity of the user to assure that the named files are local to the executing directory of the TranAir modules.

A common use of this option is to separate the large amount of input data associated with the network geometry (**$POI**) from the remainder of the input data. This has two benefits. In the first, `$CASE.i.inp` is much easier to edit and print. In the second, the `$CASE.poi` file with a few changes can be placed in a form to view and revise the surface network geometry.

$END

Required to be the last input in *any* TranAir input file.

Card Images

$TITle (First data block)								**tit1**
title	-	-	-	-	-	-	-	**tit2**
user	-	-	-	-	-	-	-	**tit3**
$DATacheck (Second data block)								**dat1**
ndtchk	*imspt*	*ilnl*	*igeom*	*ipcut*	*ngstop*	-	-	**dat2**
$FILe - Insert a file (May be placed where required)								**fil1**
fnam	-	-	-	-	-	-	-	**fil2**
$END - Required termination line (Last data block)								**end1**

Input Data Description

Card	Field	Name	Description
tit1	all	*icard*	**$TIT** keyword.
tit2	all	*title*	Title of the run. Format (**A**).
tit3	all	*name*	User name and address. Format (**A**).
dat1	1	*icard*	**$DAT** keyword.
dat2	1	*ndtchk*	Parameter indicating the specific step in the program where it is to be stopped. = 0.0 After final solution. = 1.0 After grid obtained. (Finest grid, if grid sequencing option. Initial grid, if solution adaptive grid option.) = 2.0 Data interpretation and abutment analysis.
	2	*imspt*	Option to output a file (**MSPTS**) containing all the network points after the abutment processing. = 0.0 No. = 1.0 Yes.
	3	*ilnl*	Option to output the linear solution result after the first linearization in addition to the nonlinear solution. = 0.0 No. = 1.0 Yes.
	4	*igeom*	Option to check geometry for excessive edge point or panel normal movement after abutment processor finishes. = 0.0 No check. = 1.0 Perform check.
	5	*ipcut*	Option to check whether any panel in the configuration intersects any other. This check is fairly costly (approximately 100 Cray X-MP seconds for a 15,000 panel case). = 0.0 No. = 1.0 Yes.
Card	Field	Name	Description
fil1	all	*icard*	**$FIL** keyword.
fil2	1	*fnam*	Name of the file to be inserted into the input. (The file must be local to the working directory of the TranAir module.
end1	all	*icard*	**$END** Keyword.

Default Values

All of the keywords in this category (except **$END**) may be omitted from the input data. Default values for the data described in this data block are given below.

Data Block	Parameter	Default Value
$TIT	*title*	"My Run"
	name	"My Name"
$DAT	*ndtchk*	0.
	imspt	0.
	ilnl	0.
	igeom	0.
	ipcut	0.

4–3 FLOW PROPERTIES

The flow properties group of input controls major flow features for the problem. This includes:

- **$SYMmetry** - unsymmetric, one or two planes of the flow symmetry.
- **$MACh** number - freestream Mach number and miscellaneous options.
- **$ANGle** of attach and
- **$SIDeslip** angle - direction of freestream.

Some of the important features for the **$MAC** data block are described below:

- *linr* - Linear flow flag. For subsonic freestream cases, TranAir can be run in a mode in which the linear Prandtl-Glauert equation, the linear approximation of the full potential equation, is solved.
- *amfgas* - Fictitious gas Mach number. In some situations, the program calculates solutions with very large velocities approaching vacuum pressures. This means that the flow density goes to zero and leads to unrealistic solutions and numerical problems. To avoid this behavior, the density calculation in the code is modified to change the isentropic behavior of the gas for velocities above the fictitious gas Mach number.
- *fsvmi* - Freestream velocity magnitude. If zero, redefined to be unity.
- *upwnd* - Upwinding flag. Selects either density upwinding or mass flux upwinding. It is recommended that the viscosity damping flag *nvis* in **$ITE** should always be *zero* if mass flux upwinding has been selected.

Card Images

$SYMmetry								**sym1**
xzsym	*xysym*	-	-	-	-	-	-	**sym2**
$MACh Number								**mac1**
amach	*linr*	*amfgas*	*fsvmi*	*upwnd*	-	-	-	**mac2**
$ANGle of Attack (in degrees)								**ang1**
alpha	-	-	-	-	-	-	-	**ang2**
! Use the following only if the flow is unsymmetric (*xzsym*=0.0 and *xysym*=0.0).								
$SIDeslip Angle (in degrees)								**sid1**
beta	-	-	-	-	-	-	-	**sid2**

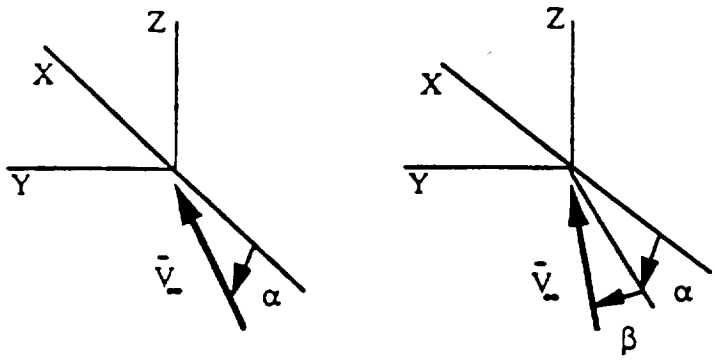


Figure 4.3: Reference Coordinate System and Freestream Flow Direction

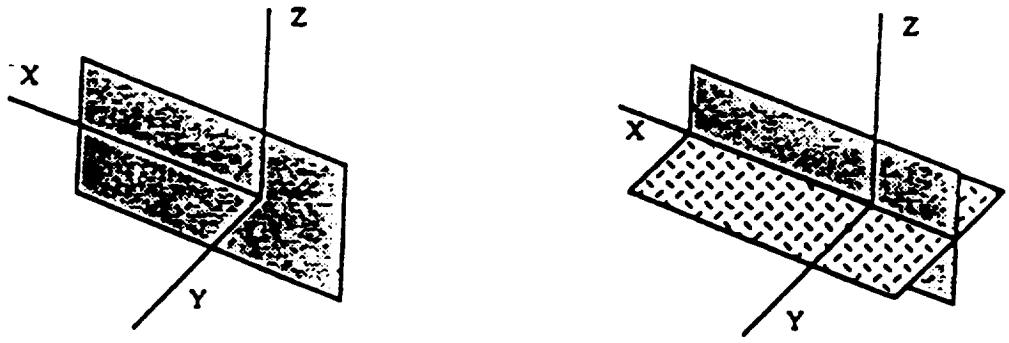


Figure 4.4: Symmetry

Input Data Description

Card	Field	Name	Description
sym1	all	*icard*	Keyword **$SYM**.
sym2	1	*xzsym*	Parameter to specify flow symmetry about xz plane. = 0.0 No symmetry. = 1.0 xz symmetry or symmetry about y=0 plane.
	2	*xysym*	Parameter to specify flow symmetry about xy plane. = 0.0 No symmetry. = 1.0 xy symmetry or symmetry about z=0 plane. This is meaningful only if *xzsym* is 1.
mac1	all	*icard*	**$MAC** Keyword.
mac2	-1	*amach*	Mach number of the freestream. **Note:** If *amach* is defaulted to zero by omitting the keyword or is input as a number less than 0.0001, the program will change it to 0.0001.
	2	*linr*	Parameter determining the type of analysis. =0. Nonlinear analysis. =1. Linear analysis.
	3	*amfgas*	Mach number above which the density formulas are modified to avoid vacuum conditions. The default value is $\sqrt{5}$. Any value less than $\sqrt{5}$ is reset to $\sqrt{5}$.
	4	*fsvmi*	Scale factor for the freestream velocity magnitude. = 0.0 Freestream velocity magnitude is unity. This has been used to simulate near-zero onset flow.
	5	*upwnd*	Parameter controlling upwinding to capture shocks. =0. Density biasing. =1. Flux biasing. (recommended)
Card	Field	Name	Description
ang1	all	*icard*	**$ANG** Keyword.
ang2	1	*alpha*	Angle of attack in degrees of freestream (see figure 4.3).
sid1	all	*icard*	**$SID** Keyword.
sid2	1	*beta*	Sideslip angle in degrees of freestream (see figure 4.3).

Default Values

All keywords in this category may be omitted from the input stream. The default values for the data described in this data block are given below.

Data Block	Parameter	Default Value
$SYM	*xzsum*	0.
	xysum	0.
$MAC	*amach*	0.0001
	linr	0.
	amfgas	2.23606798 ($\sqrt{5}$)
	fsvmi	1.0
	upwnd	1.0
$ANG	*alpc*	0.0
	alpha	0.0
$SID	*beta*	0.0

4–4 MATERIAL PROPERTIES DEFINED

Most analysis have two distinct regions with well-defined total pressure and temperature (i.e., freestream and stagnation). For analyses with regions of total pressure and/or temperature (referred to as "material properties") different than freestream, the user must specify what the total pressure and temperature are. In addition, these regions must be enclosed in one or more separate regions bounded by wakes, exhaust or impermeable surfaces. Typical of such regions is a fan cowl exhaust plume.

Specification of the material properties is accomplished in two parts. In the first part (**$MAT**), all the different material properties existing in the flow are defined. In the second part (**$THE**), networks are keyed to the defined properties.

4–4.1 Material Property Table ($MAT)

The keyword **$MAT** is used to specify a table of material properties (see figure 4.5). Freestream air and the region inside the object (called stagnation region) are two items in the table that are generally present in every problem and always exist in the material property table by default. The input to TranAir through this keyword only refers to any additional properties that may be specified.

Data in this keyword block is also used to set the exit Mach number for a base network in supersonic flow.

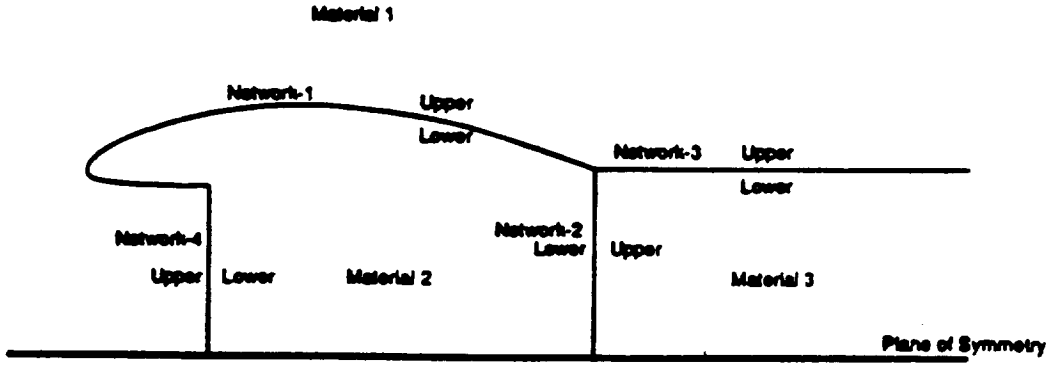


Figure 4.5: Material Properties

Card Images

$MATerial Properties	**mat1**
nmat - - - - - - -	**mat2**
! card **mat3** is to be repeated *nmat* times.	
tpres ttemp adpmat exmach propname - - -	**mat3**

Input Data Description

Card	Field	Name	Description
mat1	all	*icard*	**$MAT** Keyword.
mat2	1	*nmat*	Number of sets of material properties (other than freestream or stagnation).
mat3	1	*tpres*	Ratio of total pressure in the region to the freestream total pressure.
	2	*ttemp*	Ratio of total temperature in the region to the freestream total temperature.
	3	*adpmat*	Adaptive grid scale factor for volume associated with this material type. (Same as *adpfac* in **$LBO** and **$TOL**.) Relevant only if solution adaptive gridding is used or if scaled error estimates are requested in the TranAir graphics output. =1.0 or 0.0 No special gridding emphasis in this volume. Recommended value. = 2.0 For example, will cause additional grid refinement. = 0.5 For example, will cause less grid refinement.
	4	*exmach*	Mach number for base network in this region to establish exit condition for supersonic flow out of base. This may be used to select from subsonic/supersonic solution branches for one dimensional throated nozzles as well as to set the exit conditions for straight-duct exit nozzles.
	5	*mtname*	Material property name (`FORMAT(A)` data).

A parameter requiring some additional explanation is:

- *adpmat* - Also associated with the region identified in an adaptive scale factor used to emphasize or de-emphasize the error indicators for the volume occupied by this material property. This is used only for solution adaptive gridding.

Default Values

All keywords in this category may be omitted from the input stream. The default values for the data described in this data block are given below.

Parameter	Default Value
nmat	0.

4–4.2 Material Property Surface Assignment ($THE)

The keyword **$THE** is used to associate the material properties with the table defined by the keyword **$MAT**. Typically, one material property is associated with each side of a network. Note, if either side of any network has a material defined in the keyword $MAT, then material properties for both sides must be assigned even though one of the sides has freestream or stagnation properties associated with it. On the other hand, if the material on both sides of a network happen to be either freestream or stagnation, then it is not necessary to assign material properties to it through this keyword. Those are assigned automatically.

Card Images

$THErmal Property Association	**the1**
nthe - - - - - - -	**the2**
! card the3 is to be repeated *nthe* times.	
nno *matu* *matl* - - - - -	**the3**

Input Data Description

Card	Field	Name	Description
the1	all	*icard*	**$THE** Keyword.
the2	1	*nthe*	Number of networks defining boundaries of nonstandard material properties.
the3	1	*nno*	Network number or name.
	2	*matu*	Upper surface material property index or name. = 1.0 Freestream. = 2.0 Stagnation. = 3.0 First material property defined in **$MAT**. etc. = *nmat*+2 Last material property defined in **$MAT**.
	3	*matl*	Lower surface material property index or name. See *matu* for possible input values.

Default Values

Selection of boundary conditions normally assigns material properties automatically with regard to the two default properties, freestream and stagnation. If additional properties are desired in your model, you must define the material properties on *both* sides of each network whose surface contacts a region with one of the additional material properties. For example, on a base network exiting into a powered plume, the

lower surface is defined automatically to have stagnation properties and the upper surface is defined to have freestream properties. If you wish to model a powered plume, you must provide the total pressure and temperature data in the **$MAT** keyword data block *and* you must include the base network in the list of networks in the **$THE** keyword block. In the base network entry you must describe its lower surface as having stagnation properties, and its upper surface as having the material property index corresponding to the appropriate entry from the data introduced by the **$MAT** keyword.

If your configuration only contains stagnation and freestream properties, you do not have to provide this input.

4–5 SOLUTION CONTROL AND FIELD GRID GENERATION

The inputs which control the solution and generation of the field grid are described in this section. A brief outline of the section is summarized below:

- 4-5.1 Controlling Iterative Solution (**$ITE**)
- 4-5.2 Global Grid (**$BOX**)
- 4-5.3 Local Refinement of Global Grid (**$TOL, $LBO, $NRE, $SBO**)
- 4-5.4 Grid Refinement Procedure - Grid Sequencing
- 4-5.5 Grid Refinement Procedure - Solution Adaptive Grid (**$ADA**)
- 4-5.6 Boundary Layer Coupling (**$BOU**)

The first four sections are important for assembling a "Grid Sequencing" TranAir analysis. The first five sections are important for assembling a "Solution Adaptive" TranAir analysis. The latter approach costs more computer resources to run but is favored over the former because the adaptive grid locally refines the grid as needed to resovle gradients in the solution. The best guideline for developing an adaptive grid strategy is to locate a previously run case which is similar in nature to the one that you would like to run. All of the sections are important for including a coupled boundary layer analysis with the TranAir solution.

4–5.1 Controlling Iterative Solution ($ITE)

TranAir solves a set of nonlinear equations which simulate the full-potential equation using the Newton method. In applying the Newton method, the problem is linearized and the linear problem is solved during each iterative step. The linear problem is also solved iteratively using the GMRES method, (see Theory Document [2]). This keyword block specifies a number of parameters which control this part of the solution process. They include:

- *niter* - The maximum number of GMRES iterations for all Newton steps permitted to achieve a solution to this problem. The number of Newton steps which will be permitted is approximately this number divided by *nsrch* (see below).

- *tol* - The number of digits (orders of magnitude) reduction in the residuals required before the solution is assumed to be solved.

- *nvis* - The number of Newton cycles using viscosity damping. Viscosity damping should only be used if density biasing is invoked (see the *upwnd* parameter under the **$MAC** keyword datablock). With a non-zero value for *nvis*, the viscosity is increased by a factor of *nvis* for the first grid and first Newton step. After partially converging the problem, the viscosity is reduced and the process continues until the additional viscosity goes to zero. At this point, the solution has been converged on the first grid. For subsonic freestreams, all subsequent grids do not increase the viscosity. For supersonic freestreams, all subsequent grids reset the viscosity and reduce it as convergence is achieved.

- *dropt* - During each linear step, a sparse solver preconditioner is applied. Hence, it is necessary to construct and decompose the sparse matrix. The performance of GMRES in solving the linear problem depends on the quality of the preconditioner and the accuracy of the decomposition. The accuracy of the decomposition of the sparse matrix is determined by a drop tolerance specified through the parameter *dropt*. Generally, the smaller the value of *dropt*, the better the convergence is at the expense of greater CPU and storage cost for the decomposition.

- *nsrch* - The total number of search directions permitted to GMRES for linearization. The code tries to solve the linear problem within a specified tolerance (set internally) and may take less than *nsrch* iterations. *niter/nsrch* is approximately equal to the maximum number of Newton steps permitted to attain a solution on any single grid.

- *njac* - The sparse matrix and its decomposition are periodically recomputed. The parameter *njac* specifies the maximum number of such computations on a given grid. Note, however, that the code will calculate a new matrix only if it is absolutely necessary (when the linear problem has difficulty converging).

Note that if *njac* and *nsrch* are both too small then the solution process may have difficulty converging.

Card Images

$ITErations						**ite1**
niter	*tol*	*nvis*	*dropt*	*nsrch*	*njac*	**ite2**

Input Data Description

Card	Field	Name	Description
ite1	all	*icard*	**$ITE**ration Keyword.
ite2	1	*niter*	Maximum number of linear GMRES iterations to be carried out per grid. Recommended value is 600.
	2	*tol*	Magnitude for base 10 exponent of convergence tolerance. Convergence tolerance is 10^{-tol}. =7. Implies convergence tolerance of 10^{-7}. Recommended value.
	3	*nvis*	Number of steps of viscosity damping. =0. If flux biasing is used in **$MAC**. =1. For subsonic freestream. Recommended value. =2. For supersonic freestream. Recommended value.
	4	*dropt*	Drop tolerance in the matrix decomposition. =.0001 to .001 Recommended values.
	5	*nsrch*	Maximum number of linear GMRES iterations to be carried out *per Newton method step*, per grid. =60. Recommended value. **Note:** The maximum number of Newton steps performed on a grid is approximately equal to *niter*/*nsrch*. (This number should not be less than 10.)
	6	*njac*	Maximum number of sparse matrix preconditioners to be computed for a grid. =2. Recommended value. **Note:** If zero is input, the code attempts to compute (at least) as many preconditioner matrices as it needs to reach convergence. Setting *njac* = 0 is *not* recommended.

Default Values

All of the keywords in this category may be omitted from the input stream. The default values for the data described in this data block are given below.

Parameter	Default Value
niter	600.
tol	7.
nvis	1.
dropt	.001
nsrch	60.
njac	2.

4–5.2 Global Grid ($BOX)

Defining a Finite Computational Domain

This step consists of identifying the portion of space over which the computations are performed to solve the boundary value problem. For the sake of efficiency, it is desirable to make this region as small as possible. The computational domain need only enclose all configuration and wake surfaces, a buffer zone away from all configuration surfaces (more details are given below), and significant flow regions. For subsonic freestream cases, these flow regions include anticipated supersonic zones. In supersonic freestream cases, these regions include anticipated subsonic zones; in addition, the computational domain must be extended sufficiently that no Mach waves reflect from the top or sides of the box and strike an aftward portion of the configuration.

The computational domain is a box described by bounding coordinate planes: $x = xi$, $x = xf$, $y = yi$, $y = yf$, $z = zi$ and $z = zf$. The parameters $xi < xf$, $yi < yf$, and $zi < zf$ are specified as input or are determined internally by the program from the configuration geometry (see the **$BOX** keyword description below).

Refinement of Computational Domain to Form Global Grid

The computational domain is uniformly refined to create a global grid. A global grid is characterized by the number of grid points in the three coordinate directions; i.e., nx, ny and nz which can be specified. Such a grid contains $(nx-1)*(ny-1)*(nz-1)$ rectangular cells each with side lengths dx, dy, and dz, where:

$$\begin{aligned} dx &= (xf - xi)/(nx - 1), \\ dy &= (yf - yi)/(ny - 1), \\ dz &= (zf - zi)/(nz - 1). \end{aligned} \tag{4.1}$$

There are two restrictions on the input numbers nx, ny, nz. First, they may take only certain integer values. Specifically, nx must be of the form:

$$nx = 1 + 2^{n_2}3^{n_3}5^{n_5}, \tag{4.2}$$

where n_2, n_3, n_5 are nonnegative integers; i.e., $(nx - 1)$ may have only 2, 3, or 5 as prime factors. The same restriction applies to ny and nz (with possibly different exponents n_2, n_3 and n_5).

The second restriction regards the buffer zone about the configuration that was mentioned earlier. The computational domain and the numbers nx, ny and nz must be such that the buffer zone includes at least two planes of global grid cells at the downstream face $(+x)$ of the computational domain and one plane of global grid cells at the other five bounding faces. For example, nx must be such that:

$$xi < x_{min} - dx \tag{4.3}$$

and

$$xf > x_{max} + 2dx \tag{4.4}$$

where x_{min} is the minimum of the x coordinate values of all network corner points and x_{max} is the maximum x coordinate value of the penultimate corner point of all the columns in all the standard wake networks. Restrictions similar to equation (4.3) are applicable at both of the extreme faces of the computational domain in each of the other two coordinate directions. However, if the problem has a symmetry plane(s) (e.g., xz symmetry), then the buffer zone restriction(s) is not applicable (e.g., at $y = 0$).

The global grid resolution affects the accuracy of computed solutions in two ways. Generally, the global grid is a subset of the finest automatically constructed grid used to solve a problem. Hence, the grid sizes dx, dy and dz generally determine the minimum grid resolution used *throughout* the computational domain. The second effect is a consequence of the way in which the global grid is used in representing the far field condition on potential. The accuracy with which far field effects are resolved improves with decreasing values of dx, dy and dz.

The global grid is the computational domain over which a solution is developed in TranAir. Care should be taken to define the global grid to include all significant regions of nonlinear flow (e.g., all supersonic regions for analysis with the subsonic freestream). Finally, TranAir generally performs best when the local grid aspect ratios (dy/dx, dz/dy, dx/dz, where $dx = (xf - xi)/(nx - 1)$ etc.) do not differ significantly from one (maximum should be about three). Since each locally refined grid cell has the same aspect ratio as a cell in the global grid, care must be taken to ensure that the global grid cell aspect ratios are not too large.

A top and side view of a simple global grid are illustrated in figure 4.6. The parameters used to define the global grid box are shown.

Card Images

$BOX								**box1**
xi	xf	nx	-	-	-	-	-	**box2**
yi	yf	ny	-	-	-	-	-	**box3**
zi	zf	nz	-	-	-	-	-	**box4**

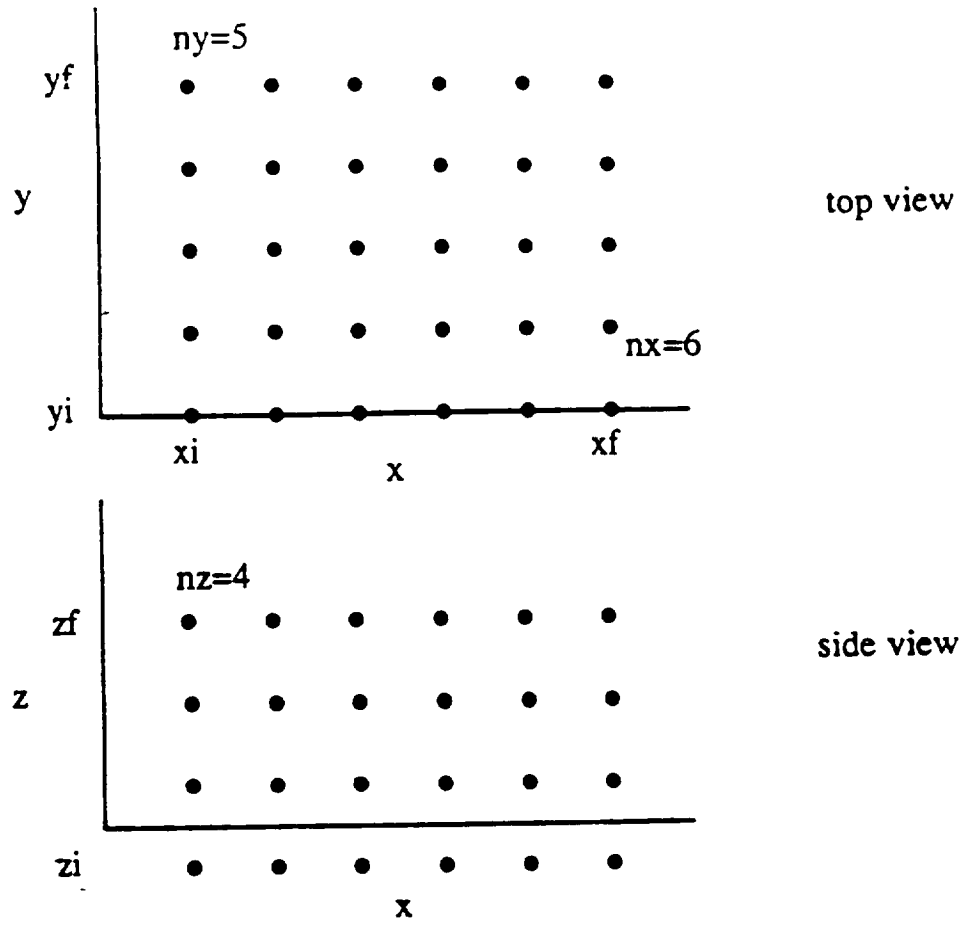


Figure 4.6: Parameters Describing Global Grid

Input Data Description

Card	Field	Name	Description
box1	all	*icard*	Keyword **$BOX**.
box2	1	xi	Extent of the computational box in the $-x$ direction.
	2	xf	Extent of the computational box in the $+x$ direction.
	3	nx	Number of global grid points in the x direction (see *note below).
			Note: If both xi and xf are input as zero, then the program will compute these values from the physical extent of all configuration networks (similarly for the y and z directions). However, this will not ensure that the computational domain includes all significant regions of nonlinear flow (e.g., all supersonic regions in subsonic freestream cases). For this reason, the computational box should be explicitly defined.
box3	1	yi	Extent of the computational box in the $-y$ direction.
	2	yf	Extent of the computational box in the $+y$ direction.
	3	ny	Number of global grid points in the y direction (see *note below).
box4	1	zi	Extent of the computational box in the $-z$ direction.
	2	zf	Extent of the computational box in the $+z$ direction.
	3	nz	Number of global grid points in the z direction (see *note below).

*Acceptable Values

3	10	21	37	61	91	129	181	244	321	406	513	649	801
4	11	25	41	65	97	136	193	251	325	433	541	676	811
5	13	26	46	73	101	145	201	257	361	451	577	721	865
6	16	28	49	76	109	151	217	271	376	481	601	730	901
7	17	31	51	81	121	161	226	289	385	487	626	751	961
9	19	33	55	82	126	163	241	301	401	501	641	769	973

4–5.3 Local Grid Refinement ($TOL, $LBO, $NRE, and $SBO)

$TOL

In the process of creating the finest grid, the decision of whether or not to refine a grid cell depends on user-specified control parameters. These parameters include a maximum box size $dxmax$, a minimum box size $dxmin$ and a scale factor for comparison of box sizes to local panel lengths. Specification of these parameters is done through the **$TOL** and **$LBO** keyword blocks. The keywords specify values of these parameters which hold over an "operative volume" of space. The "operative volume" may be either the volume occupied by boxes "near the boundary," or may include volumes up to and including the entire computational grid. Thus, the parameters may vary through different portions of the grid. The refinement of a grid cell is controlled by these parameters according to three criteria:

1. The first criterion, which applies over any cell in the operative volume, is that the cell length in the x direction may not exceed a given value $dxmax$. If a box has a cell length in the x direction which exceeds the value of $dxmax$ which is operative in the volume occupied by the box, the box will be refined.

2. The second criterion affects boxes in the operative volume which are also in the "neighborhood of a boundary." The criterion is that one of the cell's three length scales must be less than the "scaled diameter of some nearby panel."

3. The third criterion also applies to cells in the operative volume which are also near a boundary (boundary-based). The length of such a cell in the x direction cannot be less a given value $dxmin$. Thus, no cell will be refined to such a level that its length in the x direction is less than the value of $dxmin$ operative in the volume occupied by the cell.

For the second criterion, "the neighborhood of the boundary" means that a magnified version of the cell intersects some panel of some network, where the magnified version of a cell is a cell with the same center and which is scaled by the specified value of the parameter $epsf$.

"Scaled diameter of some nearby panel" refers to $|panfac| * psize$, where $panfac$ is specified as input and $psize$ is the minimum of the diameters of all panels cut by the magnified cell. The diameter of the panel is the minimum of the distances between two opposite edge midpoints. Figure 4.7 illustrates the process of boundary-based grid refinement for a cell in a case where $panfac = 1.0$ and $epsf = 1.5$. Typical values for $|panfac|$ are in the range (0.5,5.0), which result in local grid sizes that are about .5 to 5 times as large as the smallest nearby panel.

The operative volume is the volume of boxes in the neighborhood of the boundary if $panfac$ is positive. It includes the entire volume referred to by the particular keyword if $panfac$ is negative. The keyword **$TOL** refers to the entire volume of the computational grid. The keyword **$LBO** contains within it a definition of the volume affected by the parameters defined in the keyword. One additional grid refinement

keyword is available which only affects refinement in the neighborhood of a boundary. This keyword is **$NRE**. Its parameters affect refinement of boxes which are in the neighborhood of a specified network or networks. Finally, there is an obsolete keyword **$SBO** which may be used in lieu of **$LBO**; it is documented, but its use is discouraged.[2]

Any particular box may be affected by more than one set of grid refinement parameters (for example, **$LBO**'s may intersect.) A precendence hierarchy is used to determine which controls are used for a particular box. For any box in the neighborhood of a boundary affected by an **$NRE**, the parameters for the **$NRE** take precedence. For any box affected by one or more **$LBO**'s, precedence is given to the first-occurring **$LBO** in input order. All **$LBO**'s take precedence over specifications defined by **$TOL**.

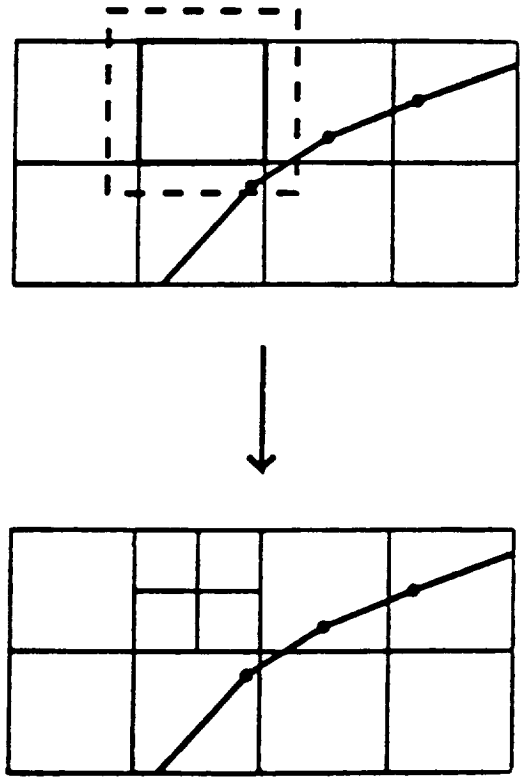


Figure 4.7: Application of Boundary-based Grid Refinement Criterion

The keyword **$TOL** can be used to control grid resolution over the entire computational domain (volume-based) or near all configuration and wake networks (boundary-based). The values of the minimum and maximum grid sizes *dxmin* and *dxmax*, mentioned earlier, are set equal to the values of the **$TOL** parameters *dxmin* and *dxmax*. The value of the **$TOL** parameter *epsf* (see earlier description) is generally

[2]Internally, **$SBO's** are converted to an equivalent **$LBO**, but solution adaptive controls cannot be applied to the converted **$SBO's**. See the discussion of Solution Adaptive Grid Options.

used for all specified grid refinement controls. Specifying larger *epsf* values has the effect of extending local grid refinement near a boundary further into the field.

$LBO

The **$LBO** keyword may be used to control grid resolution within a hexahedral shaped special region of interest. Many such regions may be provided as input, each with its own **$LBO** keyword and parameters. The geometry of such a special region is specified with coordinates of the region's eight corner points (see figure 4.8). The corners should be distinct and define a convex region. The limiting grid sizes *dxmin* and *dxmax*, mentioned earlier, vary trilinearly in each special region. At any point in the special region, these sizes are set equal to the values of functions defined by eight corner point values. For example, at the special region's ith corner point,

$$dxmin = dxmin \cdot reldxi, \tag{4.5}$$

$$dxmax = dxmax \cdot reldxi, \tag{4.6}$$

where *dxmin*, *dxmax* and *reldx*i, for i= 1,..,8, are specified **$LBO** parameters.

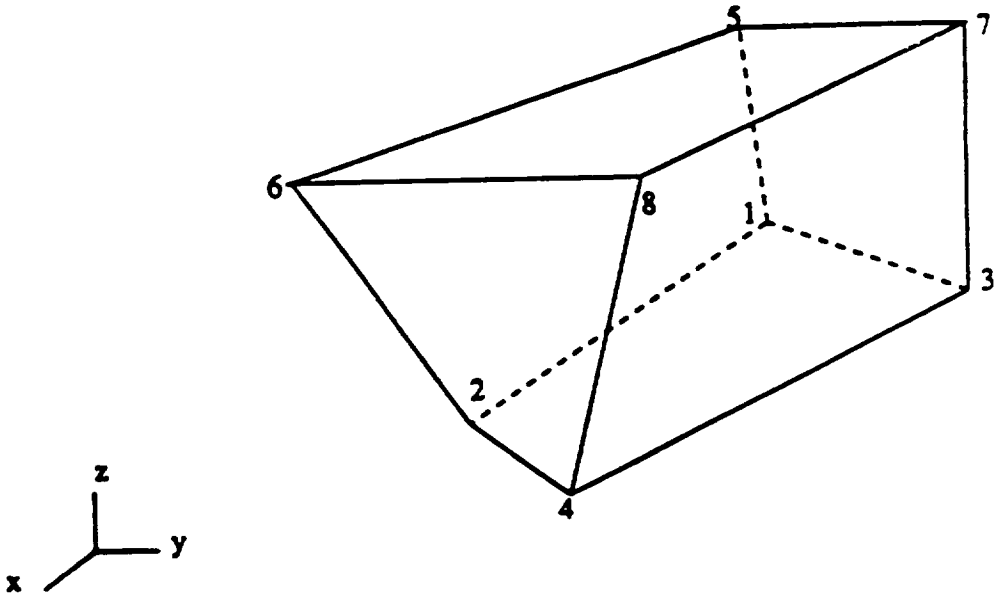


Figure 4.8: Corner Point Numbering Convention for a Special Region of Interest specified by an **$LBO**

$SBO

The **$SBO** keyword is a less general version of the **$LBO** keyword with a slightly different input format. The grid refinement limits are constant within the region of interest (rather than trilinearly varying) and **$SBO**'s cannot be changed in adaptive cycles. Specification of *dxmin* and *dxmax* is through specification of *minlvl* and *maxlvl*, levels of grid refinement relative to the global grid. Internal to the code, the **$SBO** input is converted to the **$LBO** format with:

$$dxmin = dx/(2^{maxlvl+\frac{1}{2}}) \tag{4.7}$$

$$dxmax = dx/(2^{minlvl-\frac{1}{2}}) \tag{4.8}$$

where dx is the x-dimension of a global grid box. Refinement is then performed in the same way as described below for an **$LBO**. Since the **$SBO** keyword is far less general than the **$LBO** keyword, its use is discouraged.

$NRE

The **$NRE** keyword may be used to control local grid refinement near a specific network. Parameters provided with the **$NRE** keyword are analogous to the **$TOL** parameters. In particular, a value for *epsf* is specified that overrides the analogous **$TOL** parameter.

Example

The keywords **$TOL**, **$LBO**, and **$NRE** are quite flexible and can used to specify an appropriate grid for a **grid sequenced** TranAir solution. By way of example, a case involving a typical wing configuration is now briefly sketched. The **$TOL** keyword can be used with $panfac = 2.0$, $epsf = 1.5$, and non-limiting values of *dxmin* and *dxmax* (i.e., small *dxmin* and large *dxmax*) to specify a moderate amount of local grid refinement near the wing. Most of the refined cells typically would be located in the vicinity of the thinnest panels at the wing leading edge. A box-shaped **$LBO** enclosing the wing tip might also be specified. Specifying $panfac < 0$, a large maximum grid size *dxmax*, and unity values for $reldx1, .., reldx8$ in the **$LBO** would override **$TOL** and keep the grid coarse about the wing tip for a run in which flow there is not of interest. Finally, one or more **$LBO** keywords can be used with nonpositive *panfac* values to specify local grid refinement in regions of interest above and away from the wing. If the ratio of outboard to inboard wing chord were small and finer grid were desired outboard then inboard, a linearly decreasing (from root to tip) maximum grid size could be used in each such **$LBO** to achieve the desired grid effects.

Card Images

$TOLerance for Global Grid Refinement								**tol1**
! (outside of **$LBO**'s, **$NRE**'s, and **$SBO**'s)								**tol1**
adpfac	*panfac*	*dxmin*	*dxmax*	*epsf*	-	-	-	**tol2**
$LBO Grid Refinement for Special Region								**lbo1**
adpfac	*panfac*	*dxmin*	*dxmax*	*lbnam*	-	-	-	**lbo2**
$x1$	$y1$	$z1$	*reldx*1	-	-	-	-	**lbo3**
$x2$	$y2$	$z2$	*reldx*2	-	-	-	-	**lbo4**
$x3$	$y3$	$z3$	*reldx*3	-	-	-	-	**lbo5**
$x4$	$y4$	$z4$	*reldx*4	-	-	-	-	**lbo6**
$x5$	$y5$	$z5$	*reldx*5	-	-	-	-	**lbo7**
$x6$	$y6$	$z6$	*reldx*6	-	-	-	-	**lbo8**
$x7$	$y7$	$z7$	*reldx*7	-	-	-	-	**lbo9**
$x8$	$y8$	$z8$	*reldx*8	-	-	-	-	**lb10**
$NRE Network (Network Based Grid Refinement)								**nre1**
numnre	-	-	-	-	-	-	-	**nre2**
! Card **nre3** is provided *numnre* times.								
netnam	*panfac*	*dxmin*	*dxmax*	*epsf*	-	-	-	**nre3**
$SBO Grid Refinement for Special Region (archaic form)								**sbo1**
adpfac	*panfac*	*minlvl*	*maxlvl*	-	-	-	-	**sbo2**
$x1$	$y1$	$z1$	$x2$	$y2$	$z2$	-	-	**sbo3**
$x3$	$y3$	$z3$	$x4$	$y4$	$z4$	-	-	**sbo4**
$x5$	$y5$	$z5$	$x6$	$y6$	$z6$	-	-	**sbo5**
$x7$	$y7$	$z7$	$x8$	$y8$	$z8$	-	-	**sbo6**

Input Data Description

Card	Field	Name	Description
tol1	all	*icard*	Keyword **$TOL**.
tol2	1	*adpfac*	Scale factor applied to error estimates used in solution adaptive gridding process. Parameter ignored for grid sequencing. Values *greater* than 1.0 will scale up the error and result in *more* field grid. Values *less* than 1.0 will scale down the error and result in *less* field grid. = 1.0 Recommended value.
	2	*panfac*	Sign controls the region of grid refinement. If *positive*, applies only to grid boxes in the neighborhood of the panel boundary. If *negative*, applies to all grid boxes of the region (volume-based). Magnitude of factor is applied as a scale factor to the minimum panel diameter. The local grid dimension must be less than the above value. (i.e., $\|panfac\| = L_{box}/L_{panel}$)
	3	*dxmin*	Minimum grid box length in the x direction.
	4	*dxmax*	Maximum grid box length in the x direction. **Note:** *dxmin* and *dxmax* must satisfy $0.0 < dxmin \leq dxmax$.
	5	*epsf*	Factor by which a grid box is magnified to determine if a nearby surface panel intersects it. The parameter *epsf* must be non-negative. = 1.51 Recommended value. = 0. Implies *epsf*=1.51.

Card	Field	Name	Description
lbo1	all	*icard*	Keyword **$LBO**.
lbo2	1	*adpfac*	Scale factor applied to error estimates used in solution adaptive gridding process. Parameter ignored for grid sequencing. Values *greater* than 1.0 will scale up the error and result in *more* field grid. Values *less* than 1.0 will scale down the error and result in *less* field grid. If, for example, there is twice the interest in resolving flow effects in this special region as in all other parts of the flow field, and the solution adaptive grid option is selected, $adpfac = 2.0$ and the **$TOL** $adpfac = 1.0$ should be set.
	2	*panfac*	Sign controls the region of grid refinement. If positive, applies only to grid boxes in the neighborhood of the panel boundary. If negative, applies to all grid boxes of the region (volume-based). Magnitude of factor is applied as a scale factor to the minimum panel diameter. The local grid dimension must be less than the above value.
	3	*dxmin*	Minimum grid box length in the x direction. Value on the right-hand side of equation (4.5).
	4	*dxmax*	Maximum grid box length in the x direction. Value on the right-hand side of equation (4.6). **Note:** *dxmin* and *dxmax* must satisfy $0.0 < dxmin \leq dxmax$
	5	*lbnam*	**$LBO** name. Ten characters or less. Format (**A**). Used in re-defining the above parameters in the course of a solution adaptive grid run.

Card	Field	Name	Description
lbo3	1,2,3	$x1, y1, z1$	x, y, z coordinates of corner point 1.
	4	$reldx1$	$dxmin$ and $dxmax$ scale factor at corner point 1.
lbo4	1,2,3	$x2, y2, z2$	x, y, z coordinates of corner point 2
	4	$reldx2$	$dxmin$ and $dxmax$ scale factor at corner point 2.
lbo5	1,2,3	$x3, y3, z3$	x, y, z coordinates of corner point 3.
	4	$reldx3$	$dxmin$ and $dxmax$ scale factor at corner point 3.
lbo6	1,2,3	$x4, y4, z4$	x, y, z coordinates of corner point 4.
	4	$reldx4$	$dxmin$ and $dxmax$ scale factor at corner point 4.
lbo7	1,2,3	$x5, y5, z5$	x, y, z coordinates of corner point 5.
	4	$reldx5$	$dxmin$ and $dxmax$ scale factor at corner point 5.
lbo8	1,2,3	$x6, y6, z6$	x, y, z coordinates of corner point 6.
	4	$reldx6$	$dxmin$ and $dxmax$ scale factor at corner point 6.
lbo9	1,2,3	$x7, y7, z7$	x, y, z coordinates of corner point 7.
	4	$reldx7$	$dxmin$ and $dxmax$ scale factor at corner point 7.
lbo10	1,2,3	$x8, y8, z8$	x, y, z coordinates of corner point 8.
	4	$reldx8$	$dxmin$ and $dxmax$ scale factor at corner point 8. **Note**: The convention for numbering the eight corner points is shown in figure 4.8. **Note**: The hexahedral shaped special region defined by the **$LBO** coordinates must not have collapsed edges or faces. **Note**: If, for example, $panfac < 0.0$ and the relative grid size factors $reldx1 = ... = reldx8 = 1.0$, then $dxmin$ and $dxmax$ are constant, equal to $dxmin$ and $dxmax$ specified. **Note**: The parameters $reldx1, ...reldx8$ in each **$LBO** should be greater than or equal to one. If input as a negative number, or input as zero, each such value is reset to 1.0 internally by the program.

Card	Field	Name	Description
nre1	all	*icard*	Keyword **$NRE**.
nre2	1	*numnre*	Number of networks for which **$NRE** grid refinement controls are specified.
nre3	1	*netnam*	Number or character name of the network near which grid refinement control is specified. Format (**A10**).
	2	*panfac*	Scale factor applied to the minimum panel diameter. The local grid dimension must be less than the above value.
	3	*dxmin*	Minumum grid box length in the x direction.
	4	*dxmax*	Maximum grid box length in the x direction. **Note**: *dxmin* and *dxmax* must satisfy $0.0 < dxmin \leq dxmax$.
	5	*epsf*	Factor by which a grid box is magnified to determine if a nearby surface panel intersects it. The parameter *epsf* must be non-negative. = 1.51 Recommended value. = 0. Implies *epsf*=1.51.

Card	Field	Name	Description
sbo1	all	*icard*	Keyword **$SBO**.
sbo2	1	*adpfac*	Scale factor applied to error estimates used in solution adaptive gridding process. Parameter ignored for grid sequencing.
	2	*panfac*	Sign controls the region of grid refinement. If positive, applies only to grid boxes in the neighborhood of the panel boundary. If negative, applies to all grid boxes of the region (volume-based). Magnitude of factor is applied as a scale factor to the minimum panel diameter. The local grid dimension must be less than the above value.
	3	*minlvl*	Minimum level of refinement relative to the global grid. See equation (4.8).
	4	*maxlvl*	Maximum level of refinement relative to the global grid. See equation (4.7).
sbo3	1,2,3	$x1, y1, z1$	x, y, z coordinates of corner point 1.
	4,5,6	$x2, y2, z2$	x, y, z coordinates of corner point 2.
sbo4	1,2,3	$x3, y3, z3$	x, y, z coordinates of corner point 3.
	4,5,6	$x4, y4, z4$	x, y, z coordinates of corner point 4.
sbo5	1,2,3	$x5, y5, z5$	x, y, z coordinates of corner point 5.
	4,5,6	$x6, y6, z6$	x, y, z coordinates of corner point 6.
sbo6	1,2,3	$x7, y7, z7$	x, y, z coordinates of corner point 7.
	4,5,6	$x8, y8, z8$	x, y, z coordinates of corner point 8.

4–5.4 Grid Refinement Procedure - Grid Sequencing

The computational grid used in TranAir consists of locally refined rectangular cells (also referred to as box elements, elements, or regions). All rectangular cells are geometrically similar, with any two cells differing from one another by at most a single scale factor and a translation. TranAir automatically constructs all grids using control parameters that are provided as input. Two grid options are available in the code: **grid sequencing** (name explained below), which is the default option, and the **solution adaptive grid** option. With either option, the following steps are taken to construct a suitable computational grid:

- The finite computational domain is defined.
- The computational domain is uniformly refined to form a *global grid.*
- The global grid is locally refined.
- If the solution adaptive grid option has been specified, further local grid refinements and derefinements are carried out in conjunction with the problem solution.

For grid sequencing, the final solution is computed on the finest grid as defined by **$TOL**, **$LBO**, **$NRE**, and/or **$SBO**. The coarser grids in the nested sequence are used to obtain the final solution in an efficient and reliable manner. Typically, two or three such intermediate grids are used. Thus, no additional input is required to implement the grid sequencing grid refinement procedure.

4–5.5 Grid Refinement Procedure - Solution Adaptive Grid ($ADA)

This option is invoked when the **$ADA**pt keyword is used. Starting with the initial grid described in the previous subsection, a sequence of one or more additional grids is constructed according to user-specified constraints and local estimates of the solution error on each grid. The maximum number of adaptive grids and the target number of cells in the final adaptive grid in the sequence are specified by the user. Generally, the solution defined on the final grid is returned to the user. It is also possible, however, to cycle many times through the process of creating a sequence of adaptive grids. This is specified by setting the **$ADA**pt keyword parameter *ncycle* > 1 and supplying *ncycle* sets of control parameters. If two or more cycles are used, the initial grid in each later cycle is taken to be the final adaptive grid in the previous cycle and the final solution returned to the user is that on the final adaptive grid in the final cycle. If *ncycle* > 1 is specified, the control parameters are changed during the solution process. Before this is discussed further, the control parameters used in each cycle are described.

An adaptive grid cycle terminates successfully when a solution has been computed on a grid which either has approximately *nbxtgt* (or more) cells or which is the *maxgrd*-th grid created in the cycle. Here, *nbxtgt* is a specified target number of cells and *maxgrd* is a specified maximum number of adaptive grids in the cycle. In deciding how many cells a new adaptive grid will have, the code examines the ratio $R = ncells/nbxtgt$, where *ncells* is the number of cells in the current grid. The code compares this ratio to the values of specified parameters *ratio*1 and *ratio*2. If $R >$ *ratio*2 (a situation which typically occurs only when R is close to 1.0), the number of cells to be refined and derefined is determined internally by the code. If *ratio*1 $<$ $R <$ *ratio*2, no cells are derefined and as many as possible are refined in attempting to create a grid with about *nbxtgt* ncells. If $<$ *ratio*1, then approximately $fracrf *$ *ncells* cells are refined and up to $fracdf *$ *ells* cells are derefined. Here, $fracrf$ and $fracdf$ are nonnegative input parameters whose sum must not exceed 1.0.

In creating an adaptive grid, the cells refined (derefined) are those with the largest (smallest) weighted error estimates. The weights (adaptive scale factors) are input by a user with the *adpfac* parameter in **$TOL** and with *adpfac* in each **$LBO**. The **$TOL** *adpfac* value is used to scale a cell's error estimate when the **$TOL** parameter $panfac \leq 0.0$ or when $panfac > 0$ and the cell intersects a network. This scaling is overridden if the cell lies in a special region of interest defined by an **$LBO**. If one or more such special regions enclose the cell, the *adpfac* value of the first **$LBO** appearing in the input file is used. The *adpfac* values (which are generally between 0.1 and 10.0) specified in **$TOL** and **$LBO** provide a convenient way to emphasize/de-emphasize local grid refinement in regions of space of the most/least interest.

Another way that the **$TOL** and **$LBO** keywords affect solution adaptive grid refinement and derefinement is through their maximum and minimum local grid size parameters (e.g., *dxmax* and *dxmin*). Grid refinement/derefinement restrictions obtained from these limiting grid sizes are applied for every adaptive grid. If some cells

violate maximum local grid size restrictions, these cells are given priority when the code decides which cells to refine in creating the next adaptive grid. The precedence used in applying these restrictions is analogous to the precedence used with the **grid sequencing** option. This was done in order to minimize the number of changes needed to convert a **grid sequencing** input file for a solution adaptive grid run. In many cases this conversion might consist only of making sure all *adpfac* parameters are appropriately set (e.g., to be 1.0), specifying *dxmin* to be less than (e.g., one tenth to one fifth of) *dxmax* in **$TOL** and all **$LBO**'s, and adding the **$ADA**pt keyword and parameters.

In considering a cell which *does not* intersect any network, the adaptive grid algorithm applies local grid size limits when either (*i*) the **$TOL** parameter $panfac \leq 0$ or (*ii*) the cell lies in an **$LBO** special region of interest for which $panfac \leq 0$. If (*i*) holds but (*ii*) does not, then the local grid size limits are set by the **$TOL** parameters *dxmin* and *dxmax*. If (*ii*) holds, then the maximum and minimum limits are taken from the first **$LBO** keyword appearing in the input file for which (*ii*) holds. In considering a cell which *does* intersect some network, the adaptive grid algorithm always applies local grid size limits set by **$TOL** and/or **$LBO** parameters. The maximum size limit is set by the **$TOL** parameter *dxmax* unless one or more **$LBO** special regions of interest contain the cell, in which case the maximum size limit is set by the first **$LBO** appearing in the input file. The minimum local grid size limit is taken to be the smallest (least restrictive) of two limits, the first defined as above for a cell not intersecting a network and the second defined similarly, but with "$panfac \leq 0$" replaced by "$panfac > 0$."

The **$TOL** and **$LBO** keyword parameters (in particular *dxmin* and *dxmax*) are used to both create the initial grid *and* to restrict grid refinement when adaptive grids are created. In some applications, it is desirable to change some of these parameter values from time to time. The facility for this change is provided through the **$ADA**pt keyword parameter *numlbo*, which is the number of already specified **$LBO**'s whose parameters are to be reset for the current adaptive grid cycle. For each such **$LBO**, new values of *adpfac*, *panfac*, *dxmin*, and *dxmax* are specified.

A very important rule of thumb for using the **solution adaptive grid** option is the following obvious, and yet easily forgotten, adage. The code does not know which spatial regions a user is most or least interested in unless the user tells the code! With conventional codes, or with TranAir and the **grid sequencing** option, it must be communicated to the code (by *not* specifying fine local grid) that certain flow effects are not of interest. These effects could include large flow gradients near wing tips, configuration geometry irregularities, aircraft bodies where only gross effects are of interest, etc. The solution adaptive grid algorithm "sees" all such effects and should be constrained by a user (through **$LBO** parameters) in order to obtain a solution with the highest accuracy in regions of highest interest. When only interested in the gross details of the flow about a part of a configuration, the best way to constrain the algorithm is through an **$LBO** *adpfac* parameter. When there is no interest in the flow about a part of a configuration or a wake, the best way to communicate this to the code is by specifying a large value for an **$LBO** *dxmin* parameter.

It also can be useful to restrict grid refinement in a region where the flow *is* of

significant interest! This is because some flow features (e.g., leading edge effects) sometimes dominate other latent features of interest (e.g., oblique shocks). Latent flow features are those that cannot be detected until a sufficiently fine grid is already present. If it is anticipated in an application that there are latent flow features of interest, specify *ncycle* > 1 and limit grid refinement (through the *dxmin* parameter) during early cycles in **$LBO** regions containing dominant flow effects.

An example of this is the following. The first two cycles are set at *ncycle* = 3 and *nbxtgt* = $N/2$, and *nbxtgt* = N is set for the third cycle, where N is the final desired target number of grid cells. If *maxgrd* for the first grid cycle was set sufficiently large so that the final grid in the first cycle had about $N/2$ cells, then adaptive gridding in the second cycle would consist of attempting to better distribute local errors, without significantly increasing the number of grid cells. This can be particularly effective in detecting/capturing latent flow features, especially if the *dxmin* parameter values have been specified in **$LBO**'s containing dominant flow features to be large. For the third cycle, in which the number of grid cells about doubles (from $N/2$ to N), the aforementioned *dxmin* values would be decreased by a factor of two or more to allow the solution adaptive grid algorithm to refine where it chooses. With the default values of **$ADApt** parameters *ratio*1 and *ratio*2, only grid refinement (no derefinement) would be used in the third cycle. Hence, any grid added to latent flow regions in the second adaptive grid cycle would be retained in the final grid.

Take note of the following. It is possible that the final adaptive grid used in a run will have two characteristics that a final grid obtained via the default **grid sequencing** option would not. First, the global grid specified by a user may not be contained in the final adaptive grid. This is because the initial grid used with the **solution adaptive grid** option is a coarsened version of the finest grid one would get with the **grid sequencing** option specified and the same input file (minus **$ADApt**). Second, the computational domain defined by the **$BOX** keyword parameters is sometimes enlarged in creating an adaptive grid. This occurs in subsonic freestream flow cases whenever the weighted solution error estimates near any of five computational domain boundaries ($-x$, $-y$, $+y$, $-z$ and $+z$) are relatively large.

One final note is added on solution adaptive grid refinement. In creating an adaptive grid, only *single-level* grid refinements and derefinements are performed. This means, for example, that no cell is recursively refined more than once in creating a new grid. Hence, in the course of an adaptive grid cycle, no cell's length in any of the three coordinate directions can be decreased to less than 2^{-j} times its value at the beginning of the cycle, where j is the number of adaptive grids created in the cycle.

Card Images

$ADApt								**ada1**
ncycle	-	-	-	-	-	-	-	**ada2**
! Cards **ada3** to **ada5** must be repeated *ncycle* times.								
maxgrd	*nbxtgt*	*ratio1*	*ratio2*	*fracrf*	*fracdf*	-	-	**ada3**
numlbo	-	-	-	-	-	-	-	**ada4**
! Card **ada5** is provided *numlbo* times.								
adpfac	*panfac*	*dxmin*	*dxmax*	*lbnam*	-	-	-	**ada5**

Input Data Description

Card	Field	Name	Description
ada1	all	*icard*	Keyword **$ADA**pt. If used, this keyword must follow all of the **$LBO** cards that have been input. The absence of this keyword indicates the choice of the **grid sequencing** option.
ada2	1	*ncycle*	Number of solution adaptive grid cycles.
ada3	1	*maxgrd*	Maximum number of solution adaptive grids used during the current cycle.
	2	*nbxtgt*	Target number of grid boxes in a grid in the present cycle. If this number is less than the number in the current grid, it is reset to the number in the current grid.
	3	*ratio1*	Cutoff number between 0. and 1. to which the ratio *ncells/nbxtgt* is compared, where *ncells* refers to the number of cells in the present grid. (See notes at the end of this section.) = 0.4 Recommended value.
	4	*ratio2*	Cutoff number between *ratio1* and 1. to which ratio *ncells/nbxtgt* is compared. = 0.8 Recommended value.
	5	*fracrf*	Approximate fraction (between 0. and 1.) of present grid cells that are refined in creating a new grid. = 0.2 Recommended value.
	6	*fracdf*	Approximate fraction (between 0. and 1. − *fracrf*) of present grid cells that are derefined in creating a new grid. The parameters *fracrf* and *fracdf* are used only if *ncells/nbxtgt* is less than *ratio1*. = 0.4 Recommended value.

Card	Field	Name	Description
ada4	1	*numlbo*	Number of already specified **$LBO**'s whose parameters are to be redefined for all grids in the present cycle.
ada5	1	*adpfac*	Redefined non-negative weighting factor used to emphasize or de-emphasize grid refinement (see **$LBO** description for this parameter).
	2	*panfac*	Redefined *panfac* value (see **$LBO** description for this parameter).
	3	*dxmin*	Lower grid box length in the x direction.
	4	*dxmax*	Upper grid box length in the x direction. **Note:** *dxmin* and *dxmax* must satisfy $0.0 < dxmin \leq dxmax$.
	5	*lbnam*	Character name used to identify this special region of interest. Format (**A**).

Note: Let $R = ncells/nbxtgt$, where *ncells* are the number of boxes in the current grid. Then, if $R < ratio1$ for this cycle, approximately $fracrf * ncells$ boxes will be refined and approximately $fracdf * ncells$ boxes will be derefined. If $ratio1 < R < ratio2$, no cells will be derefined and as many as possible will be refined in an attempt to reach *nbxtgt*. If $R > ratio2$, the number of cells to be refined or derefined is determined internally by the program.

Note: It could happen that the number of cells in the last solution adaptive grid in a cycle is much less than *nbxtgt* if *maxgrd* was not specified large enough. It could also happen that the last grid in a cycle has more than *nbxtgt* cells if the parameters *ratio*1, *ratio*2, *fracrf*, and *fracdf* are chosen inappropriately. Default values for *ratio*1 and *ratio*2 are 0.4 and 0.8, respectively. These values are invoked when a user specifies neither or specifies both to be zero. The default values for *fracrf* and *fracdf* are 0.2 and 0.4 and are invoked in a similar manner.

Note: When parameters *fracrf* and *fracdf* are used in the code to create a new grid, the new grid generally will have between $n1 * ncells$ and $n2 * ncells$ cells, where *ncells* denotes the number of cells in the previous grid, $n2 = 1. + 7. * fracrf$, and $n1 = n2 - (7./8.) * fracdf$.

4–5.6 Boundary Layer Coupling ($BOU)

The **$BOU** keyword enables an approximate coupled boundary layer/inviscid potential flow solution to be made by applying infinite swept tapered wing boundary layer approximations on selected lifting surfaces of the configuration.

The boundary layer is defined on lifting surfaces of the configuration by defining a series of "ribs" and their associated wakes. A rib is a list of network corner points running from the trailing edge of the "upper" surface to the leading edge of the "upper" surface, joining the leading edge of the "lower" surface and continuing to the trailing edge of the "lower surface." The "upper" and "lower" surfaces may be arbitrary with regard to configuration geometry but the trailing edge points must be the same location. The wake associated with a rib is a wake network joining the trailing edge of the lifting surface which has a column of corner points, the first of which is at the same location as the trailing edge coordinate of the rib. In addition, only type 14 and 23 wakes may be used as wake surfaces for boundary layer calculations.

Boundary layer ribs are collected in groups with a set of parameters associated with each group. Controls are provided to allow boundary layer coupling to begin for all ribs in a group at any particular grid in a grid sequenced or solution adaptive grid run. In general, it is recommended that distinct lifting surfaces should be defined in separate groups. Sometimes it is advantageous to separate ribs on a single lifting surface into more than one group. For example, outboard wing pressure distributions on early grids may be so poorly described that the boundary layer solution may fail. In such a case, a coupled solution may still be obtained by deferring initiation of the coupling to a subsequent grid, after sufficient grid density has accumulated to present a reasonable pressure to the boundary layer code.

Modeling guidelines for use with boundary layer coupling are presently in their infancy. For the present, it is recommended that fairly fine paneling (e.g., cosine spaced paneling) be used both on the lifting surfaces and the associated wake surface in the neighborhood of the trailing edge of the lifting surface. The wake associated with a lifting surface should extend at least 1/2 chord length downstream of the trailing edge of the lifting surface before joining a type 18 wake and exiting the computational grid.

This keyword defines parameters used globally for the boundary calculation, parameters which independently affect the "upper" and "lower" surfaces of the boundary layers on all ribs, parameters which affect groups of ribs considered together and parameters which affect each single rib. The parameters used globally include:

- *reinf* - The Reynolds number per unit length in the length units used to define the input geometry. See *crefg* below.
- *tinf* - The freestream static temperature in degrees Kelvin.
- *fomgbl* - Damping factor for transpirations used on successive Newton steps. The transpirations used are $fomgbl * h^n + (1 - fomgbl) * h^{n-1}$, where h^i is the boundary layer transpirations for the $i-$th Newton step.

- *blfnopt* - Flag for determining how many copies of the boundary layer solution files should be saved.
- *blnew* - Number of Newton cycles to continue viscous-inviscid coupling.
- *xt* - Convergence tolerance of boundary layer solution. Recommended value is 0.0001 or less.
- *dmpsep* - Desired increment above the default change in upstream perpendicular Mach number of 1.4 for a final shock induced separation criterion. Shock induced separation will occur for upstream perpendicular Mach number changes of 1.4 + *dmpsep*.
- *pr* - Prandtl number. $pr = 0.71$ for air.
- *prt* - Turbulent Prandtl number. $prt = 0.9$ is the recommended value.
- *eta2* - Value of the first boundary layer grid point normal to the surface. $eta2 = 0.0001$ is the recommended value.
- *etamax* - Maximum grid point normal to the surface for boundary layer grid. $etamax = 15.0$ is the recommended value.

If any of *xt*, *pr*, *prt*, *eta2* or *etamax* are zero (blank), they are set to their recommended values.

The following parameters are defined independently for the upper and lower surfaces of all boundary layer ribs. The upper and lower surfaces are defined in terms of where the boundary layer preprocessor discovers the stagnation point at the leading edge of the lifting surface.

- *fngrpl* - Number of groups of boundary layer ribs.
- *namdau* - File name containing "upper surface" boundary layer controls. If none is provided, values read in below are used. The input conventions for the boundary layer controls are defined in the documentation for the **A411** program.
- *namdal* - File name containing "lower surface" boundary layer controls. If none is provided, values read in below are used. The input conventions for the boundary layer controls are defined in the documentation for the **A411** program.
- *namde* - File name containing boundary layer controls applied which affect primarily boundary layer grid generation. If none is provided, default values are used. The input conventions for the boundary layer controls are defined in the documentation for the **A411** program.
- *gofac* - Turbulence model factor. A value of 0. indicates use of A411's "old eddy viscosity" model and a value of 1.0 indicates use of A411's "new eddy viscosity" model. Values between 0.0 and 1.0 compute an eddy viscosity which is a linear combination of the two.

- *pfac* - Parameter controlling the imposition of log-law behavior in the boundary layer. If a boundary layer is separating or is close to separation, an increase in this parameter will tend to make the boundary layer remain attached (keep the skin friction from becoming zero) with only modest effects on the δ^*. Recommended values range from 0.0 for purists, in the 0.2 to 0.3 range for some transport wings and up to 1.0 for those who do not want the boundary layer to separate at all, no matter what.

- *east* - Parameter controlling anisotropy in the eddy viscosity of the boundary layer. Recommended value is 1.0 for no anisotropy, and 0.4 for anisotropies observed in some experiments. For the infinite swept tapered wing approximations, anisotropic eddy viscosities may not make good sense.

- *finv* - Pseudo inverse flag. If the boundary layer gets close to separation (skin friction $C_f < 0.0002$), the boundary layer solver enters a pseudo inverse mode if this flag is set to zero. In this mode, the skin friction is held constant at 0.0002, and there may be a mismatch in the outer edge velocity of the boundary layer and the inviscid edge velocities. The boundary layer solution attempts to minimize this difference by imposing a smooth and rapid growth in δ^*. If the flag is set to 1.0, and the boundary layer is close to separation, the boundary layer solution enters a "forbidden" state and δ^* remains constant from that point on.

- *jeta* - Number of points in the boundary layer grid normal to the surface. *jeta* = 40 is the recommended value. *jeta* ≤ 50.

- *prof* - Flag indicating whether to write out velocity profiles at every station on the boundary layer grid. This produces some very large files. A zero value is default, indicating no profiles are to be written out. A value of 1.0 will produce files containing boundary layer profiles.

- *blndm*1 - Blending parameter for shock boundary layer interaction. Recommended value is 0.

- *blndm*2 - See *blndm*1 above. Recommended value is 0.

- *blndr*1 - Blending parameter for skin friction in shock boundary layer interaction. Recommended value is 0.

- *blndr*2 - See *blndr*1 above. Recommended value is 0.

- *fwedge* - Flag indicating the desire to add to the boundary layer growth in the area of a shock. A zero value indicates that the "shock wedge" will be computed internally and a value of 1.0 indicates that the shock wedge will be determined from the following parameters.

- *fwdg*1 - Scale factor on δ^* in the shock wedge for the first shock encountered after the attachement line. Recommended value is 1. for purists.

- *fwdg2* - Scale factor on δ^* in the shock wedge for the second shock encountered after the attachement line. Recommended value is 1. for purists.
- *fblnd11* - Blending factor for first shock wedge. Recommended value is 1.
- *fblnd21* - Blending factor for second shock wedge. Recommended value is 1.

The following parameters are defined for each group of boundary layer ribs.

- *nrib* - The number of ribs in this group.
- *coupflg* - The coupling flag for the group. A zero value means solve the boundary layer equations only for post-processing and do not couple the resulting transpirations to the inviscid solution. A non-zero value means begin coupling the boundary layer transpirations to the inviscid solution on the *coupflg* grid of a grid-sequenced or solution adaptive grid run.
- *crefg* - Configuration model scale for comparison to test data. Often a configuration is defined in full-scale dimensional units, but comparisons are desired to wind tunnel test data on models which are not full scale models. The *crefg* factor is applied to correctly scale the Reynolds number per unit length in *reinf* to the correct model scale for comparison with test data.

The following parameters are read in for each boundary layer rib.

- *nnetr* - Number of networks making up this rib.
- *swple* - Leading edge sweep for this rib.
- *swpte* - Trailing edge sweep for this rib.
- *xctl* - Trip location (as percent of rib chord) for transition to turbulent boundary layer equations for the "lower" surface of the rib.
- *xctlu* - Trip location (as percent of rib chord) for transition to turbulent boundary layer equations for the "upper" surface of the rib.
- *nsuc* - Number of terms in suction table.
- *issuc* - Surface selection for suction.
- *net* - Network name or index for a network taking part in a rib.
- *colst* - Column index of the first point on the network taking part in the rib. (Note the rib convention of trailing edge upper surface to leading edge upper surface to leading edge lower surface to trailing edge lower surface for ordering of network columns and rows defining a rib.)

- *rowst* - Row index of the first point on the network taking part in the rib. (Note the rib convention of trailing edge upper surface to leading edge upper surface to leading edge lower surface to trailing edge lower surface for ordering of network columns and rows defining a rib.)

- *colend* - Column index of the last point on the network taking part in the rib. (Note the rib convention of trailing edge upper surface to leading edge upper surface to leading edge lower surface to trailing edge lower surface for ordering of network columns and rows defining a rib.)

- *rowend* - Row index of the last point on the network taking part in the rib. (Note the rib convention of trailing edge upper surface to leading edge upper surface to leading edge lower surface to trailing edge lower surface for ordering of network columns and rows defining a rib.)

- *ssuc* - Arc length parameter for application of suction or blowing.

- *vssuc* - Suction or blowing transpiration, ratioed to the outer edge velocity of the boundary layer.

Card Images

$BOUndary Layer								**boul**
reinf	*tinf*	*fomgbl*	*blfnopt*	*blnewt*	—	-	-	**bou2**
xt	*dmpsep*	*pr*	*prt*	*eta2*	*etamax*	-	-	**bou3**
fngrpl	*namdau*	*namdal*	*namde*	-	-	-	-	**bou4**
gfacu	*pfacu*	*eastu*	*finvu*	*jetau*	*profu*	-	-	**bou5**
blndm1u	*blndm2u*	*blndr1u*	*blndr2u*	-	-	-	-	**bou6**
fwedgu	*fwdg1u*	*fwdg2u*	*fblnd1u*	*fblnd2u*	-	-	-	**bou7**
gfacl	*pfacl*	*eastl*	*finvl*	*jetal*	*profl*	-	-	**bou8**
blndm1l	*blndm2l*	*blndr1l*	*blndr2l*	-	-	-	-	**bou9**
fwedgl	*fwdg1l*	*fwdg2l*	*fblnd1l*	*fblnd2l*	-	-	-	**bou10**
! Cards **boul1** through **bou13** must be repeated *fngrpl* times.								
nrib	*coupflg*	*crefg*	-	-	-	-	-	**boul1**
! Cards **bou12** and **bou13** is provided *nrib* times.								
nnetr	*swple*	*swpte*	*xctl*	*xctu*	*nsuc*	-	-	**bou12**
! Card **bou13** is provided only if *nsuc* is non zero.								
issuc	-	-	-	-	-	-	-	**bou13**
! Card **bou14** is provided *nnetr* times.								
net	*colst*	*rowst*	*colend*	*rowend*	-	-	-	**bou14**
! Card **bou15** is provided *nsuc* times if $nsuc \neq 0$.								
ssuc	*vssuc*	-	-	-	-	-	-	**bou15**

Input Data Description

Card	Field	Name	Description
bou1	all	*icard*	Keyword $BOUndary layer.
bou2	1	*reinf*	Reynolds number per unit length, typically per unit chord.
	2	*tinf*	Freestream static temperature.
	3	*fomgbl*	Damping factor for computing transpiration. *fomgbl* times current boundary layer transpiration plus 1 − *fomgbl* times the previous transpiration is used as a transpiration on the next Newton step in the solution process. Recommended starting value is 0.5, but other values may provide solutions at lower costs, depending on the configuration.
	4	*blfnopt*	Flag indicating how many different boundary layer solution files to save. Recommended value is 0, meaning save only the latest boundary layer solution file. A value of 1. will save the boundary layer solution files for each grid. A value of 2. will save them for each grid and Newton step. As these files can be quite large, it is strongly recommended that only a value of 0 be used.
	5	*blnewt*	Number of Newton steps to continue viscous-inviscid coupling. Recommended value ≈ 3 for starters.
bou4	1	*fngrpl*	Number of groups of boundary layer ribs. All ribs in a group use the same coupling strategy. Coupling strategies allow the user to specify that the boundary layer be computed only in postprocessing, or that the ribs in the group be coupled in on the *coupflg* grid.
	2	*namdau*	File name for the upper surface rib DA file used by BLGL to set up the boundary layer calculations. Preparation of these files are discussed in the A411 Users' Manual. They provide the user with more detailed control over the boundary layer system, but most users will not require this detailed capability, as the following input parameters provide most controls of interest to users.
	3	*namdal*	File name for the lower surface DA file. Comments in *namdau* above apply.

Card	Field	Name	Description
bou4	4	*namde*	File name for the DE file used to control some features of the boundary layer grid and solution strategy. Described in the A411 Users' Manual. Most users will not need to change any of these parameters. It is recommended that the default values (no files specified for *namdau*, *namdal* and *namde*) The format for the three filenames is A, ten character filenames.
bou5	1	*gfacu*	Turbulence model flag for upper surface.
	2	*pfacu*	Log layer control for upper surface.
	3	*eastu*	Anisotropy control for eddy viscosity on upper surface.
	4	*finvu*	Pseudo inverse flag for upper surface.
	5	*jetau*	Number of points in boundary layer grid normal to upper surface.
	6	*proflu*	Flag to turn on creation of file containing velocity profiles in the upper surface boundary layer.
bou6	1	*blndm1u*	Blending parameter for shock boundary layer interaction for upper surface.
	2	*blndm2u*	Blending parameter for shock boundary layer interaction for upper surface.
	3	*blndr1u*	Blending parameter for skin friction in shock boundary layer interaction for upper surface.
	4	*blndr2u*	Blending parameter for skin friction in shock boundary layer interaction for upper surface.
bou7	1	*fwedgel*	Flag for user-supplied shock wedge data.
	2	*fwdg1l*	Scale factor for δ^* for the first shock encountered on the lower surface.
	3	*fwdg2l*	Scale factor for δ^* for the second shock encountered on the lower surface.
	4	*fblnd1l*	Blending factor for the user-supplied shock wedge for the first shock encountered on the lower surface.
	4	*fblnd2l*	Blending factor for the user-supplied shock wedge for the second shock encountered on the lower surface.

Card	Field	Name	Description
bou8	1	*gfacl*	Turbulence model flag for lower surface.
	2	*pfacl*	Log layer control for lower surface.
	3	*eastl*	Anisotropy control for eddy viscosity on lower surface.
	4	*finvl*	Pseudo inverse flag for lower surface.
	5	*jetal*	Number of points in boundary layer grid normal to lower surface.
	6	*profll*	Flag to turn on creation of file containing velocity profiles in the lower surface boundary layer.
bou9	1	*blndm1l*	Blending parameter for shock boundary layer interaction for lower surface.
	2	*blndm2l*	Blending parameter for shock boundary layer interaction for lower surface.
	3	*blndr1l*	Blending parameter for skin friction in shock boundary layer interaction for lower surface.
	4	*blndr2l*	Blending parameter for skin friction in shock boundary layer interaction for lower surface.
bou10	1	*fwedgel*	Flag for user-supplied shock wedge data.
	2	*fwdg1l*	Scale factor for δ^* for the first shock encountered on the lower surface.
	3	*fwdg2l*	Scale factor for δ^* for the second shock encountered on the lower surface.
	4	*fblnd1l*	Blending factor for the user supplied shock wedge for the first shock encountered on the lower surface.
	4	*fblnd2l*	Blending factor for the user supplied shock wedge for the second shock encountered on the lower surface.
bou11	1	*nrib*	Number of boundary layer ribs in this group. Total number of ribs in all groups ≤ 100.
	2	*coupflg*	Coupling Flag for ribs in this group. = 0 No coupling. Post-processed Boundary Layer = 1 Couple boundary layer to inviscid solution beginning on the first grid. = n Couple boundary layer on the n-th grid (n $\leq$ 10).
	3	*crefg*	Scaling factor for rescaling chord to other than input units for the purpose of local evaluation of Reynolds number.
	4	*fatlin*	Attachment line flag.

Card	Field	Name	Description
bou13	1	*issuc*	Surface for suction/blowing. Upper surface = 1.0 and lower surface =2.0
bou12	1	*nnetr*	Number of networks in this rib.
	2	*swple*	Leading edge sweep angle.
	3	*swpte*	Trailing edge sweep angle.
	4	*xctl*	Lower surface trip location as percent of chord.
	5	*xctu*	Upper surface trip location as percent of chord.
	6	*nsuc*	Number of points in suction/blowing table.
bou14	1	*net*	Network name or index for network making up part of this rib.
	2	*colst*	Starting column number of the part of this network making up this rib.
	3	*rowst*	Starting row number of the part of this network making up this rib.
	4	*colend*	Ending column number of the part of this network making up this rib.
	5	*rowend*	Ending row number of the part of this network making up this rib.
bou15	1	*ssuc*	Parametric arc length for suction/blowing table.
	2	*vssuc*	Suction or blowing at this arc length station.

4–6 SURFACE & BOUNDARY CONDITIONS

4–6.1 Panel Network Fundamentals

The near field surface geometry is described to the program in the form of **networks** of **panels**. The concepts of panels, networks, and associated conventions are described below.

A **panel** is an approximation to a small portion of a 3D surface (see figure 4.9). It is completely defined by its four **corner points**. The **panel edges** are straight lines connecting neighboring pairs of corner points. Thus, a network surface is continuous. It is possible to have one of the edges of a panel collapsed to form a triangular panel (but each of the two coincident points has its own identity). A panel can be subdivided into triangular subpanels by joining midpoints of the panel edges. The middle four triangles form a flat parallelogram which lies in the **mean plane** of the panel. The quality of the panel as a discrete element representing a given surface could be measured by the normal distance between the corner points and the average plane. However, for the purpose of efficient evaluation of the finite element operators in TranAir (see Theory Document), a quadrilateral panel is divided into two triangular subpanels by joining one pair of opposite corner points. Associated with each triangular subpanel is a **normal** direction. The curvature is simulated by allowing linear variation of the normal vector over the subpanel.

A rectangularly indexed array of panel corner points forms a **network** (see figure 4.10). A typical network has nm rows and nn columns of corner points. The end columns or rows of the network can be collapsed (see figure 4.11), however, the code does not allow networks with internal columns or rows collapsed or with both an end row and an end column collapsed.

The specific convention used in TranAir to identify network edges, corner points, etc., is shown in figure 4.10. Each network has an **upper** and a **lower** surface. The upper surface of a network is given by a vector obtained from a cross product $\hat{n} \otimes \hat{m}$, where the $\hat{m}$ and $\hat{n}$ are unit vectors along the row and the column directions respectively. For the network oriented in the manner shown in figure 4.10, the upper surface is the front of the page and the lower surface is the back of the page. That is, the normal pointing out of the upper surface is as shown in figure 4.10. It is customary to define networks such that the outward normal vector points into the flow. The program needs only the corner point data ordered by columns to completely define all the **attributes** of the network.

The most common way to provide input geometry is through direct specification of corner points. The keyword associated with this data block is **$POI**. Many other simpler geometries can be computed within the program from a relatively small amount of input data. Commonly used geometry preprocessors are listed below.

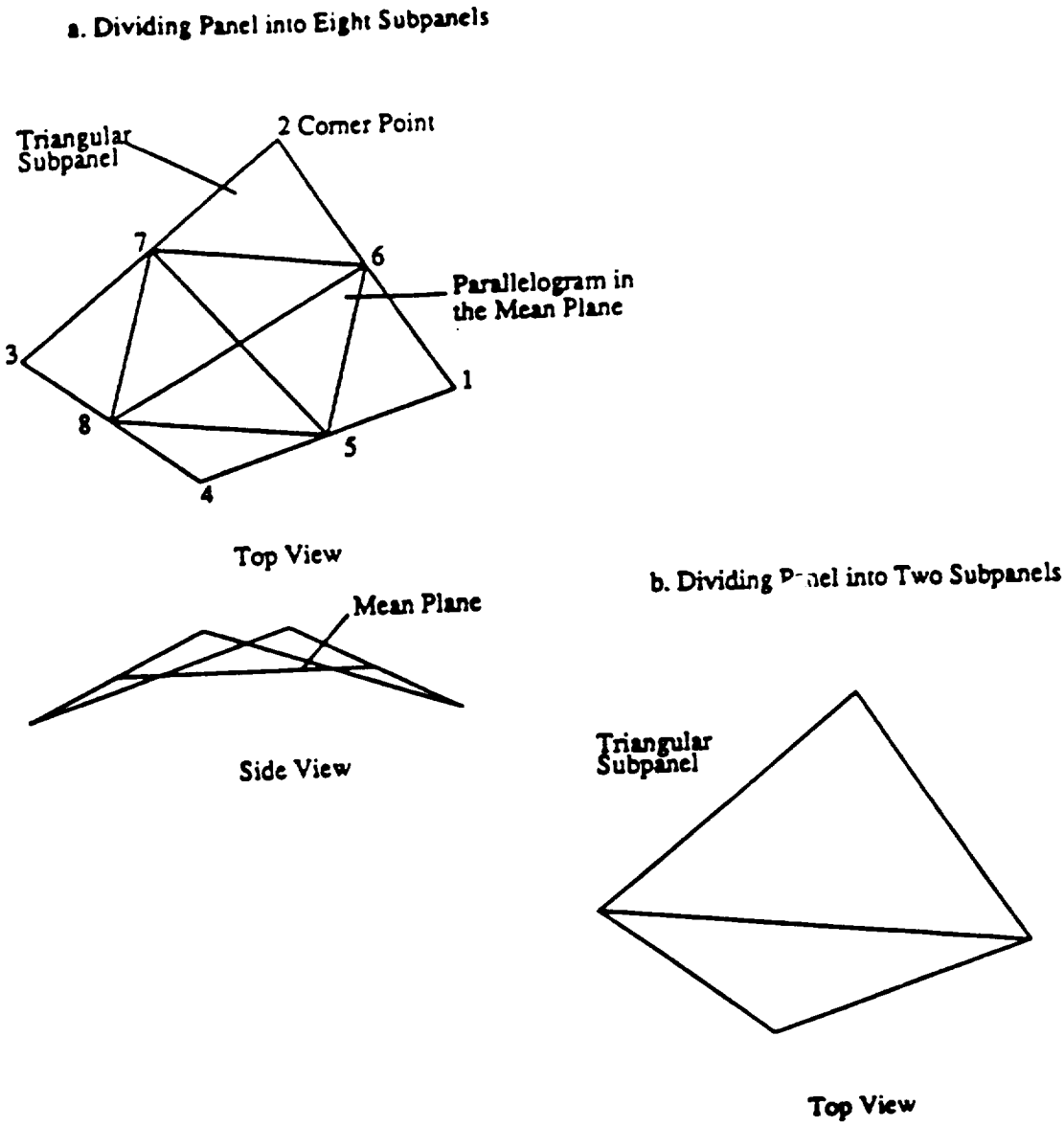


Figure 4.9: Panels, Corner Points, and Associated Conventions

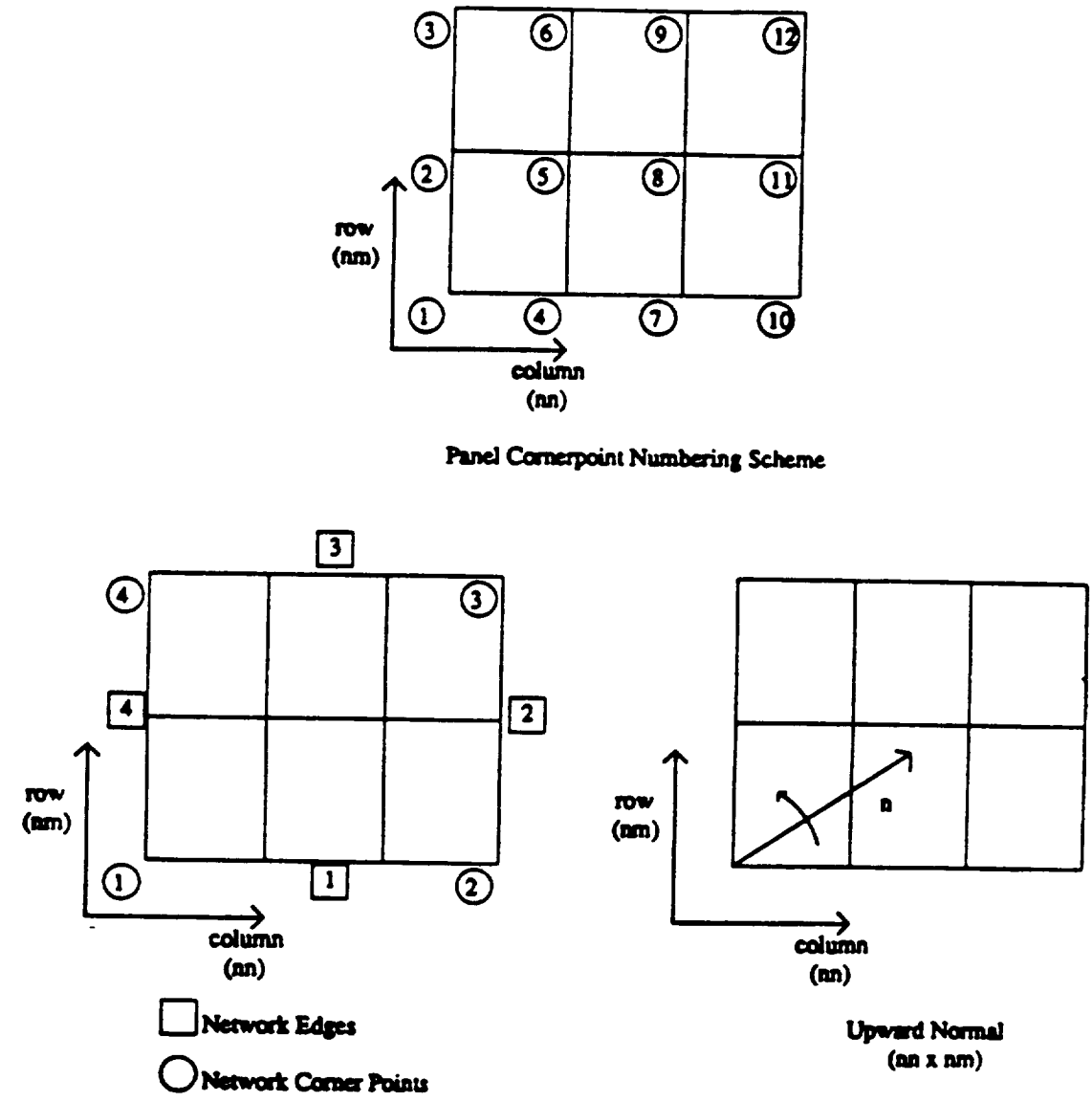
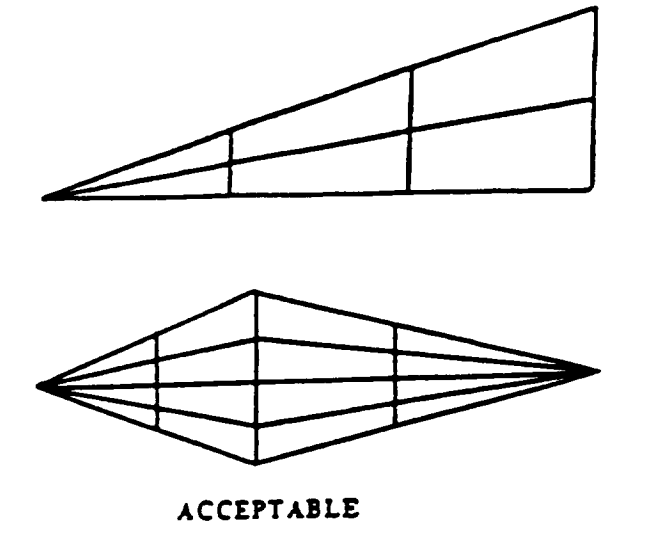


Figure 4.10: Networks and Associated Conventions



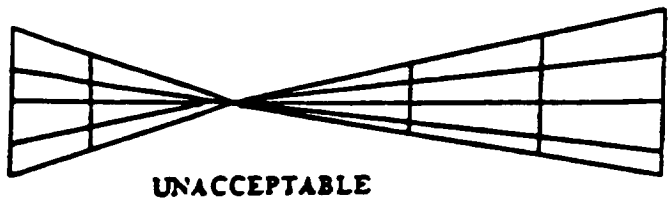


Figure 4.11: Unacceptable and Acceptable Networks

Keyword	Geometric features of the input Network
$POI	Corner point data provided directly to the program.
$TRA	Geometry of a trailing wake network computed in program.
$CIR	Geometry of a circular arc type network computed by the program.
$QUA	Geometry of a quadrilateral network computed by the program.

The networks may be input in any order.

The panel density of a network can be changed, via linear interpolation existing points, using the parameter *dnsm* (see figure 4.12). The rows and columns can be interchanged using parameter *mnsw* (see figure 4.13). This changes the direction of the network normal. The parameter *mnsw* is useful if the geometry of the input network is such that the upper surface normal does not point into the flow.

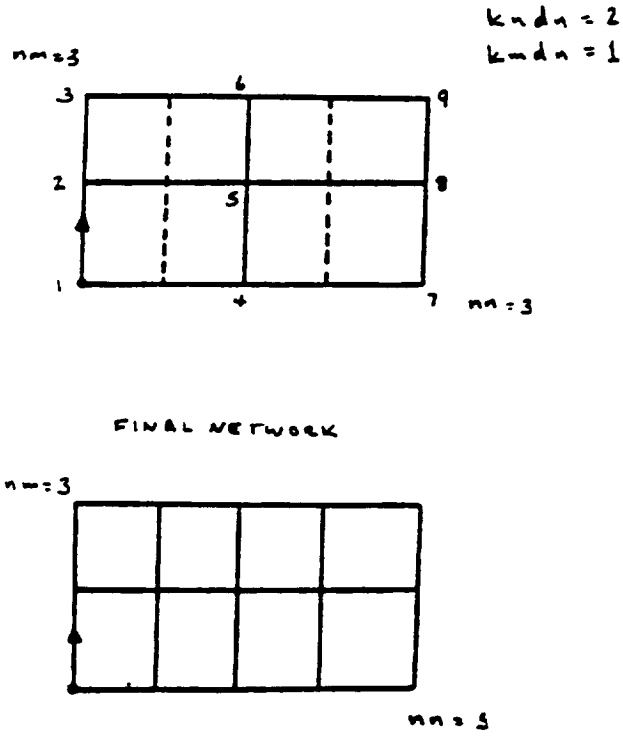


Figure 4.12: Changing the Panel Density of the Networks

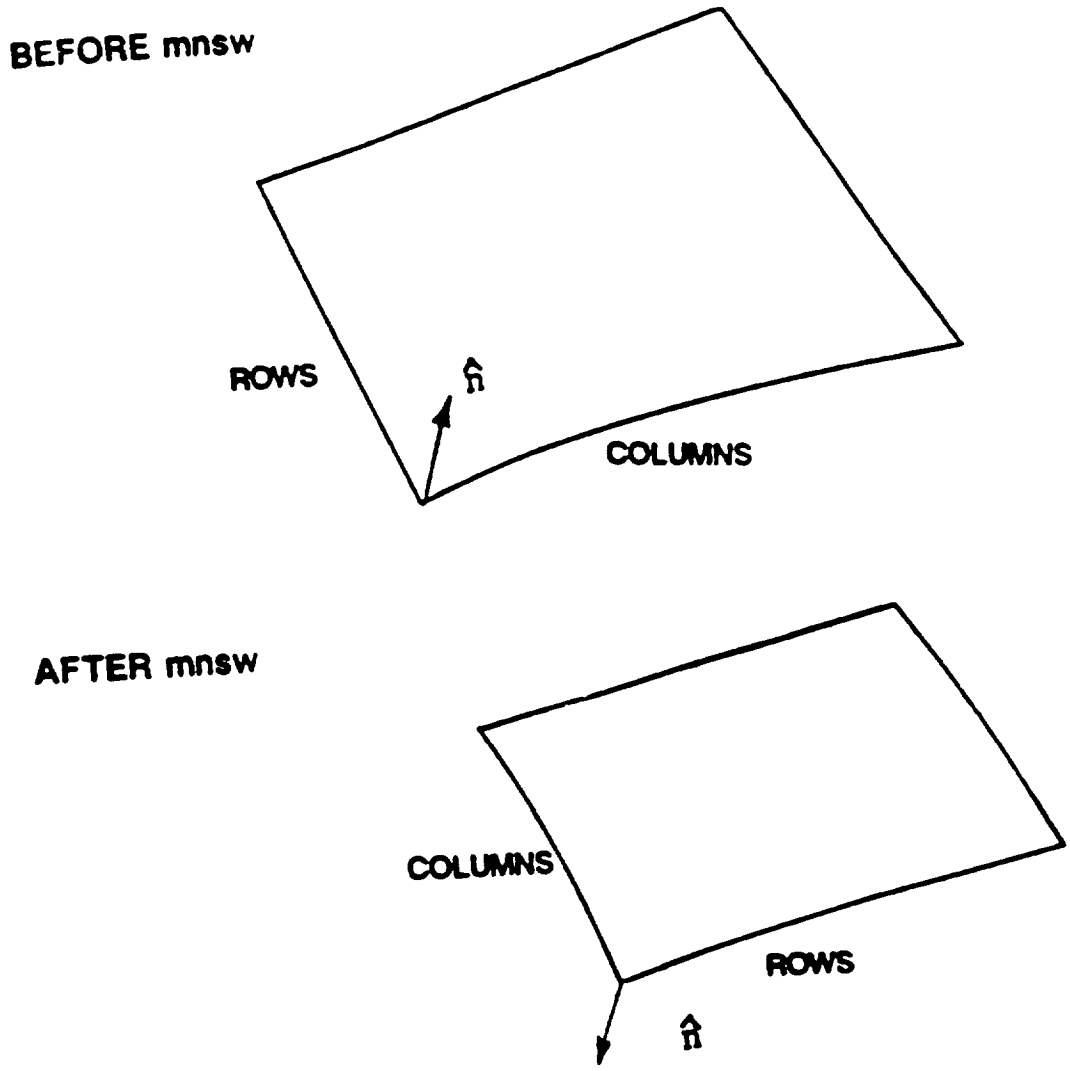


Figure 4.13: Changing the Outward Normal of the Networks

4–6.2 Defining a Surface Network and Common Boundary Conditions ($POI)

For each network, a unique boundary condition type has to be selected. The inputs specifying the boundary conditions are also part of the input associated with one of the data blocks specifying the geometry. The primary vehicle for specifying boundary conditions on a given network is the parameter *kt*. Most commonly used options are listed below.

The following notation is used in the boundary condition equations.

Notation	Description
$\vec{V}_\infty$	The freestream velocity.
ϕ	The perturbation potential.
$\vec{V} = V_\infty + \vec{\nabla}\phi$	The total velocity.
ρ	The density.
$\vec{W}$	The total mass flus $\rho\vec{V}$.
C_p	The pressure coefficient.
$\vec{R}$	The position vector.
$\hat{n}$	The unit normal vector.
$\hat{t}$	The local tangent vector in the direction of the corner point column.
μ	Doublet wake strength.
All other symbols	Defined in the text.
Subscripts u and l	The upper and lower surface values, respectively.
Subscript ∞	Conditions at freestream.

The following input data are common to all the forms network input and is listed once. This avoids the repetition of these definitions for each type of network input.

Common Input Data Description for $POI

Card	Field	Name	Description
poi1	1	*icard*	Keyword **$POI**.
poi2	1	*kn*	Number of networks in this data block.
poi3	1	*kt*	Specifies the boundary condition. More than one network with the same boundary condition can be defined in the **$POI**, **$TRA**, **$QUA**, and **$CIR** data blocks. Simply specify the number of networks (*kn*) and repeat the remaining lines of input *kn* times.
	4	*lrhsc*	Option for *bet* specification (location and form). The order follows the order of the network points. = 0.0 Panel center points and continuous over the network. = 1.0 Panel corner points and continuous over the network. (Recommended) = 4.0 Panel center point and continuous over the column. = 5.0 Panel corner point and continuous over the column.
	5	*mnsw*	Parameter to control switching the network normal from one surface to the other by switching rows and columns. = 0. No. = 1. Yes.
	6	*dnsm*	Parameter to control coarsening or densing the network rows and columns. = 0. No. = 1. Yes. In this case, card **poi6** must be included.

Card	Field	Name	Description
poi4	1	*nm*	Number of points in a column of a network.
	2	*nn*	Number of columns in a network.
	8	*ntnm*	Name of the network in columns 71-80. The names associated with networks can be used later to process trailing wakes, force partial edge abutments, or associate material properties. Format (**A**).
poi5	1-6	x, y, z	x, y, z coordinates of the network corner points. Network corner points are to be entered two per line (6 numbers per line) until a column is exhausted. The new column is to be started on the next line. In all, the program expects $nm * nn$ points in groups of nm points.
poi6	1	*kndn*	Parameter to specify alteration of the number of panel columns. = 1.0 Do not change the number of panel columns. = 2.0 Double the number of panel columns. = 3.0 Triple the number of panel columns. = etc. = minus a number will reduce the number of panel columns. The number must be exactly divisible by the number of panel columns (nn-1).
	2	*kmdn*	Parameter to alter the number of panel rows. Inputs are the same as *dn*, but applied to the panel rows.

Impermeable Surface (kt=1)

$$\vec{W}_u \cdot \hat{n} = 0. \tag{4.9}$$

$$\phi_l = 0 \tag{4.10}$$

Card Images

$POInts								poi1
kn	-	-	-	-	-	-	-	poi2
1.	-	-	-	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card poi6 is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6

Thin Surface (kt=2)

With this boundary condition, a number of sub-options may be specified through the input parameter tkw. These correspond to a variety of modeling conditions:

- tkw=0 - Thin surface with neither thickness nor camber simulation:

$$\vec{W}_u \cdot \hat{n} = 0. \tag{4.11}$$
$$\vec{W}_l \cdot \hat{n} = 0. \tag{4.12}$$

- tkw=1 - Thin surface with simulated thickness but no camber simulation:

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = bet1 \tag{4.13}$$
$$\vec{W}_u \cdot \hat{n} + \vec{W}_l \cdot \hat{n} = 0. \tag{4.14}$$

- tkw=2 - Thin surface with no thickness but with camber simulation:

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = 0. \tag{4.15}$$
$$\vec{W}_u \cdot \hat{n} + \vec{W}_l \cdot \hat{n} = bet2 \tag{4.16}$$

- tkw=3 - Thin surface with thickness and camber simulation:

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = bet1 \tag{4.17}$$
$$\vec{W}_u \cdot \hat{n} + \vec{W}_l \cdot \hat{n} = bet2 \tag{4.18}$$

- tkw=4 - Thin lower surface with simulated upper surface thickness:

$$\vec{W}_u \cdot \hat{n} = bet1 \tag{4.19}$$
$$\vec{W}_l \cdot \hat{n} = 0. \tag{4.20}$$

- tkw=5 - Thin upper surface with simulated lower surface thickness:

$$\vec{W}_u \cdot \hat{n} = 0. \tag{4.21}$$
$$\vec{W}_l \cdot \hat{n} = bet2 \tag{4.22}$$

- tkw=6 - Thin surface with simulated thickness on both upper and lower surfaces:

$$\vec{W}_u \cdot \hat{n} = bet1 \tag{4.23}$$
$$\vec{W}_l \cdot \hat{n} = bet2 \tag{4.24}$$

Card Images

$POInts								poi1
kn	*tkw*	-	-	-	-	-	-	poi2
1.	-	-	-	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card poi6 is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6
! First boundary condition for the right-hand side.								
! Required if *tkw* equals 1, 3, 4, or 6.								
*bet*1	*bet*1	*bet*1	*bet*1	*bet*1	*bet*1	-	-	poi7
*bet*1	*bet*1	*bet*1...	-	-	-	-	-	poi7
! Second boundary condition on the right-hand side.								
! Required if *tkw* equals 2, 3, 5, or 6.								
*bet*2	*bet*2	*bet*2	*bet*2	*bet*2	*bet*2	-	-	poi8
*bet*2	*bet*2	*bet*2...	-	-	-	-	-	poi8

Input Data Description

Card	Field	Name	Description
poi3	2	***tkw***	Thickness simulation control. = 0. No thickness or camber. = 1. Thickness, no camber. = 2. Camber, no thickness. = 3. Camber and thickness. = 4. Upper surface thickness. = 5. Lower surface thickness. = 6. Upper and lower surface thickness.
poi7	1-6	***bet*1**	Surface transpiration as required by the selected *tkw* option. For the location and the form of the input, see *lrhsc* on page 99.
poi8	1-6	***bet*2**	Surface transpiration as required by the selected *tkw* option. For the location and the form of the input, see *lrhsc* on page 99.

C-2

Fanface (kt=4) with perturbation mass flow specified

Original way to define flow into a fanface. Current recommended approach is to use kt=9.

$$\vec{W}_u \cdot \hat{n} = \rho_\infty \vec{V}_\infty \cdot \hat{n} + bet \quad (4.25)$$

$$\phi_l = 0 \quad (4.26)$$

Card Images

$POInts								**poi1**
kn	-	-	-	-	-	-	-	**poi2**
4.	-	-	*lrhsc*	*mnsw*	*dnsm*	-	-	**poi3**
nm	*nn*	-	-	-	-	-	*ntnm*	**poi4**
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	**poi5**
*x*13	*y*13	*z*13	-	-	-	-	-	**poi5**
*x*21	*y*21	*z*21	-	-	-	-	-	**poi5**
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	**poi6**
bet	-	-	-	-	-	-	-	**poi7**

Input Data Description

Card	Field	Name	Description
poi7	1-6	*bet*	Non-dimensional upper surface perturbation mass flow $(\rho V - \rho_\infty V_\infty)/\rho_\infty V_\infty$ normal to the surface. For the location and the form of the input, see *lrhsc* on page 99.

Bases (kt=5)

$$\phi_u = -\vec{V}_\infty \cdot \vec{R} \quad (4.27)$$

$$\phi_l = 0 \quad (4.28)$$

Card Images

$POInts								poi1
kn	-	-	-	-	-	-	-	poi2
5.	-	-	-	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6

Sample Networks (kt=6) determining field flow properties

$$\phi_u - \phi_l = 0. \quad (4.29)$$

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = 0 \quad (4.30)$$

Card Images

$POInts								poi1
kn	-	-	-	-	-	-	-	poi2
6.	-	-	-	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6

Fanface (kt=9) **with total mass flow specified**

$$\vec{W}_u \cdot \hat{n} = bet \tag{4.31}$$

$$\phi_l = 0 \tag{4.32}$$

Card Images

$POInts								poi1
kn	-	-	-	-	-	-	-	poi2
9.	-	-	*lrhsc*	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card poi6 is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6
bet	*bet*	*bet*	*bet*	*bet*	*bet*	-	-	poi7
bet	*bet*	*bet*...	-	-	-	-	-	poi7

Input Data Description

Card	Field	Name	Description
poi7	1-6	*bet*	Non-dimensional mass flow ($\rho V/\rho_\infty V_\infty$) along the upper surface normal. Since surface normals are outward from the upper surface, the value of *bet* is negative for flow into a surface. For the location and the form of the input, see *lrhsc* on page 99.

Porous Wall (kt=10)

$$-R\rho_\infty \hat{t} \cdot \vec{\nabla}\phi_u + \vec{W} \cdot \hat{n} = bet \quad (4.33)$$

$$\phi_l = 0 \quad (4.34)$$

Card Images

$PONts								poi1
kn	-	-	-	-	-	-	-	poi2
10.	*R*	*iopt2*	*lrhsc*	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card poi6 is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6
! Card poi7 is required if *iopt2* > 0.								
bet	*bet*	*bet*	*bet*	*bet*	*bet*	-	-	poi7
bet	*bet*	*bet*...	-	-	-	-	-	poi7

Input Data Description

Card	Field	Name	Description
poi3	3	*R*	Constant of proportionality.
	3	*iopt2*	Input control for specification of the right-hand sides. = 0.0 No right-hand side. = 1.0 Specified right-hand side.
poi7	1-6	*bet*	Non-dimensional mass flow ($\rho V/\rho_\infty V_\infty$) along the upper surface normal. Since surface normals are outward from the upper surface, the value of *bet* is negative for flow into a surface. For the location and the form of the input, see *lrhsc* on page 99.

Standard Wakes With Explicitly Convection Equation (kt=14)

This wake model is analogous to kt=18 wakes, but it permits simulation of thickness through transpiration, an essential capability for modeling the near-trailing edge behavior of boundary layers. In addition, it explicitly solves a convection equation to describe the behavior of the doublet strength along columns of the wake network. This permits the imposition of a known streamwise doublet variation on the wake.

For steady flows, the explicit equation imposed is identical to the one in kt=18 wakes, except that additional degrees of freedom are introduced into the matrix and then eliminated directly. The streamwise behavior of doublet strength varies in a known fashion in the case of time harmonic perturbations about a steady transonic state. Thus, this type of wake is used to model wake behavior in time harmonic perturbations about a transonic flow condition as is implemented in the Unsteady Harmonic extension of TranAir.

The cross stream behavior of the doublet strength is determined from a Kutta condition (or doublet matching condition to an abutting wake network). This boundary condition is imposed at the leading edge of the wake network (side 1). At all other wake points, the boundary conditions imposed are:

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = bet \tag{4.35}$$

$$\phi_u - \phi_l - \mu = 0 \tag{4.36}$$

Card Images

$POInts								**poi1**
kn	-	-	-	-	*cpnr*	-	-	**poi2**
18.	-	-	-	*mnsw*	*dnsm*	-	-	**poi3**
nm	*nn*	-	-	-	-	-	*ntnm*	**poi4**
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	**poi5**
*x*13	*y*13	*z*13...	-	-	-	-	-	**poi5**
*x*21	*y*21	*z*21...	-	-	-	-	-	**poi5**
! Card **poi6** is required if *dnsm* > 0.								
kndn	***kmdn***	-	-	-	-	-	-	**poi6**

Standard Wakes With Explicit Convection & Thickness (*kt*=16)

This wake model is analogous to *kt*=20 wakes, but it permits simulation of thickness through transpiration, an essential capability for modeling the near-trailing edge behavior of boundary layers. In addition, it explicitly solves a convection equation to describe the behavior of the doublet strength along columns of the wake network. This permits the imposition of a known streamwise doublet variation on the wake.

For steady flows, the explicit equation imposed is identical to the one in *kt*=18 wakes except that additional degrees of freedom are introduced into the matrix and then eliminated directly. The streamwise behavior of doublet strength varies in a known fashion in the case of time harmonic perturbations about a steady transonic state. Thus, this type of wake is used to model wake behavior in time harmonic perturbations about a transonic flow condition as is implemented in the Unsteady Harmonic extension of TranAir.

The cross stream behavior of the doublet strength is determined from a Kutta condition (or doublet matching condition to an abutting wake network). Depending on the abutment characteristics of adjoining wakes, either the Kutta condition or doublet matching condition is imposed at the first point of the leading edge of the wake network (side 1). As in *kt*=20 wake networks, the doublet strength is constant in the cross stream direction. At all other wake points, the boundary conditions imposed are in effect:

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = 0 \tag{4.37}$$

$$\phi_u - \phi_l - \mu = 0 \tag{4.38}$$

Card Images

$POInts								poi1
kn	-	-	-	-	*cpnr*	-	-	poi2
18.	-	-	-	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6

Standard Wakes (kt=18)

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = 0 \quad (4.39)$$

$$\phi_u - \phi_l - \mu = 0 \quad (4.40)$$

All wakes must be specified with the upstream edge as network side edge one. This edge will be attached to the configuration or a downstream edge of another wake.

Card Images

$POInts								**poi1**
kn	-	-	-	-	-	-	-	**poi2**
18.	-	-	-	*mnsw*	*dnsm*	-	-	**poi3**
nm	*nn*	-	-	-	-	-	*ntnm*	**poi4**
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	**poi5**
*x*13	*y*13	*z*13...	-	-	-	-	-	**poi5**
*x*21	*y*21	*z*21...	-	-	-	-	-	**poi5**
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	**poi6**

Design Wakes (kt=19)

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = 0 \quad (4.41)$$

$$C_{p_u} - C_{p_l} = 0 \quad (4.42)$$

Card Images

$POInts								**poi1**
kn	-	-	-	-	-	-	-	**poi2**
19.	-	-	-	*mnsw*	*dnsm*	-	-	**poi3**
nm	*nn*	-	-	-	-	-	*ntnm*	**poi4**
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	**poi5**
*x*13	*y*13	*z*13...	-	-	-	-	-	**poi5**
*x*21	*y*21	*z*21...	-	-	-	-	-	**poi5**
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	**poi6**

Carry Over Wake (kt=20)

$$\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n} = 0 \quad (4.43)$$

$$\phi_u - \phi_l - \mu = 0 \quad (4.44)$$

These are the same as kt=18 wake boundary conditions, but are used to define a wake with a collapsed upstream edge (network edge one). This boundary condition is commonly used to carry over the wing wake to the side of the body as shown in the examples.

Card Images

$POInts								poi1
kn	-	-	-	-	-	-	-	poi2
20.	-	-	-	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13...	-	-	-	-	-	poi5
*x*21	*y*21	*z*21...	-	-	-	-	-	poi5
! Card **poi6** is required if $dnsm > 0$.								
kndn	*kmdn*	-	-	-	-	-	-	poi6

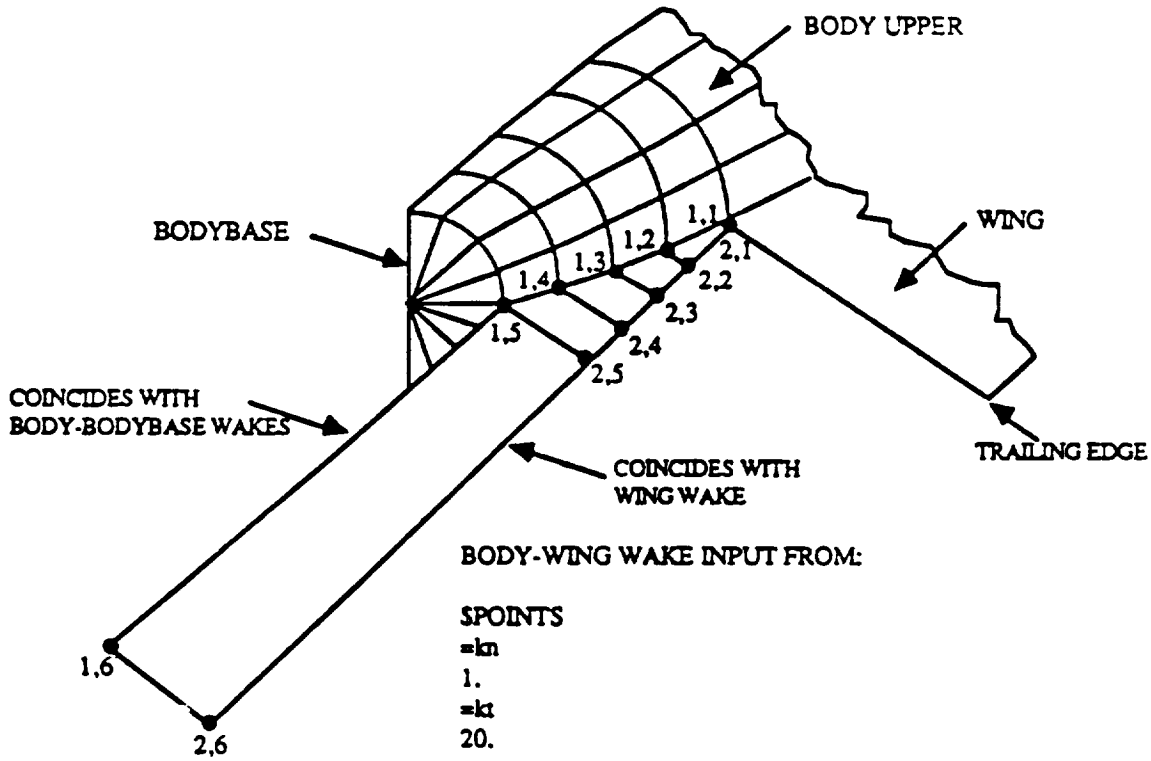


Figure 4.14: Input for Simple Body-Wing Wake

Upper Surface Design (*kt*=22)

$$C_{p_x} = bet \quad (4.45)$$

$$\phi_l = 0 \quad (4.46)$$

Card Images

$POInts								poi1
kn	-	-	-	-	-	-	-	poi2
22.	*iopt*1	-	*lrhsc*	*mnsw*	*dnsm*	-	-	poi3
nm	*nn*	-	-	-	-	-	*ntnm*	poi4
*x*11	*y*11	*z*11	*x*12	*y*12	*z*12	-	-	poi5
*x*13	*y*13	*z*13	-	-	-	-	-	poi5
*x*21	*y*21	*z*21	-	-	-	-	-	poi5
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	poi6
! Card **poi7** is required if *iopt*1 > 0.								
bet	*bet*	*bet*	*bet*	*bet*	*bet*	-	-	poi7
bet	*bet*	*bet*...	-	-	-	-	-	poi7

Input Data Description

Card	Field	Name	Description
poi3	2	*iopt*1	Input control for C_{p_x} specification. = 1.0 C_{p_u} specified. = 2.0 C_{p_l} specified. = 3.0 C_{p_a} specified, subscript *a* denotes averaging. = 4.0 C_{p_d} is specified, subscript *d* denotes the jump across the boundary. Value of *iopt*1 may be input as -1, -2, -3, or -4. In that case, the specified right-hand side is assumed to be zero and need not be input on card **poi7**. If *iopt*1 is positive then card **poi7** must be provided.
poi7	1-6	*bet*	Specified pressure coefficient as defined by *iopt*1. For the location and the form of the input, see *lrhsc* on page 99.

Thin Surface Design and Wake (*kt*=23)

$$C_{p_x} = bet1 \tag{4.47}$$

$$\vec{W}_x \cdot \hat{n} = bet2 \tag{4.48}$$

This boundary condition is commonly used to represent a powered plume.

Card Images

$POInts								**poi1**
kn	-	-	-	-	-	-	-	**poi2**
23.	*iopt1*	*iopt2*	*lrhsc*	*mnsw*	*dnsm*	-	-	**poi3**
nm	*nn*	-	-	-	-	-	*ntnm*	**poi4**
x11	*y11*	*z11*	*x12*	*y12*	*z12*	-	-	**poi5**
x13	*y13*	*z13...*	-	-	-	-	-	**poi5**
x21	*y21*	*z21...*	-	-	-	-	-	**poi5**
! Card **poi6** is required if *dnsm* > 0.								
kndn	*kmdn*	-	-	-	-	-	-	**poi6**
! Card **poi7** is required if *iopt1* > 0.								
bet1	*bet1*	*bet1*	*bet1*	*bet1*	*bet1*	-	-	**poi7**
bet1	*bet1*	*bet1...*	-	-	-	-	-	**poi7**
! Card **poi8** is required if *iopt2* > 0.								
bet2	*bet2*	*bet2*	*bet2*	*bet2*	*bet2*	-	-	**poi8**
bet2	*bet2*	*bet2...*	-	-	-	-	-	**poi8**

Input Data Description

Card	Field	Name	Description
poi3	2	*iopt*1	Input control for C_p specification. = 1.0 C_{p_u} is specified. = 2.0 C_{p_l} is specified. = 3.0 C_{p_a} is specified, subscript a denotes averaging. = 4.0 C_{p_d} is specified, subscript d denotes the jump across the boundary. Value of *iopt*1 may be input as -1, -2, -3, or -4. In that case, the specified the right-hand side is assumed to be zero and need not be input on card **poi7**. If *iopt*1 is positive, then card **poi7** must be provided.
	3	*iopt*2	Input control for specification of $\vec{W} \cdot \hat{n}$. = 1.0 $\vec{W}_u \cdot \hat{n}$ is specified. = 2.0 $\vec{W}_l \cdot \hat{n}$ is specified. = 3.0 $\vec{W}_a \cdot \hat{n}$ is specified, subscript a denotes averaging. = 4.0 $\vec{W}_d \cdot \hat{n}$ is specified, subscript d denotes the jump across the boundary. Value of *iopt*2 may be input as -1, -2, -3, or -4. In that case, the specified right-hand side is assumed to be zero and need not be input on card **poi8**. If *iopt*2 is positive, then card **poi8** must be provided.
poi7	1-6	*bet*1	Specified pressure coefficient as defined by *iopt*1. For the location and the form of the input, see *lrhsc* on page 99.
poi8	1-6	*bet*2	Specified normal mass flow as defined by *iopt*2. For the location and the form of the input, see *lrhsc* on page 99.

4–6.3 Defining a Surface Network and General Boundary Condition Equation (kt=30)

Caution: This capability is not completely developed.

Boundary conditions may also be specified by defining the left- and right-hand side terms as given in this section. Two boundary condition equations are specified at each surface point and are of the same form over the surface network.

This section has two sets of terms for defining the general boundary conditions. The first set is based on being computable with the A502 terms (reference 25, pages C-14 to C-19). The second is a new set of terms developed for TranAir.

Caution: For this method of boundary condition input, the user assumes responsibility for formulating a meaningful boundary value problem.

Caution: Because these boundary conditions are infrequently used, they have not been completely validated. If they are used, check the surface flow property results to see that they have been satisfied.

Card Images

$POInts								**poi1**
kn	-	-	-	-	-	-	-	**poi2**
kt	*nts*	*ntd*	*ipot*	*mnsw*	*dnsm*	-	-	**poi3**
nlopt1	*nropt1*	*nlopt2*	*nropt2*			-	-	**bc1**
! Repeat lines **poi4** through **poi8** *kn* times.								
nm	*nn*	-	-	-	-	-	*ntnm*	**poi4**
$x11$	$y11$	$z11$	$x12$	$y12$	$z12$	-	-	**poi5**
$x13$	$y13$	$z13$...	-	-	-	-	-	**poi5**
$x21$	$y21$	$z21$...	-	-	-	-	-	**poi5**
! Card **poi6** is required if $dnsm > 0$.								
kndn	*kmdn*	-	-	-	-	-	-	**poi6**
! First boundary condition right-hand side specified flow property.								
! Input only if term is present in right hand ($nropt = 1, 5, 6$ or 7)								
bet1	-	-	-	-	-	-	-	**poi7**
bet1	-	-	-	-	-	-	-	**poi7**
etc.								
bet1	-	-	-	-	-	-	-	**poi7**
! Second boundary condition right-hand side specified flow property.								
! Input only if term is present in right hand ($nropt = 1, 5, 6$ or 7)								
bet2	-	-	-	-	-	-	-	**poi8**
bet2	-	-	-	-	-	-	-	**poi8**
etc.								
bet2	-	-	-	-	-	-	-	**poi8**

Input Data Description

Card	Field	Name	Description
poi3	1	*kt*	=30. Boundary condition defined from selected terms.
	2	*nts*	Network source type, if *kt*=30. = 1. Yes. = 0. No.
	3	*ntd*	Network doublet type, if *kt*=30. = 1. Yes. = 0. No.
	4	*ipot*	Input control for right- and left-hand side terms. = 1. Use ***nlopt*** and *nropt* options from A502 (currently not operational). = 3. Use ***nlopt*** and *nropt* options listed below.

Card	Field	Name	Description
bc1	1	*nlopt*1	Specifies the terms on left-hand side of the first B.C. $= 1$ ϕ_u $= 2$ ϕ_l $= 3$ $(\phi_u + \phi_l) * 0.5$ $= 4$ $\phi_u - \phi_l$ $= 5$ $\vec{W}_u \cdot \hat{n}$ $= 6$ $\vec{W}_l \cdot \hat{n}$ $= 7$ $(\vec{W}_u \cdot \hat{n} + \vec{W}_l \cdot \hat{n}) * 0.5$ $= 8$ $\vec{W}_u \cdot \hat{n} - \vec{W}_l \cdot \hat{n}$ $= 9$ $\vec{V}_u \cdot \hat{n}$ $= 10$ $\vec{V}_l \cdot \hat{n}$ $= 11$ $(\vec{V}_u \cdot \hat{n} + \vec{V}_l \cdot \hat{n}) * 0.5$ $= 12$ $\vec{V}_u \cdot \hat{n} - \vec{V}_l \cdot \hat{n}$ $= 13$ C_{p_u} $= 14$ C_{p_l} $= 15$ $(C_{p_u} + C_{p_l}) * 0.5$ $= 16$ $C_{p_u} - C_{p_l}$
	2	*nropt*1	Specifies the terms on right-hand side of the first B.C. $= 1$ *bet* $= 2$ 0 $= 3$ $-\vec{V}_\infty \cdot \hat{n}$ $= 4$ $+\vec{V}_\infty \cdot \hat{n}$ $= 5$ $-\vec{V}_\infty \cdot \hat{n} +$ *bet* $= 6$ $+\vec{V}_\infty \cdot \hat{n} +$ *bet* $= 7$ $+\vec{V}_\infty \cdot \vec{R} +$ *bet*
	3	*nlopt*2	Specifies the terms on left-hand side of the second B.C. (See *nlopt*1.)
	4	*nropt*2	Specifies the terms on right-hand side of the second B.C. (See *nropt*1.)

Card	Field	Name	Description
poi7	1	*bet*1	Specified right-hand side for first boundary condition. **Comment:** One value per line is defined for each panel corner point. Values are ordered in the same manner as the network points.
poi8	1	*bet*2	Specified right-hand side for second boundary condition. **Comment:** One value per line is defined for each panel corner point. Values are ordered in the same manner as the network points.

Default Values

The geometry and the boundary conditions cards are necessary for every TranAir run. No defaults are provided.

4-6.4 Trailing Wakes ($TRA)

The keyword **$TRA** can be used to generate corner points for trailing wake networks that extend to a specified x value. The wake is constructed in such a manner that every column of corner points is parallel to the x axis, or the freestream. This wake attaches to the downstream edge of either the configuration network or the other wake network and are constructed as shown in figures 4.15 and 4.16. Note that only the kt=18 option may be used.

Card Images

$TRAiling Wakes								**tra1**
kn	-	-	-	-	-	-	-	**tra2**
kt	-	-	-	*mnsw*	*dnsm*	-	-	**tra3**
inat	*insd*	*xwake*	*twake*	-	-	*ntnm*	-	**tra4**

Input Data Description

Card	Field	Name	Description
tra1	all	*icard*	Keyword **$TRA**.
tra2	1	*kn*	Number of networks in this data block.
tra3	1	*kt*	Must equal 18.0.
tra4	1	*inat*	Index of the network to which the wake is attached. It is possible to supply the name of a network in this field. In this case, it is necessary to ensure that the name is less than 10 characters.
	2	*insd*	Edge number of network *inat* to which edge 1 of the wake is attached.
	3	*xwake*	x-coordinate of the downstream edge of the wake. This number should be larger than the downstream plane of the computational domain. See section 4-5.2 below.
	4	*twake*	Wake direction. = 0.0 Wake parallel to reference x-axis. = 1.0 Wake parallel to freestream. (Not operational.)
	71-80	*ntnm*	Name of the network in columns 71-80. The names associated with networks can be used later to process trailing wakes, force partial edge abutments, or associate material properties. This input is read in using (**A**) format into a character variable that is 10 characters long.

Examples

Figures 4.15 and 4.16 give examples of a wing and a body wake.

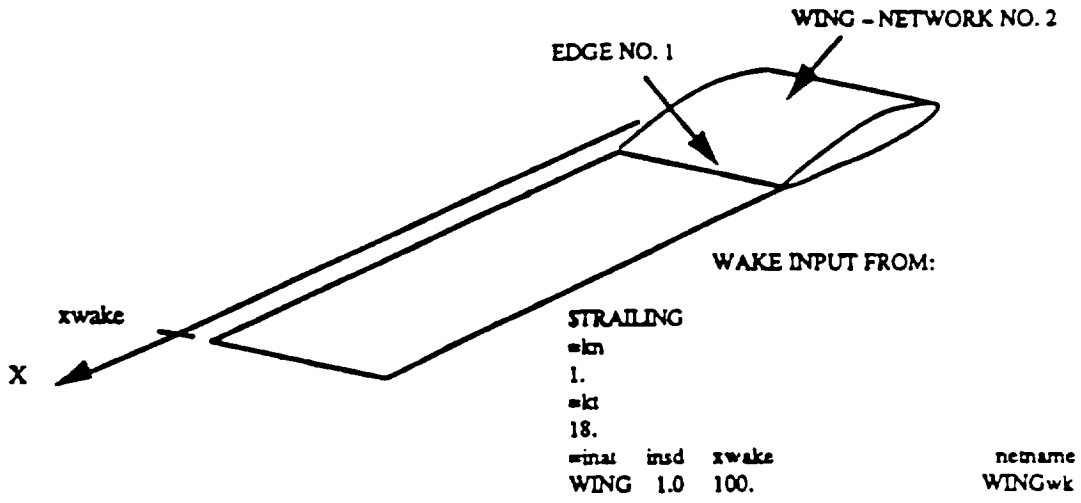


Figure 4.15: Input for Generation of Simple Wing Wake

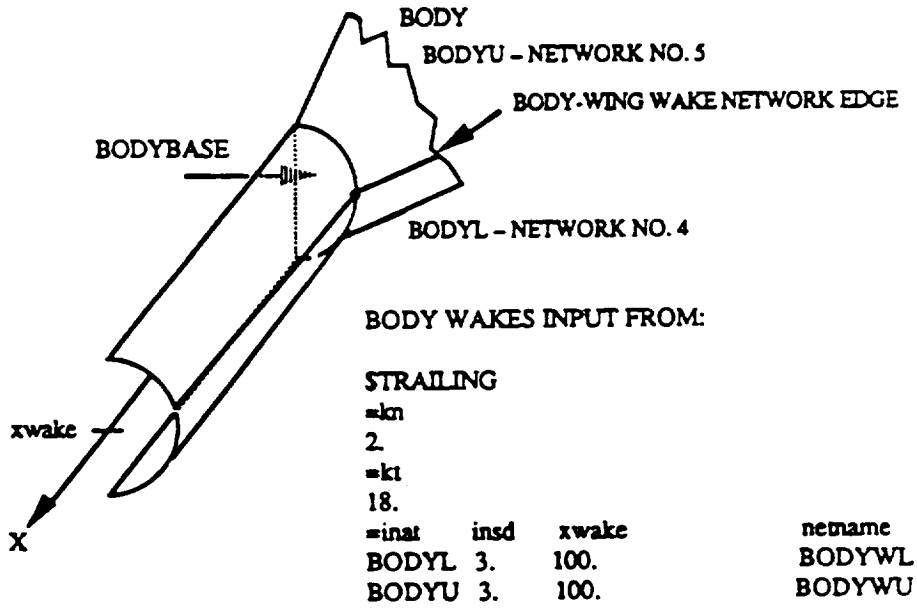


Figure 4.16: Input for Generation of Body-Bodybase Wake

4–6.5 Quadrilateral Networks ($QUA)

This keyword may be used to generate the network corner points for quadrilateral networks. The quadrilateral networks have straight edges and the panel division is made by linearly interpolating on the line joining the network corner points (see figure 4.17).

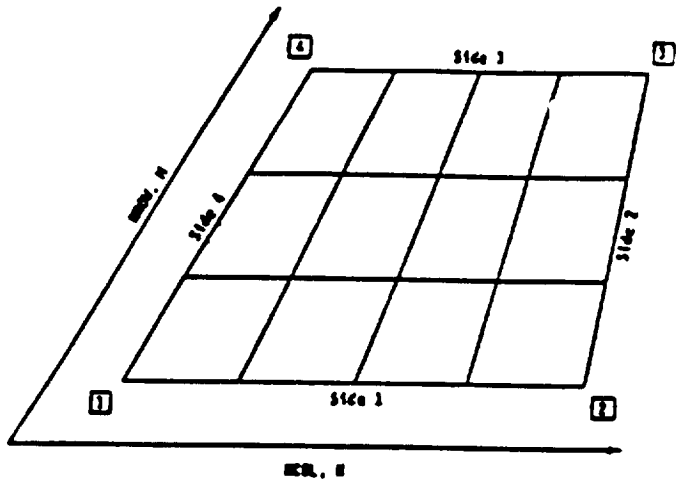


Figure 4.17: Quadrilateral Network

Note that it is possible to specify any boundary conditions on this network. The boundary condition data input follows the same cards as described earlier for the keyword **$POI** in section 4–6.2. In the following discussion, the description of boundary conditions is omitted.

Card Images

$QUAdrilateral Networks								**qua1**
kn	-	-	-	-	-	-	-	**qua2**
! card **qua3** has the same form as card **poi3** described before.								
kt	-	-	-	-	-	-	-	**qua3**
sc(1,1)	*sc*(2,1)	*sc*(3,1)	*sc*(1,2)	*sc*(2,2)	*sc*(3,2)	-	*ntnm*	**qua4**
sc(1,3)	*sc*(2,3)	*sc*(3,3)	*sc*(1,4)	*sc*(2,4)	*sc*(3,4)	-	-	**qua4**
nrow	-	-	-	-	-	-	-	**qua5**
ypc(1)	*ypc*(2)	etc.	-	-	-	-	-	**qua6**
ncol	-	-	-	-	-	-	-	**qua7**
xpc(1)	*xpc*(2)	etc.	-	-	-	-	-	**qua8**

Input Data Description

Card	Field	Name	Description
qua1	all	*icard*	Keyword **$QUA**.
qua2	1	*kn*	Number of networks in this data block.
qua3			Same as card **poi3** in **$POI**. **Note**: If card **qua3** calls for other boundary condition cards designated by **bc1** and described earlier, those must be input accordingly.
qua4	1	$sc(1,1)$	x coordinates of the first corner point.
	2	$sc(2,1)$	y coordinates of the first corner point.
	3	$sc(3,1)$	z coordinates of the first corner point.
	4	$sc(1,2)$	x coordinates of the second corner point.
	5	$sc(2,2)$	y coordinates of the second corner point.
	6	$sc(3,2)$	z coordinates of the second corner point.
	71-80	*ntnm*	Name of the network in columns 71-80. The names associated with networks can be used later to process trailing wakes, force partial edge abutments, or associate material properties. This input is read in using (A) format into a character variable that is 10 characters long.
	1	$sc(1,3)$	x coordinates of the third corner point.
	2	$sc(2,3)$	y coordinates of the third corner point.
	3	$sc(3,3)$	z coordinates of the third corner point.
	4	$sc(1,4)$	x coordinates of the fourth corner point.
	5	$sc(2,4)$	y coordinates of the fourth corner point.
	6	$sc(3,4)$	z coordinates of the fourth corner point. **Note**: the corner points of the network are to be ordered as in figure 4.17.
qua5	1	*nrow*	Number of rows of mesh points.
qua6	1-6	$ypc(i)$	Fractional values (0. to 1.) of the row cuts for edges 2 and 4. There are to be *nrow* numbers, provided 6 per line. These values must include 0. and 1.
qua7	1	*ncol*	Number of columns of mesh points.
qua8	1-6	$xpc(i)$	Fractional values (0. to 1.) of the column cuts for edges 1 and 3. There are to be *ncol* numbers, provided 6 per line. These values must include 0. and 1.

4–6.6 Circular Networks ($CIR)

The keyword **$CIR** may be used to generate the network corner points for a circular surface. The details of this type of network are shown in figure 4.18.

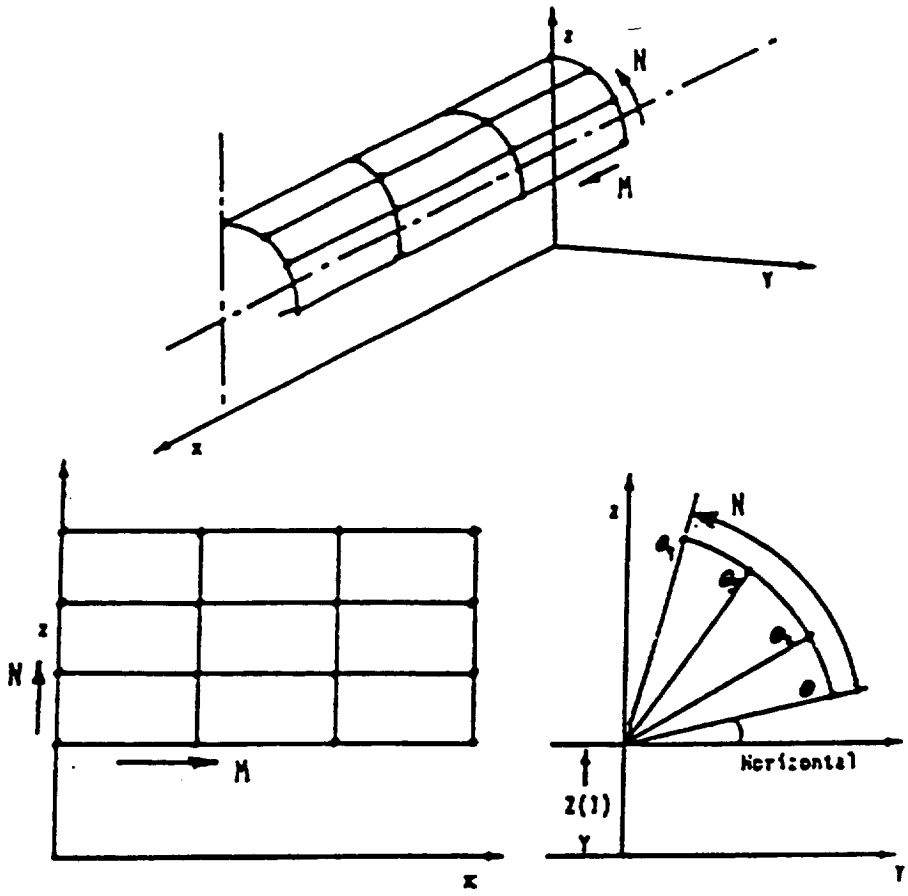


Figure 4.18: Circular Network

Note that it is possible to specify any boundary conditions on this network. The boundary condition data input follows the same lines as described earlier for keyword **$POI** in section 4–6.2. In the following discussion the description of boundary conditions is omitted.

Card Images

$CIRcular Networks								**cir1**
kn	-	-	-	-	-	-	-	**cir2**
! card **cir3** has the same form as described for **poi3**.								
kt	-	-	-	-	-	-	-	**cir3**
zopt	-	-	-	-	-	-	*ntnm*	**cir4**
nm	-	-	-	-	-	-	-	**cir5**
xs(1)	*ri*(1)	*xs*(2)	*ri*(2)	*xs*(3)	*ri*(3)	-	-	**cir6**
xs(4)	*ri*(4)	etc.	-	-	-	-	-	**cir6**
z(1)	*z*(2)	*z*(3)	*z*(4)	etc.	-	-	-	**cir7**
nn	-	-	-	-	-	-	-	**cir8**
th(1)	*th*(2)	etc.	-	-	-	-	-	**cir9**

Input Data Description

Card	Field	Name	Description
cir1	all	*icard*	Keyword **$CIR**.
cir2	1	*kn*	Number of networks in this data block.
cir3			Same as card **poi3** in **$POI**.
cir4	1	*zopt*	Option to control input of a curved centerline. = 0.0 No. = 1.0 Yes. Must include card **cir7**. On card **cir7** it is necessary to input *nm* *z* translations with 6 numbers per card.
	71-80	*ntnm*	Name of the network in columns 71-80. The names associated with networks can be used later to process trailing wakes, force partial edge abutments, or associate material properties. This input is read in using (**A**) format into a character variable that is 10 characters long.
cir5	1	*nm*	Number of *x* stations. On card **cir6** it is necessary to input *nm* pairs with 6 numbers per card.

Card	Field	Name	Description
cir6	1	$xs(i)$	x coordinates of the ith station.
	2	$ri(i)$	Radius at this x station. These pairs of numbers are to be input 6 numbers per line.
cir7	1	$z(i)$	Amount to be added to the z coordinate at ith x station. Input 6 numbers per line, nm numbers in all.
cir8	1	nn	Number of azimuthal angles going around the arc of the circle. There must be nn numbers input on line **cir9**.
cir9	1-6	th(i)	Values of the azimuthal angles input 6 per line.

4–6.7 Rotate, Scale, and/or Translate ($REA)

The TranAir input processor is designed to rotate, scale, and/or translate existing networks. One transform is applied to a specified group of networks at a time. The group is specified by the first and last network numbers. Multiple transformations can be defined by providing multiple **$REA** keywords or by providing for multiple rearrangements within a single **$REA** data block. The transformations will be applied sequentially in the same order as they occur in the input file.

Card Images

$REArrange networks								rea1
nrearr	-	-	-	-	-	-	-	rea2
! Repeat **rea3** through **rea13** as required *nrearr* times.								
ntr	-	-	-	-	-	-	-	rea3
! Rotation about a line. (*ntr*=1., input **rea4** through **rea6**)								
ntnm1	*ntnm2*	-	-	-	-	-	-	rea4
x1	*y1*	*z1*	*x2*	*y2*	*z2*	-	-	rea5
phi	-	-	-	-	-	-	-	rea6
! One to three orthogonal rotations (*ntr*=2.. input **rea7** through **rea9**)								
ntnm1	*ntnm2*	-	-	-	-	-	-	rea7
phx	*phy*	*phz*	*a1*	*a2*	*a3*	-	-	rea8
xo	*yo*	*zo*	-	-	-	-	-	rea9
! Scale (*ntr*=3., input (**rea10** through **rea11**)								
ntnm1	*ntnm2*	-	-	-	-	-	-	rea10
sx	*sy*	*sz*	-	-	-	-	-	rea11
! Translate (*ntr*=4., input **rea12** through **rea13**)								
ntnm1	*ntnm2*	-	-	-	-	-	-	rea12
tx	*ty*	*tz*	-	-	-	-	-	rea13

Input Data Description

Card	Field	Name	Description
rea1	1	*icard*	Keyword **$REA.**
rea2	1	*nrearr*	Number of operations to be performed.
rea3	1	*ntr*	Parameter to select the operation. = 1.0 Rotated about a given line through a given angle. = 2.0 One to three orthogonal rotations about a specified point. The axes system used for the rotations are parallel to the reference coordinate system. = 3.0 Scale x, y, z about the origin. = 4.0 Translate.
rea4 **rea7** **rea10** **rea12**	1-2	*ntnm1*, *ntnm2*	First and last network name or number (defined from **$POI**, **$TRA**, **$QUA** and/or **$CIR**) to be rearranged by.
rea5	1-6	*x1*,*y1*,*z1*, *x2*,*y2*,*z2*	Coordinates of two points defining the axis of rotation.
rea6	1	*phi*	Angle of rotation in degrees. Positive rotation follows the right-hand rule with the direction of the thumb going from point 1 to 2.
rea8	1-3	*phx*,*phy*, *phz*	Angles of rotation in degrees about axes parallel to the x, y, z reference axes and running through the specified point.
	4-6	*a1*,*a2*,*a3*	Parameters to define the first, second, and third rotations. = (1.,2.,3.) Order of rotation *phx*, *phy*, *phz*. = (2.,3.,1.) Order of rotation *phy*, *phz*, *phx*. etc.
rea9	1-3	*xo*,*yo*,*zo*	Coordinates defining the center of rotation (origin) for the orthogonal rotations.
rea11	1-3	*sx*,*sy*,*sz*	x, y, z scale factors applied to the reference coordinates.
rea13	1-3	*tx*,*ty*,*tz*	x, y, z translation values (added to an existing network to produce a modified network).

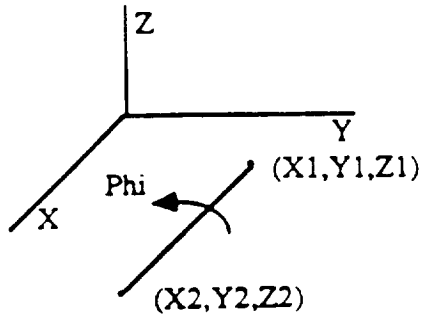


Figure 4.19: Example of Parameters for Rotation About a Line (**$REA**)

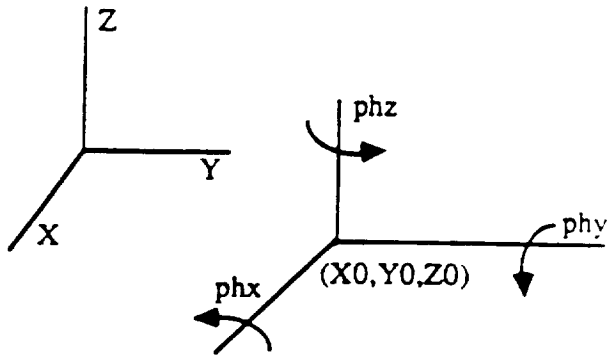


Figure 4.20: Example of Parameters for the Orthogonal Rotations (**$REA**)

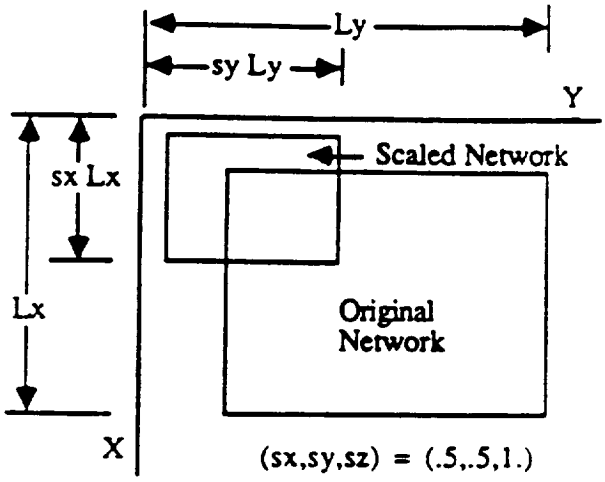


Figure 4.21: Example of Parameters for Scaling About the Origin (**$REA**)

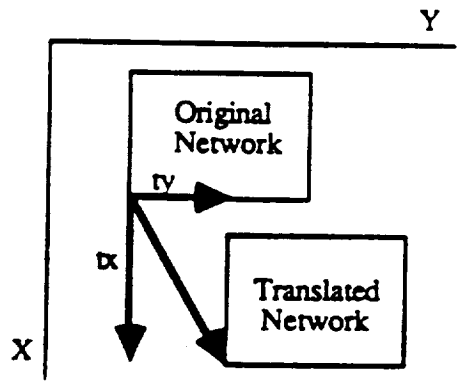


Figure 4.22: Example of Parameters for Translation (**$REA**)

4–7 NETWORK EDGE ABUTMENTS

In TranAir, networks are used to define the configuration boundary, which divides the 3D space into regions with different flow properties. For example, a boundary may separate the interior and exterior of an aircraft. A boundary may define a wake with jump conditions, or it may be used to define plumes where there may be many regions with differing total pressure and temperature. To build the correct set of discrete operators, it is essential that such regions do not communicate with each other (that is, the fluid from one side does not leak to the other side) across the boundary.

After configuration geometry has been described and the proper boundary conditions have been defined, the next important task is to check the network edge abutments. The potential formulation requires that the regions of differing material properties (e.g., surfaces exposed to freestream flow and interior regions of stagnation) may not be connected. Hence, there may be no gaps or leaks at the network edges. The user is responsible for reviewing and accepting the abutments as determined by the program. Thus, the user must determine that all the necessary abutments (relationships between adjacent network edges) have been made and that no extraneous abutments have been formed. Part of this check determines exposed network edges; i.e., edges with no ties to adjacent network edges. For a wake tip side edge, this is acceptable. For a normal wing surface (closed surface), this is not acceptable.

The abutting network edges must have exact panel edge points which match along the network edge, or panel edge points which are on the straight line between the exact points. Thus, the interfaces of two or more network surfaces must match; i.e., have no gaps between adjacent networks. This requirement can be met by ensuring any one or a combination of the following:

- Input geometry has exact matching of every panel edge point along abutting network edges.

- Input geometry nearly matches for every panel edge point along abutting network edges, and the liberalized abutment capability (**$EAT**) makes the abutment identical for points within a single tolerance. Small adjustments are made to the network edge points to make them abut without any gaps and to help eliminate the small round-off error in the input network geometry.

- Input geometry contains some mismatched points along abutting network edges. These edges must be identified for special treatment. Use the partial (or full) network edge abutment capability (**$PEA**) to form a new common edge from matching points along a network. All non-matching points are projected onto the new network edge.

Note that the latter two situations modify the original input geometry along abutting network edges. This modified geometry is used in solving the boundary-value problem and can result in small errors in the resulting surface flow properties. To

reduce network edge distortion, keep non-matching points to a minimum with only small mismatches.

Basic assumptions concerning abutments are listed below. Refer to subsequent paragraphs for details of inputs in each of these categories.

1. All network edge abutments are assumed to start and end at identical (within a small tolerance; for example, *epsgeo* in **$EAT**) network edge points.
2. To maintain the quality of the original input geometry along an abutment, match as many points as possible.
3. Abutments are processed in the following order:
   a. Partial or full network edge abutments (**$PEA**) forced for user-specified network edges.
   b. Liberalized abutments (**$EAT**) for all network edges.
4. The user is responsible for reviewing and validating abutments (abutment summary printout) before making a solution run. The important questions to answer are:
   a. Have all the abutments been made?
   b. Are the appropriate networks abutted?
   c. Are there extraneous abutments?

To assist in verifying abutments, several printouts are of interest:

- The abutment summary printout lists all network edge abutments determined by the program. This list includes the network edges that are unabutted, but do abut the plane of symmetry.
- The abutment intersection summary lists all abutment end points determined by TranAir (optional output).
- The partial network edge printout lists all the details of points moved, along with the amount they have been moved. Also, this printout identifies the network edge equivalent and non-equivalent points along an abutment.
- Also printed is a list of the network edge-point coordinates for any edge which has a point moved by **$EAT**. This list is useful for reviewing the moved points. If no points have been moved by **$EAT**, a message to this effect is provided.

4–7.1 User-Specified Abutments ($PEA)

The **$PEA** data block moves (abuts) network edge points for full or partial network edges that are specified. These edge points are abutted to a common edge. Two to ten network edges may be specified in a single abutment. After **$PEA** is used, no gaps are left in the abutting network edges. The **$PEA** process is described below and is illustrated in figure 4.23.

- Determines full groups of equivalent points; i.e., network edge points found to be identical within a specified tolerance. It contains one point from every network in an abutment.
- Determines all other (non-equivalent) points, network edge points in an abutment which are not identified as equivalent to any other point.
- Defines a common network edge by forming a straight line between adjacent full groups of equivalent points.
- Projects non-equivalent points onto the new common network edge.

TranAir uses the following items as criteria for selecting the final location of a full group of equivalent points:

- Point from the first specified network in the abutment (recommended).
- Average of all points in the group.

When the recommended criterion is used, it is known which edge will remain fixed. Thus, the number of moved network edges is minimized.

If required, a tolerance is available to establish equivalent points for each abutment. The default tolerance is .0001. A tolerance may be specified for each abutment, thus overriding the default. This user-defined tolerance is used for all subsequent abutments until it is overridden by a new specification. By selecting a tolerance value slightly larger than the accuracy of the network geometry (*epsgeo*), the equivalent points can be identified and used to form the new abutted network edge.

In defining a partial network edge abutment, the network name or number, the network edge number, and the first and last edge-point numbers for each abutment edge must be specified. The network number corresponds to the input order of the network. The convention used to define network edge-point numbers is shown in figure 4.24. Observe that, for edges 1 and 2, the edge-point numbers correspond to column and row number conventions used to define a network input. For edges 3 and 4, the edge-point numbers correspond to the reverse order of the column and row conventions.

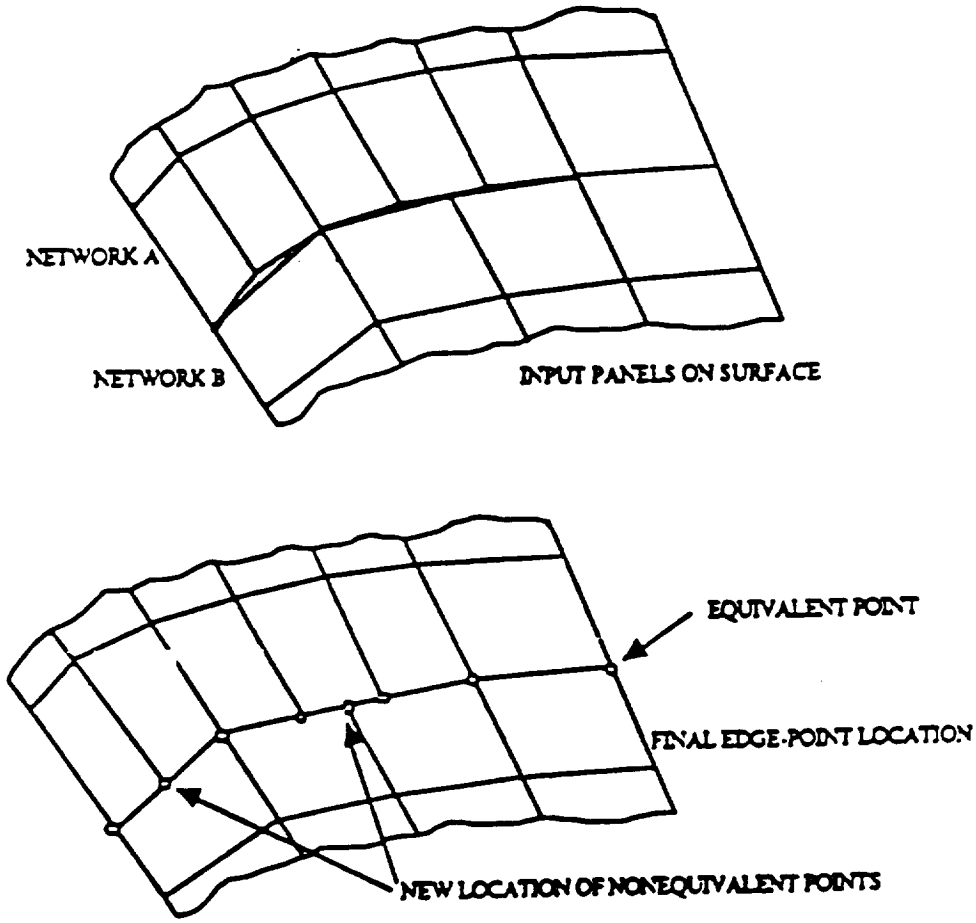


Figure 4.23: Partial Edge Abutment Forcing

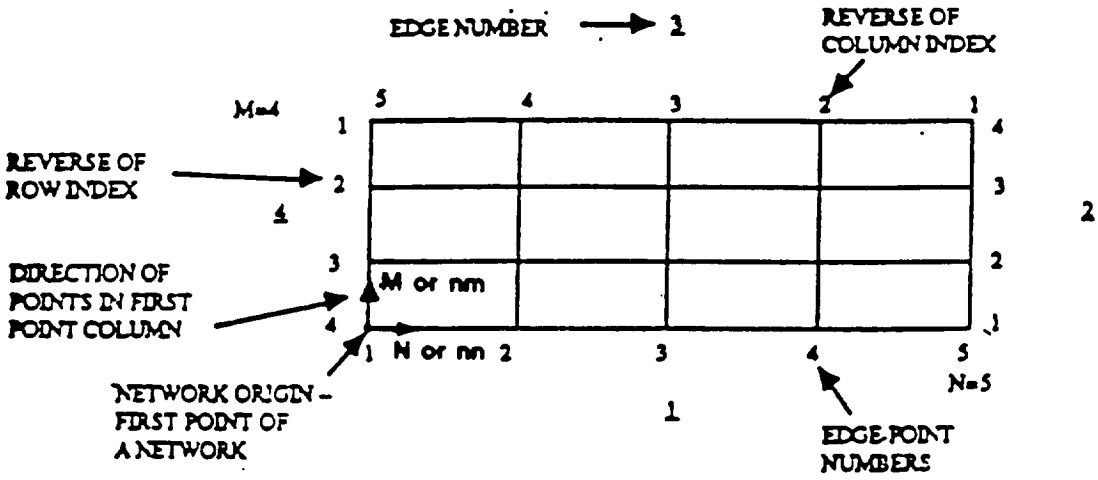


Figure 4.24: Edge and Edge-Point Number Convention

4–7.1

Card Images

$PEA	**pea1**
npa *iopfor* *ipeapt* - - - - -	**pea2**
! Cards **pea3** and **pea4** are to be provided *npa* times	
ne *peatol* - - - - - -	**pea3**
! Card **pea4** is to be input *ne* times	
ntnm *en* *epi* *epl* - - - -	**pea4**

Input Data Description

Card	Field	Name	Description
pea1	all	*icard*	Keyword **$PEA**.
pea2	1	*npa*	Number of forced partial network edge abutments specified; must be less than 500.
	2	*iopfor*	Parameter to select the equivalency option. = 0.0 First specified network becomes the abutment and all the other network edges are moved to this abutment. = 1.0 Abutment is found by averaging the selected equivalent points. Equivalent points are those points which lie within a tolerance distance of each other.
	3	*ipeapt*	Control for printout from the forced partial network edge procedure. = 0.0 Print out points before and after **$PEA** processing. = 2.0 Additional levels of diagnostic print. = 3.0 Yet more levels of diagnostic print.
pea3	1	*ne*	Number of network edges specified in current abutment; must be less than 20.
	2	*peatol*	Tolerance distance used to establish equivalent points of the current abutment. This number should be bigger than the biggest gap but smaller than the smallest edge of the smallest panel. The default *peatol* = 0.0001

Card	Field	Name	Description
pea4	1	*ntnm*	Network name or number.
	2	*en*	Edge number. (See figure 4.24.) **Note**: To specify a full network edge, omit the next two parameters.
	3	*epi*	First network edge-point number in abutment; first and last edge-point numbers are interchangeable (see figure 4.24). The order of points do not need to be consistent between the various networks involved.
	4	*epl*	Last network edge-point number in abutment. The order of points does not need to be consistent between the various networks involved.

4-7.2 Program-Determined Abutments ($EAT)

The liberalized abutment program function (**$EAT**) establishes network edge relationships. It determines full or partial abutment within a specified tolerance. The procedure used is similar to that for **$PEA**. However, **$PEA** is applied only to specified network edges and, in general, has a larger tolerance for determining equivalent points. **$EAT** geometrically averages full equivalence groups of points and projects other points onto the common abutment edge. It moves network edge points to eliminate mismatches between network edges. While comparing network edge points, **$EAT** also establishes the matching information on network edge points required to solve the potential flow problem.

$EAT examines network edge points to find groups of equivalent edge points (identical points within a tolerance). Points which are equivalent but different are all changed to the average value of the equivalent points. With the liberalized abutment capability, TranAir does not require network edge abutments to be exact to the last significant figure input. **$EAT** allows calculation by the program or by user input of the tolerance used to establish equivalent points. Program-selected default tolerance is .001 times the minimum panel diagonal of all panels. The maximum user-specified tolerance is .03 times the minimum panel diagonal of all panels.

If the tolerance is too large, it is reduced to the maximum allowable value. Too large a tolerance can cause inappropriate points to be equal and bring together unwanted network edge points. It is the user's responsibility to check points moved by **$EAT** for unwanted equivalent points. The liberalized abutment capability for moving network edge points cannot be turned off.

To eliminate small mismatches in equivalent network edge points, specify a tolerance equal to the accuracy of the network geometry. If the network geometry is accurate to two significant figures to the right of the decimal point, set the tolerance equal to .01. This allows abutments to be met even when the network geometry contains a small error.

Caution: If the tolerance is large, unwanted abutments may occur. Always check the summary of moved points.

To prevent **$EAT** from moving points if the network edge geometry is identical, specify a tolerance smaller than the number of significant figures of the input geometry. For example, if geometry is defined using four significant figures to the right of the decimal point, set the tolerance to .00009.

In addition to the input tolerance, **$EAT** contains other options for detailed printout. Normally used options are defaulted. The following are the **$EAT** options and details that are provided in the input section:

- Network-by-network cross-reference printed for abutment and abutment intersections.
- Control of abutment printout.

Card Images

$EAT (Edge Abutment Tolerance)							eat1
epsgeo	-	-	*nwxref*	-	*iabsum*	-	-	eat2

Input Data Description

Card	Field	Name	Description
eat1	all	*icard*	Keyword **$EAT**.
eat2	1	*epsgeo*	Radial distance used to establish equivalent network edge points. Range of acceptable values $d_{min} * 0.001 \leq epsgeo \leq d_{min} * 0.03$, where d_{min} is the minimum panel diagonal. If the input is out of range, the value is reset to the upper or lower limiting value. To override the program limit, enter *epsgeo* as a negative number. =0; assumes $epsgeo = d_{min} * 0.001$.
	4	*nwxref*	Control to print the abutment cross reference; for each network, all abutments and abutment intersections are described. = 0.0 No. = 1.0 Yes.
	6	*iabsum*	Prints abutment data. = 0.0 Abutment summary. = 1.0 *iabsum*=0.0 plus intersection summary. = 2.0 *iabsum*=1.0 plus complete list of each network and the associated abutments (currently unavailable). = 3.0 *iabsum*=2.0 plus diagnostic data for the program developer. = -1.0 No abutment printout.

Default Values

All keywords in this category may be omitted from the input stream. If omitted, the program chooses a tolerance based on a fraction of the minimum panel diagonal. The default values for the data described in this data block are given below.

Parameter	Default Value
epsgeo	Minimum panel diagonal times 0.001
nwxref	0.
iabsum	0.

4–8 OPTIONAL OUTPUT

All of the optional computations based on a TranAir solution are covered in this section. These involve printout, integration of surface properties, relofting of plume surfaces, and output data files. Almost all of these options supply input to the output module (`fdout`) of TranAir (for the exception, see **$PRI**nt). Thus, they can be run with a complete solution (as part of the input to the `fdinp` program) or as a restart from a solution (as input to the `fdout` program) when restarting for postprocessing. If run in the latter mode, the input data must be terminated by a **$END** line of input.

4–8.1 Printout ($PRI)

The printout control parameters are specified via the **$PRI** keyword. The portions of printed output relate to the analysis of the input data, abutments, solver output, and finally, the resulting surface flow properties and the surface forces and moments.

Card Images

$PRIntout Control	**pri1**
iprpan - - - - - - -	**pri2**
iprmus *iprcen* *iprcrn* *iprfm0* - - - -	**pri3**

Input Data Description

Card	Field	Name	Description
pri1	all	*icard*	**$PRI** Keyword.
pri2	1	*iprpan*	Panel geometry output control. The input geometry provided to the program can be printed. The program may also change some corner points to either force abutments or to meet the liberalized tolerances. The panel corner point data may be printed before and/or after it has been adjusted by the program. Recommended value is 0. = 0.0 Does not print. = 1.0 Prints only before the adjustment. = 2.0 Prints only after the adjustment. = 3.0 Prints both before and after the adjustment.

Card	Field	Name	Description
pri3	1	*iprmus*	Printout control for wake strengths in the output processor. = 0.0 Does not print. = 1.0 Prints.
	2	*iprcen*	Printout control for printing aerodynamic properties at panel center points in the output processor. = 0.0 Does not print. = 1.0 Prints output for 13 aerodynamic properties. = 2.0 Prints output for 49 aerodynamic properties.
	3	*iprcrn*	Printout control for printing aerodynamic properties at corner points in the output processor. = 0.0 Does not print. = 1.0 Prints 15 parameters per corner point including the location, the level of the nearest box, the grid box number, the velocity, pressure coefficient, Mach number, potential, and the normal mass flux.
	4	*iprfm0*	Printout control for network forces and moments. = 0.0 Does not print. = 1.0 Prints data for the network and all the accumulated networks considered until that point. = 2.0 Prints data for the columns, the network and all the accumulated networks considered until that point.

Default Values

All keywords in this category may be omitted from the input stream. The default values for the data described in this data block are given below.

Parameter	Default Value
iprpan	0.
iprmus	1.
iprcen	0.
iprcrn	1.
iprfm0	1.

4–8.2 Surface Forces and Moments ($REF)

Reference area and lengths used to non-dimensionalize the configuration forces and moments are specified in the data block below via the keyword **$REF**. The printout control of this data is in **$PRI**nt (*iprfm*0, see section 4–7.1).

Card Images

$REFerence Lengths and Areas								ref1
xref	*yref*	*zref*	-	-	-	-	-	ref2
sref	*bref*	*cref*	*dref*	-	-	-	-	ref3

Input Data Description

Card	Field	Name	Description
ref1	all	*icard*	Keyword **$REF**erence Quantities.
ref2	1	*xref*	Moment reference coordinates in x direction.
	2	*yref*	Moment reference coordinates in y direction.
	3	*zref*	Moment reference coordinates in z direction.
ref3	1	*sref*	Reference area.
	2	*bref*	Reference length for rolling moment.
	3	*cref*	Reference length for pitching moment.
	4	*dref*	Reference length for yawing moment.

Use the full wing reference area for the *reference area* defined for an airplane with one plane of symmetry, along with inputs defining half an airplane. With this reference area, the output forces and moments represent the input network surfaces and the accumulated totals for these surfaces.

Default Values

All keywords in this category may be omitted from the input stream. The default values for the data described in this data block are given below.

Parameter	Default Value
xref	0.0
yref	0.0
zref	0.0
sref	1.0
bref	1.0
cref	1.0
dref	1.0

Forces and Moments Formulas

The 3-D forces (FX, FY, FZ) and moments (MX, MY, MZ) are calculated by integration of the pressure coefficient over the input non-wake network surfaces. The force per panel is represented by the following equation:

$$\vec{F} = \frac{-1}{sref} \int_{panel} C_p \, \hat{n} \, dA \tag{4.49}$$

where:

$\vec{F}$	=	(FX, FY, FZ)
$sref$	=	reference area
$\hat{n}$	=	unit normal vector to surface
A	=	panel area
C_p	=	isentropic pressure coefficient

The moment per panel is:

$$\vec{M} = \frac{-1}{sref \; L_r} \int_{panel} C_p(\vec{P} - \vec{R}_o) \times \; \hat{n} \, dA \tag{4.50}$$

where:

$\vec{M}$	=	(MX, MY, MZ)
L_r	=	reference lengths for moment components as given below:
$bref$	=	reference length for MX (span)
$cref$	=	reference length for MY (chord)
$dref$	=	reference length for MZ (fuselage length)
$\vec{P}$	=	point of integration on panel
$\vec{R}_o$	=	$(xref, yref, zref)$, moment reference location

Output

The 3D forces (FX, FY, FZ) and moments (MX, MY, MZ) are calculated by integration of the *pressure coefficient* over the *input non-wake network surfaces.* Pressure distribution varies linearly over each panel. Three lines of printout of forces and moments correspond to the integration over:

- Upper surface (surface of predominant interest).
- Lower surface.
- Upper surface plus lower surface (thin surface representation).

The forces and moments are calculated for the following:

- Each panel column (optional; see *iprfm*0 in **$PRINT**, section 4-8.1).

- Each network.
- All previous networks (accumulated values used to form totals).

4–8.3 Configuration Forces and Moments ($FOR)

The configuration forces and moments summary gives the lift (CL), induced drag (CD), side force (CY), and forces and moments about the reference coordinate system (FX, FY, FZ, MX, MY, MZ) for both the inputs and the complete configuration. Results are based on the appropriate summation of the previously mentioned 3D network surface forces and moments.

To assemble a configuration forces and moments summary, the program assumes a configuration defined from all upper surfaces of the impermeable and thin surface networks (kt=1 and kt=2). In general, body bases, fan faces, wakes, etc. are not included in the total configuration forces because they are fictitious surfaces to the real flow. The associated data block is used to specify network surfaces and pressure surface (upper, lower, or difference) used to define the configurations that are *different* from the assumed surfaces.

If the inputs to the configuration forces and moments summary are not supplied, the program outputs a forces-and-moments summary using the upper surface pressure coefficient integrated over a configuration defined by all networks for which kt=1 and kt=2.

The inputs required to build this group of networks is provided via the keyword **$FOR**.

Card Images

$FORces and Moments	for1
! If all impermeable and thin surface networks (kt=1 and kt=2) are desired for	
! this summary, omit lines **for2** through **for4**.	
***NET**work Selection for summary	for2
nnt *idflt* - - - - - -	for3
! Card **for4** to be repeated *nnt* times	
ntnm - - - - - - -	for4
! If all upper surface pressure coefficients are used in the summary, omit lines	
! **for5** through **for7**.	
***PRE**ssure surface selection for the summary	for5
nmsc - - - - - - -	for6
! Card **for7** to be repeated *nmsc* times	
ntnm *sfcd* - - - - - -	for7

Input Data Description

Card	Field	Name	Description
for1	all	*icard*	Keyword **$FOR**.
for2	all	*icard*	Keyword for subblock ***NET** If only all kt=1 networks are desired, omit lines **for2** through **for4**.
for3	1	*nnt*	Number of networks to be added or deleted. Must repeat card **for4** *nnt* times.
	2	*idflt*	Value defining default base of assumed networks: = 0.0 All kt=1 and 2 networks (omit card **for4**). = 1.0 All non-wake networks. = 2.0 No networks in base.
for4	1	*ntnm*	Network name or number defining networks used to modify or define a baseline set of networks. A positive number adds the network to the set. A negative number deletes the network from the set.
for5	all	*icard*	Keyword ***PRE**. If all upper surface pressure coefficients are to be used in the summary, omit lines **for5** through **for7**.
for6	1	*nmsc*	Number of networks where the pressure side is to be specified. Must repeat card **for7** *nmsc* times.
for7	1	*ntnm*	Network name or number of network where the pressure side is to be specified.
	2	*sfcd*	Pressure surface code. = 1.0 Upper surface. = 2.0 Lower surface. = 3.0 Difference (upper minus lower).

Default Values

All the keywords in this category may be omitted from the input. In that case, the program will provide a forces and moments summary based on all kt=1 and kt=2 networks.

Output

Forces and moments on the full configuration are computed on the input configuration plus the reflected images (planes of symmetry). The planes of antisymmetry do not change the definition of the full configuration.

The lift, induced drag, and side force are defined from:

$$
\begin{aligned}
CL &= -FX\sin(\alpha)\cos(\beta) + FY\sin(\alpha)\sin(\beta) + FZ\cos(\alpha) \\
CD &= FX\cos(\alpha)\cos(\beta) - FY\cos(\alpha)\sin(\beta) + FZ\sin(\alpha) \qquad (4.51) \\
CY &= +FX\sin(\beta) + FY\cos(\beta)
\end{aligned}
$$

(4.52)

where:

FX, FY, FZ = force coefficients along reference coordinate axes.

α, β = angles of attack and sideslip, respectively.

4–8.4 Sectional Properties ($SEC)

Sectional properties are the solution on the surface of the configuration at some other points other than the panel corner points which define the geometry. In TRANARI sectional properties are defined by computing the intersection of a cutting plane with the configuration. The solution on the lines of intersection is interpolated from the solution defined on the configuration surface. Since the network layout and the order in which they are input is arbitrary, it is difficult to provide the sectional properties unless the columns of panels match the sectional attributes. In the case of a general configuration where such matching is difficult, it is necessary to interpolate data at points of intersection from the surrounding panel data. The data is interpolated from a predetermined set of networks. By default, this set includes all networks with kt=1 boundary conditions. This set can be changed through input specifications. The force calculations which involve pressure at points can also be based on either the upper surface pressure, the lower surface pressures, or the difference between the upper and lower surface pressures. This choice can also be made through inputs.

A typical cut plane used is illustrated in Figure 4.25.

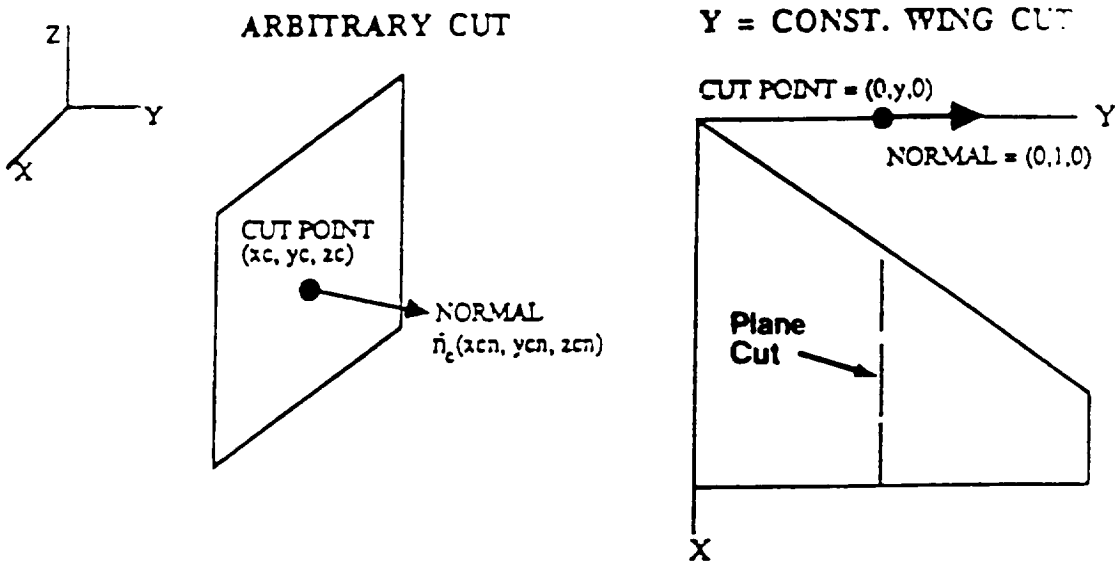
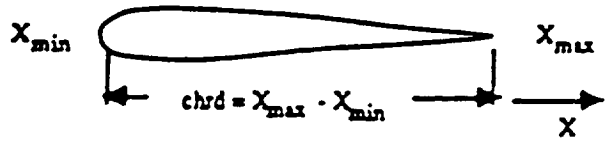


Figure 4.25: Sectional Cut Plane

The sectional chord and moment reference location can be determined by the program or specified directly. The program determines the chord from the minimum and maximum x, y, or z coordinates as a distance between the end points of the specified cut. The quantities specified through the keyword **$REF** can also be used as reference quantities for the sectional moment reference. These options are summarized

optcrd = 0.



optcrd = 1.

MAXIMUM DISTANCE
BETWEEN TWO POINTS
IN THE CUT

optcrd = 2.

Input chord value (chrd) for each cut.

Figure 4.26: Sectional Property Input Options for Chord Definition

in figures 4.26 and 4.27

Input data for the sectional properties output can be provided via the **$SEC** keyword.

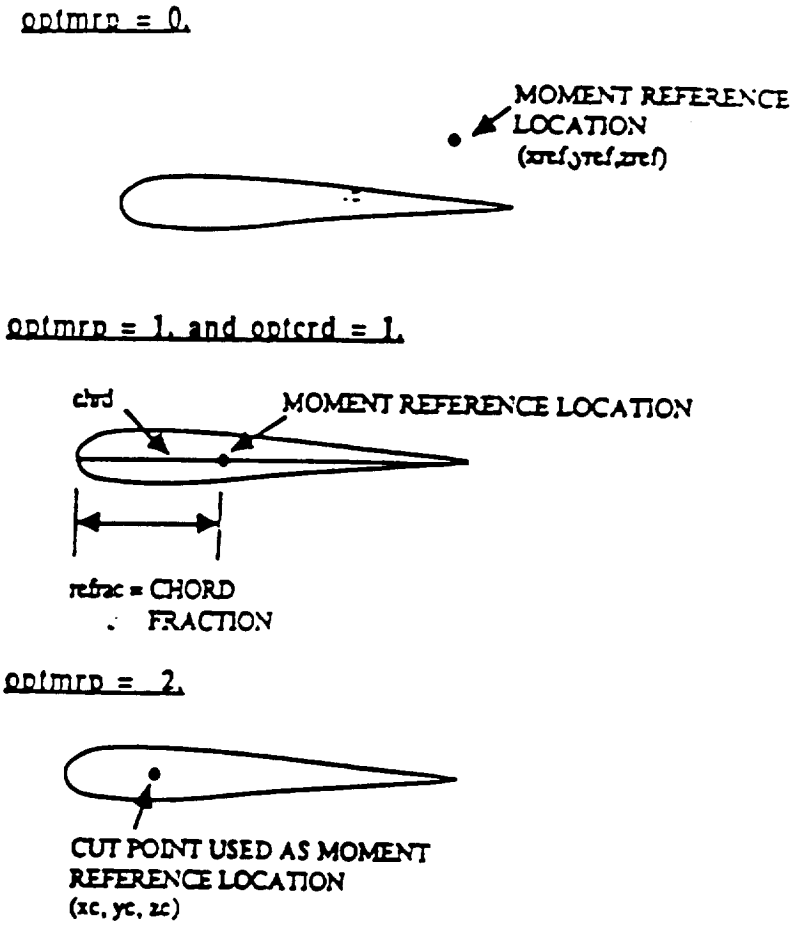


Figure 4.27: Sectional Property Input Options for Moment Reference Point

Card Images

$SECtional Properties								sec1
ngrp	-	-	-	-	-	-	-	sec2
*NETWORK Selection								sec3
nnt	-	-	-	-	-	-	-	sec4
! Card **sec5** to be repeated *nnt* times								
ntnm(1)	-	-	-	-	-	-	-	sec5
*PREssure Surface								sec6
ns	-	-	-	-	-	-	-	sec7
! Card **sec8** to be repeated *ns* times								
ntnm	*sfcd*	-	-	-	-	-	-	sec8
*CUT								sec7
optcrd	*optmrp*	*iprtnf*	*iprtpp*	*isecpr*	*ixyzop*	*reflen*	-	sec10
nc	-	-	-	-	-	-	-	sec11
! Card **sec12** to be repeated *nc* times								
xc	*yc*	*zc*	*xcn*	*ycn*	*zcn*	*len*	-	sec12
! If *optcrd*=2.0 and/or *optmrp*=1.0,								
! Replace **sec12** with the following inputs								
xc	*yc*	*zc*	*xcn*	*ycn*	*zcn*	-	-	sec12
chrd	*rfrc*	-	-	-	-	-	-	sec13

Input Data Description

Card	Field	Name	Description
sec1	all	*icard*	Keyword **$SEC**.
sec2	1	*ngrp*	Number of sectional cut groups (1. to 5.). Cards **sec3** through **sec13** must be repeated *ngrp* times.
sec3	all	*icard*	Keyword for subblock ***NET**. **Note**: If all non-wake networks are to be used in this cut group, cards **sec3** through **sec5** can be omitted.
sec4	1	*nnt*	Number of networks input on **sec5** group. If *nnt* is positive, only those networks mentioned in **sec5** will be included. If *nnt* is negative, all networks (other than $kt = 18$ and $kt = 20$ networks) will be included except for those mentioned in **sec5** input.
sec5	1	*ntnm*(1)	Network names or numbers to be included ($nnt > 0$) or excluded ($nnt < 0$) from a set of all networks. (Note that $kt = 18$ or $kt = 20$ are always excluded.)
sec6	all	*icard*	Keyword for subblock ***PRE**. **Note**: If all upper surface pressure coefficients are to be used in this cut group, cards **sec6** through **sec8** may be omitted.
sec7	1	*ns*	Number of networks for which pressure surface is specified as being different from the upper surface.

Card	Field	Name	Description
sec8	1	*ntdt*	Network number with pressure surface different from default
	2	*sfcd*	Pressure surface code. = 1.0 Upper surface. = 2.0 Lower surface. = 3.0 Difference (upper minus lower).
sec9	all	*icard*	Keyword ***CUT.**
sec10	1	*optcrd*	Chord definition: = 0.0 Maximum minus minimum x, y, or z of a cut (see definition of "*ixyzop*" below). = 1.0 Maximum distance between two points of a cut. = 2.0 Chord for each cut is to be input on card **sec13.**
	2	*optmrp*	Moment reference definition: = 0.0 3D moment reference point ($xref$, $yref$, $zref$) will be used. = 1.0 Specifies chord fraction on card **sec13** for each cut along chord. = 2.0 Uses point defining cut plane (xc, yc, zc) on card **sec12.**
	3	*iprtnf*	Option to print sectional properties for each network. = 0.0 No. = 1.0 Yes.
	4	*iprtpp*	Option to print panel pressures along a cut. = 0.0 No. = 1.0 Yes.
	5	*isecpr*	Uses a diagnostic printout to check cut traces, integration across a network, and pressure coefficients on each panel. = 0.0 No (recommended). = 1.0 Yes.
	6	*ixyzop*	Option to select the preferred direction for defining chord. = 1.0 x direction. = 2.0 y direction. = 3.0 z direction.
	7	*reflen*	Reference length for non-dimensionalizing the sectional property location. Default is 1 (see *len* below).

Card	Field	Name	Description
sec11	1	*nc*	Number of cuts.
sec12	1	*xc*	x coordinates of point in cut plane.
	2	*yc*	y coordinates of point in cut plane.
	3	*zc*	z coordinates of point in cut plane.
	4	*xcn*	x component of normal vector perpendicular to cut plane.
	5	*ycn*	y component of normal vector perpendicular to cut plane.
	6	*zcn*	z component of normal vector perpendicular to cut plane.
	7	*len*	Length associated with the sectional cut (if no input is provided, *yc* is assumed). Also ($eta = len/reflen$).
sec13	1	*chrd*	Reference chord for each cut.
	2	*refrac*	Chord fraction for each cut used to define the moment reference location.

4–8.5 Surface Properties File ($SUR)

The keyword **$SUR** is used to control surface flow properties and networks placed on the surface solution file (under UNICOS, fort.10). The input options contain several standard sets of surface flow properties. It also allows for specifying any customized set of surface flow properties.

Card Images

```
$SURface properties file                                              sur1
ipggp  -  -  -  -  -  -  -                                            sur2
! If output is over all networks and includes both upper and lower
! surfaces. omit lines sur3 through sur4.
*NETwork                                                              sur3
! As needed, repeat line sur4.
netbeg  netend  isurf  -  -  -  -  -                                  sur4
! Input *PAR only if ipggp=4.0
*PARameter, flow properties                                           sur5
iparam1  iparam2  iparam3  iparam4  iparam5  iparram6  -  -           sur6
iparam7  ...                                                          sur6
```

Input Data Description

Card	Field	Name	Description
sur1	all	*icard*	Keyword **$SUR**.
sur2	1	*ipggp*	Selection of surface flow properties to be placed on $CASE.ggp. Standard properties output for first four options: = 0.0 x, y, z, Cp, Mach = 1.0 x, y, z, Cp, Mach, u, v, w = 2.0 x, y, z, Cp, Mach, u, v, w, wn, rho material index = 3.0 x, y, z, Cpa, Cpd, Macha, Machd, wna, wnd, phia, phid User selected properties = 4.0 (see ***PAR**ameters)

Card	Field	Name	Description
sur3	all	*icard*	Keyword ***NET**.
sur4	1	*netbeg*	Network name or number to be included in output.
	2	*netend*	Network name or number. All networks between the network specified in columns 1-10 and this network (inclusive) will be added to the list. If *netend* is zero or blank, only *netbeg* is added to the list.
	3	*isurf*	Surfaces to be printed out. This is a toggle: once set, it stays the same until changed by the user. 0, 1, both — Default, except for material prop 2 (stagnation) or *ipggp*=3. 2, upper — Upper surface only. 3, lower — Lower surface only.
sur5	all	*icard*	Keyword ***PAR**.

Card	Field	Name	Description
sur6	1-6	*iparam*	Surface flow parameter names selected from the following list: CDP — Drag contribution CDPA — Drag contribution average CDPD — Drag contribution difference CLP — Lift contribution CLPA — Lift contribution average CLPD — Lift contribution difference CP — Pressure coefficient CPA — Average pressure coefficient CPD — Pressure coefficient difference MACH — Mach number MACHA — Average Mach number MACHD — Mach number difference MATERIAL — Material property index P3DQ — PLOT3D reference flow properties (Q): Density (P3DQ1) X-momentum (P3DQ2) Y-momentum (P3DQ3) Z-momentum (P3DQ4) Stagnation energy per unit volume Specification of "P3DQ" generates P3DQ1 through P3DQ5. The individual components *cannot* be specified. PHI — Potential PHIA — Average potential PHID — Potential difference RHO — Density U — Local X-velocity V — Local Y-velocity W — Local Z-velocity WN — Normal component of mass flux WNA — Normal component of mass flux average WND — Normal component of mass flux difference ZNX — X-component of normal vector ZNY — Y-component of normal vector ZNZ — Z-component of normal vector

Default

If this data block is omitted, the pressure and Mach numbers over all networks on both upper and lower surfaces (excluding surfaces exposed to stagnation regions) will be output. The default values assumed are:

Parameter	Value
ipggp	0.

4–8.6 Field Properties File ($FIE)

The keyword **$FIE** is used to control field flow properties placed on the field solution plotting files (under UNICOS, fort.2 and fort.4 for Cray and Iris binary files, respectively).

Caution: The output data file can be very large for problems with a large field grid. Thus, it becomes important to limit the data on the file to the data essential for the problem.

Note, currently a field solution plotting file cannot be run as a restart. Thus, only the properties requested for the initial solution will be placed on the output file.

Card Images

$FIEld properties file								**fie1**
npbnp	*npbnm*	*npbnr*	*npbnv*	*npbne*	*npbnq*	-	-	**fie2**

Input Data Description

Card	Field	Name	Description
fie1	all	*icard*	Keyword **$FIE**.
fie2	1	*npbnp*	Pressure coefficient (Cp) at the centroids of the cells in the file processed by the TranAir Graphics Utility. = 0.0 Not included. = 1.0 Included.
	2	*npbnm*	Mach number (Mach) at the centroids of the cells in the file processed by the TranAir Graphics Utility. = 0.0 Not included. = 1.0 Included.
	3	*npbnr*	Density (ρ) at the centroids of the cells in the file processed by the TranAir Graphics Utility. = 0.0 Not included. = 1.0 Included.
	4	*npbnv*	Velocity components (u, v, w) at the centroids of the cells in the file processed by the TranAir Graphics Utility. = 0.0 Not included. = 1.0 x component of velocity (u) included. = 2.0 y component of velocity (v) included. = 3.0 z component of velocity (w) included. = 4.0 All three components of velocity (u, v, w) included.

Card	Field	Name	Description
fie2	5	*npbne*	Nonlinear scaled error predictor at the centroids of the cells in the file processed by the TranAir Graphics Utility. = 0.0 Not included. = 1.0 Included. **Note**: These estimates are scaled by the *adpfac* values specified by the user with the **$TOL**. **$SBO** and **$LBO** keywords. Thus, *adpfac* > 0 produces the error predictor output. On the final grid in an adaptive run, the largest estimates indicate regions where additional grid refinement would occur if another grid were constructed.
	6	*npbnq*	PLOT3D reference flow properties (Q): Density (P3DQ1) X-momentum (P3DQ2) Y-momentum (P3DQ3) Z-momentum (P3DQ4) Stagnation energy per unit volume (P3DQ5) = 0.0 Not included. = 1.0 Included.

Default Values

If this data block is omitted, the pressure, Mach number and density properties will be output at the field grid location. The default values assumed are:

Parameter	Default
npbnp	1.0
npbnm	1.0
npbnr	1.0
npbnv	0.
npbne	0.

5– PROGRAM OUTPUT

As it executes, each module in the TranAir system of programs writes printed information on FORTRAN unit 6. This information describes how TranAir has interpreted the user input, how the solution process has progressed and describes various properties about the flow field in the vicinity of the configuration surface, including global properties like configuration forces and moments. This chapter describes in a general manner how to interpret this information. The printout is grouped according to the module in the code that generates the output. The major modules in the code are the **Input Processor**, the **Solver**, and the **Output processor**.

In addition to these files, TranAir also creates output files (both binary and ASCII) which are intended for use with various graphical plotting programs. (Appendix B describes TGRAF, a graphical plotting program for viewing field and surface solution data generated by TranAir.)

5–1 INPUT PROCESSOR PRINTOUT

The output of the **Input Processor** begins with a header defining the version number and release date of the code, followed by a heading containing the information provided by the user through the **$TIT** keyword. A listing of all input data provided by the user is printed next. This list summarizes the configuration definition on a network basis, describes the properties of the global grid, and describes the connectivity of the networks to one another with a summary of abutment information.

The output begins with a heading of the form:

```
- LIST OF TranAir INPUT DATA CARDS -NO. CARD IMAGES
```

After this appears a direct listing of the user's input terminating with the ending keyword:

```
$END
```

The **Input Processor** then prints each keyword it encounters as it processes them. In the course of doing this processing, warning or error messages indicating questionable or erroneous user input information may be printed. These warnings

and error messages are self-explanatory in the context in which they appear. This portion of the output terminates with the **$END** terminating keyword.

When first preparing input files for TranAir, it is common that errors will occur in the input file. In some cases, these errors may cause the **Input Processor** to terminate in an abnormal way with a call to the system ABORT routine. Should this occur, the list of keywords that have been processed provides a clue as to the next keyword that will be processed and where the error occurs in the user input file.

The **Input Processor** next prints a summary of the global grid attributes and a Quick Summary of the configuration on a network by network basis. If the user has specified a computational box, its properties are printed here. If the user has requested that the code assign a computational box based on the configuration geometry, then at this point the user is informed as to the extent and grid density of that assigned computational box.

The Quick Summary provides a list of all network names, with associated network numbers. This printout section lists all input options, along with their user-assigned values. A sample of the printout is shown in Figure 5.1:

Following this section of printout, if the user has provided forced partial edge abutments, the results obtained from processing them are printed. Each forced abutment is numbered and treated separately. Detailed information is provided including: changes to equivalent points (points within tolerance), changes to nonequivalent points, and summary of points matched and not matched along the abutting edge. If forced partial edge abutments are used, the user should review the data in this section to ensure that the specified abutments are properly processed.

Following this output, a summary of abutment information is printed. This information contains an abutment intersection summary, an "extra control point" summary (if any exist in the configuration), and an abutment summary. The abutment intersection summary and extra control point summary are provided for compatibility with some previous analysis codes and do not have particular relevance for TranAir users outside of the Boeing environment.

The Abutment Summary printout provides a complete list of all network edge abutments found by TranAir. Therefore, it is of primary interest for checking abutments. The user should carefully analyze the description of the Abutment Summary to assure that the configuration has been properly represented. The Abutment Summary contains:

- Printout of abutment number, network number, edge number, corresponding network edge-point numbers, etc., associated with each abutment.
- Minus sign (-) on network edge-point numbers identifying points which have been moved within the tolerance specified by the keyword, **$EAT**.
- Emphasis on particular abutment messages output for any network edge which does not abut another edge, or a plane of symmetry or antisymmetry. The messages are of the following types:

```
                *** QUICK SUMMARY OF TRANAIR INPUT ***
TITLE: THIN WING CASE --- SIMULATION OF RDL GRID ALONE CODE
 NAME: F. T. JOHNSON

                CASE SUMMARY
            1 = NUMBER OF CASES
     0.000100 = MACH NUMBER
     0.000000 = COMPRESSIBILITY AXIS ANGLE OF ATTACK (ALPC)
     0.000000 = COMPRESSIBILITY AXIS ANGLE OF YAW (BETC)

  CASE       ALPHA          BETA       MAG(F-S-V)
------    ----------    ----------    ----------
     1      5.000000      0.000000      1.000000

                SYMMETRY OPTIONS
            1 = NUMBER OF PLANES OF SYMMETRY
            1 = X-Z PLANE OF SYMMETRY FLAG (0 ==> NO SYMMETRY, 1==> FLOW SYMMETRY, -1 ==> FLOW ANTISYMMETRY)
            0 = X-Y PLANE OF SYMMETRY FLAG (0 ==> NO SYMMETRY, 1==> FLOW SYMMETRY, -1 ==> FLOW ANTISYMMETRY)

                CONFIGURATION SUMMARY
            2 = TOTAL NUMBER OF NETWORKS READ IN
           10 = TOTAL NUMBER OF MESH POINTS
            3 = TOTAL NUMBER OF PANELS

       NETWORK ID          INDEX    #ROWS    #COLS   SOURCE   DOUBLT  NLPT1 NRPT1 NLPT2 NRPT2  # PTS  # PANS  CPNORM   MAT1  MAT2
------------------------- -----    -----    -----   ------   ------  ----- ----- ----- -----  -----  ------  ------  ----- -----
                              1        2        2        0       12      5     2     6     2      4       1       1      1     1
                              2        3        2        0       18     16     2     8     2      6       2       1      1     1

                ABUTMENT PROCESSING OPTIONS
  0.00000E+00 = GLOBAL EDGE ABUTMENT TOLERANCE SPECIFIED BY USER.  IF THIS VALUE IS ZERO, A DEFAULT VALUE WILL BE CALCULATED
                LATER.   THIS DEFAULT VALUE IS TAKEN AS:  .001  * (MINIMUM PANEL DIAMETER)
            1 = PRINT FLAG CONTROLLING GEOMETRY PRINTOUT  B E F O R E  THE ABUTMENT PROCESSING.  ( NONZERO ==> DO PRINT )
            0 = PRINT FLAG CONTROLLING GEOMETRY PRINTOUT  A F T E R    THE ABUTMENT PROCESSING.  ( NONZERO ==> DO PRINT )
            0 = NETWORK/ABUTMENT/ABUTMENT-INTERSECTION PRINT FLAG.  ( NONZERO ==> GENERATE THE CROSS REFERENCED ABUTMENT LISTING
            0 = CONTROL INDEX FOR PANEL INTERSECTION CHECKING.  ( NONZERO ==> DO PERFORM THE CHECK. )
            1 = ABUTMENT/ABUTMENT-INTERSECTION (SHORT LISTING) PRINT FLAG ( 0 ==> SUPPRESS, NONZERO ==> GENERATE USUAL PRINT )

                FORCE AND MOMENT REFERENCE PARAMETERS
  1.00000E+00 = REFERENCE AREA FOR FORCE AND MOMENT CALCULATIONS.    (SREF)
  1.00000E+00 = ROLLING MOMENT REFERENCE LENGTH  (BREF)
  1.00000E+00 = PITCHING MOMENT REFERENCE LENGTH (CREF)
  1.00000E+00 = YAWING MOMENT REFERENCE LENGTH   (BREF)
  0.00000E+00 = X - COORDINATE FOR THE POINT ABOUT WHICH MOMENTS WILL BE CALCULATED  (XREF)
  0.00000E+00 = Y - COORDINATE FOR THE POINT ABOUT WHICH MOMENTS WILL BE CALCULATED  (YREF)
  0.00000E+00 = Z - COORDINATE FOR THE POINT ABOUT WHICH MOMENTS WILL BE CALCULATED  (ZREF)
```

Figure 5.1: Quick Summary of Inputs

*** WARNING ***	Identifies edges which do not abut another network edge or a plane of symmetry; doublet strength along these edges assumed to be zero.
** GENTLE REMINDER ***	Identifies downstream wake edges, which generally do not abut other network edges.

- Numbering of abutments in order of network edge numbers for each network as input.

The Abutment Summary describes the connectivity of the networks making up the configuration. In the process of analyzing this connectivity, the network surfaces are examined for consistency of the material properties (total pressure and temperature) associated with each surface. If an inconsistency is discovered, an error message is issued describing which networks are connected together and which surfaces have conflicting definitions of material properties. This usually occurs when a network has been specified with a confusion on the user's part about which network surface is the upper surface and which is the lower surface. Recall that by convention, the normal to the upper surface of the network should point out into the flow field. It is often the case that a single instance of confusion as to upper or lower network surface will trigger several messages about incompatible surface properties. The user is advised to look at each of the networks mentioned in the printout to determine where the confusion has arisen.

The solution unknowns μ are typically used to establish a Kutta condition at trailing edges of wings. These unknowns are defined to exist at the certain locations on wake networks. These points are called "B points". In addition to their use to establish a Kutta condition, the μ's must satisfy certain continuity conditions where network abutments come together. These continuity conditions are called "MU MATCH" conditions. Appropriate B points are selected for establishing the Kutta condition and μ matching conditions. A summary of the boundary condition selected for each B point is printed in a table labeled "B POINT BOUNDARY CONDITION TABLE". After this summary a description of which μ parameters are used to satisfy the continuity condition is described in a table labeled "MU MATCHING CONDITION DESCRIPTION".

The **Input Processor** then computes a smoothed normal distribution over the configuration surface. Should the smoothed normal differ too much from the average panel normal, a warning message is printed. If such a message appears, the user should check the configuration definition to be sure it is described as it was intended.

The **Input Processor** then prints a final statement at the conclusion of its processing:

```
COMPLETING INPUT PROCESSOR PROGRAM
```

This concludes the output of the **Input Processor** module.

5–2 SOLVER PRINTOUT

The **Solver** module produces much information that is not typically of interest to the user. Thus, a detailed description of this information is not provided in this manual. However, in some cases the user may want to look at the full solver output to get more information on convergence behavior or about the decisions made in the adaptive refinement procedure. For that reason, a brief overview of the contents of the solver printout is provided below.

The full **Solver** output contains (in order) information concerning

- Material properties
- Initial grid construction
- For each grid
  - Oct-tree data structure
  - Numbers and types of boxes and unknowns
  - Green's function
  - Initial residual
  - Sparse Matrix Decomposition
  - For each Newton Step until converged
    - * GMRES convergence history of preconditioned residual for linear problem
    - * Stepsize used for Newton step, new residual and relative residual
  - Adaptive grid refinement decisions (if in adaptive grid mode)
- Solver Summary Information

At the end of its execution, the **Solver** prints a succinct summary page which *does* contain information of interest to the user. The summary of information from **Solver** is printed after the heading

```
SOLVER SUMMARY INFORMATION
```

This summary contains the following information:

- Title
- Flow condition
- Type of run (Grid Sequencing/Adaptive) and number of grids
- For each grid

  - Grid Information
    * Global grid description and maximum level of refinement
    * Number of boxes in the grid
    * Number of finite elements
    * Number of unknowns
  - Number of supersonic points and highest Mach number
  - Convergence information
    * Initial residual, final residual and relative residual
    * Number of Newton steps, total number of linear iterations in GMRES, number of Jacobians computed
  - Cpu time for the solution on this grid

- Total cpu time for the solver
- Convergence banner

In particular the user should ensure that the problem has been converged to the level specified by the user in the input file.

5–3 OUTPUT PROCESSOR PRINTOUT

The **Output Processor** describes the surface solution properties, configuration forces and moments and sectional properties if they have been requested. The output begins with the title information provided in the **$TIT** keyword, and a description of the global grid. This is followed by a description of the freestream flow field.

Next, a summary of the values of the μ parameters are printed on a network by network basis. These parameters exist only at certain locations on wake networks. The network index of each network containing a μ is printed along with the number of network columns and rows, a number indicating the type of wake network (Design Wake: 6, Trailing Wake: 18, or Carryover Wake: 20), and the network name. After this the values of the μ unknowns located on that network are printed. This information is not usually of primary interest to the user.

Following this is a rather long summary of the surface solution properties. These properties are printed on a network by network basis and may be printed either at each corner point or at each panel center point. (The data at each panel center point is computed by averaging data at corner points. This makes center point data less accurate than corner point data. It is recommended that users do not request surface data at panel center points.) The selection of corner point or center point data is made by the **$PRI** keyword.

If corner point data is printed, the **Output Processor** prints a table with the network column number and row number of each corner point of the network, an index indicating whether the data is for the upper or lower surface (upper surface = 1 and lower surface = 2), an O-box index for the O-box which contains the corner

point, the (x, y, z) coordinates of the point, the components of the velocity vector at the corner point, the value of the potential at the corner point, the normal mass flux at the point, the Mach number at the point and the pressure coefficient, C_p.

While it is not recommended that data at panel center points be printed, the **Output Processor** will print a somewhat wider variety of information at panel center points. A brief table of information containing twelve pieces of information may be specified in the **$PRI** keyword, or a longer table of information containing forty-eight pieces of information may be selected. The brief table contains the column and row indices of the center point, the (x, y, z) coordinates of the point `X`, `Y` and `Z`, the velocity vector on the upper and lower surface of the panel (`VXU`, `VYU`, `VZU`) and (`VXL`, `VYL`, `VZL`) respectively, the density, Mach number and isentropic pressure coefficient for the upper and lower surfaces (`RHOx`, `LMACHx`, and `CPISNx`, respectively), and the source and doublet strengths at the panel center (labeled `SOURCE` and `DOUBLET`).

The more extensive table contains the column and row indices of the center point, the (x, y, z) coordinates of the point, an equivalent doublet strength at the point and the doublet gradient (`DO` and `D1X`, `D1Y`, `D1Z`, respectively), the source strength `SO`, the normal vector scaled by the panel area (`ANX`, `ANY`, `ANZ`), the upper and lower (marked by a suffix `U` or `L` respectively): Mach number `LMACHx`, velocity vector (`VXx`, `VYx`, `VZx`), mass flux vector (`WXx`, `WYx`, `WZx`), the potential denoted by `PHEx`, four values of pressure coefficients computed under linear assumptions `CPLINx`, slender body assumptions `CPSLNx`, second order expansion assumptions `CP2NDx`, and isentropic assumptions `CPISNx`, the inner product of the total mass flux times the normal vector `WNx`, the inner product of the perturbation mass flux times the normal vector `PWNx`, the magnitude of the tangential component of the total velocity `VTx`, the magnitude of the tangential components of the perturbation velocity `PVTx`, (in the preceding, a suffix `-x` indicates `U` for upper and `L` for lower), and finally, the four pressure formulas for the difference pressure `CPLIND`, `CPSLND`, `CP2NDD` and `CPISND`.

If force and moment data have been requested, TranAir prints the network force and moment data following the output of surface flow properties for each network. The printout is controlled by the parameter $iprfm0$ in keyword **$PRI**. The 3-D forces (FX, FY, FZ) and moments (MX, MY, MZ) are calculated by integration of the pressure coefficient over the input nonwake network surfaces. The pressure distribution is assumed to vary linearly over each panel. The surface area of the network is also included in the force-moment printout.

If sectional properties have been requested, the **Output Processor** next prints information describing the sectional properties. This is described in detail in the following subsection.

A table with a summary of the total configuration forces and moments ends the output of **Output Processor**.

Sectional Properties

The tables describing sectional properties are divided into four main parts. The **Output Processor** always outputs the first two parts, which give cut reference data used for developing cuts and integrated properties along each cut. The remaining

output parts are optional. The third part gives integrated properties for each network, while the fourth part contains pressure along a cut. Most users confine their interests to the first two parts.

The following symbols are commonly used in the printout:

Symbol	Description
MXP, MYP, MZP	Reference coordinate moment components along a panel cut (see the equation below).
$CREF$	Reference chord for 3-D configuration (input under **$REF**)
NX, NY, NZ	Reference coordinate unit normal vector for a panel.
u	Unit normal vector parallel to onset flow $(cos(\alpha)cos(\beta), sin(\beta), sin(\alpha)cos(\beta))$.
l	Unit lift vector in cut plane $\frac{\hat{u} \times \hat{n}}{\| (\hat{u} \times \hat{n}) \|}$ (5.1)
d	Unit drag vector in cut plane $(\hat{n} \times \hat{l})$.
R_e	Reference coordinate position vector of exit point from a panel and a cut plane.
R_i	Reference coordinate position vector of entry point into a panel and a cut plane.

The reference coordinate moment components along a panel cut are given by:

$$\vec{M} = (\frac{-s}{c_2} \times [\frac{(R_e - R_i) \times (Cp_e - Cp_i)}{12} + \frac{(R_e + R_i) \times Cp_{avg}}{2} - RR] \times \hat{n} \qquad (5.2)$$

For each cut, the **Output Processor** prints the following data:

Header	Output Description
CUT NO.	Cut number.
ETA	Coordinate for cut data; e.g., semi-span fraction or span coordinate for a wing.
XC, YC, ZC	Reference coordinates defining cut plane (input).
XCN, YCN, ZCN	Components of unit normal defining cut plane $\hat{n}$ (input).
XR, YR, ZR	Reference coordinates defining moment reference location (RR).
$CHORD$	Chord used to nondimensionalize sectional forces and moments.
CUT NO.	Cut number.
ETA	Coordinate for cut data; e.g., semi-span fraction or span coordinate for a wing.
CFX, CFY, CFZ	Reference coordinate force components.
CMX, CMY, CMZ	Reference coordinate moment components.
CDC, CNC, CLC	Induced drag in cut plane, force normal to cut, lift in the cut plane ($CF \cdot \hat{u}$, $CF \cdot \hat{n}$, $CF \cdot \hat{l}$); represents true drag and lift only if y = constant for the cut and the flow has one plane of symmetry.
$CLC * CHORD/cref$	Load in cut plane.
CMC	Moment normal to cut plane ($CM \cdot \hat{n}$).
$CUT - LENGTH$	Length of cut.

If additional portions of the sectional properties data has been selected ($iprtnf$ = 1), some additional tables are printed. These include the force-moment data for each section of a network. It is identical to the printout for the cut force-moment data except that it is broken down into network totals.

If the fourth part of the sectional properties data has been selected ($iprtpp$ = 1) the following additional tables are printed.

Header	Output Description
X, Y, Z	Reference coordinate location of middle of cut path across a panel.
CP	Average pressure coefficient: $((C_{p_i} + C_{p_e})$ * 0.5).
FXP, FYP, FZP	Reference components of force component along a panel cut: $((-\frac{s}{c}) * C_{p_{avg}} \hat{n})$.
$CUT - SEGMENT$	Length of cut along a panel.

5–4 ADDITIONAL DATA FROM THE OUTPUT PROCESSOR

The **Output Processor** creates a number of files with surface aerodynamic properties, forces and moments, and sectional properties. These files are ASCII files and are formatted in a standard manner to enable their use with plotting programs commonly used at Boeing. These files are described in Appendix B.

In addition, the **Output Processor** creates two files describing the surface solution which may be used with the PLOT3D plotting program developed at NASA/ARC. The PLOT3D files are multiple grid files with each network defined to be a grid of size NM by NN by 1. The grid file is called AELP3DG. The Q file for PLOT3D contains the density, the momentum vector and the energy, normalized according to PLOT3D conventions ($\rho_\infty = 1$ and $c_\infty = 1$). The Q file is called AELP3DQ. Note that if the solution has regions where the local Mach number exceeds the fictitious gas Mach number, PLOT3D will find an entropy variation in those regions. Thus the solution and some thermodynamic properties computed by PLOT3D may be in error in those regions where the Mach number exceeds the fictitious gas Mach number.)

Appendix A– INSTALLATION OF TranAir

This appendix describes how to install and run TranAir on a Cray Y-MP machine using the UNICOS operating system. In most cases examples of control statements are provided to perform certain procedures. It is anticipated that the control statements or commands presented in the following as examples may not be exactly applicable to a specific machine in a specific installation. However, they should provide a good starting point.

A–1 BACKGROUND

A–1.1 Program Libraries

The source code for TranAir is maintained using the **update** utility. Table A.1 provides a list of program libraries needed to successfully compile and load the TranAir system of programs. Associated with each program library is its fully compiled version (e.g. fdsol.a for fdsol.ymp).

A–1.2 Building Executable Programs

The code may be compiled and loaded into executable programs using the `cft77` compiler and the segment loader called **segldr** on the Cray machines. It is necessary to build four executable programs, namely **fdinp, fdsol, fdout, fdcic**, for the four main modules of the system. Building any of the four executable programs mentioned above requires some or all of the program libraries. Table A.2 lists these libraries. The program also uses standard Cray **SCILIB** routines.

A–1.3 Commonly Used Code Updates

The TranAir code uses a self contained memory management system to allow efficient use of the available central memory. Most of the space used in the code is contained in one large array. Different routines in the code divide this space based on the need at the particular time. One major advantage of this procedure is to allow the user to control the amount of memory needed for his specific case by changing only one statement in each main program.

Table A.1: Program Libraries for the TranAir Code

Name	Description of Types of Routines
fdinp.ymp	The driver and the associated routines for the input processor.
fdsol.ymp	The driver and the associated routines for the solver.
fdout.ymp	The driver and the associated routines for the output processor.
fdcic.ymp	The driver and the associated routines for a program to convert the Cray binary graphics file into an Iris binary file.
gpulb.ymp	General purpose utility routines to perform such functions as input/output, specialtized printouts, etc.
gpmlb.ymp	General purpose mathematical routines to perform such functions as matrix-vector multiplications, etc.
spmlb.ymp	Special purpose mathematical routines to perform such functions as oct-tree data structure manipulation.
spslb.ymp	Sparse matrix solver routines
grflb.ymp	Greens function and exterior solver routines.
abtlb.ymp	Routines to analyze and force abutment.
fdlib.ymp	Special routines required in fluid dynamics problems such as isentropic thermodynamics conversion routines.
bdylb.ymp	Boundary layer routines from BLGL and A411.
callb.ymp	Special routines written in CAL (Cray Assembly Language) to increase speed of execution.
mylib.c	Special "C" routines used on the Y-MP to make UNICOS implementation easier.

It is also possible to modify routines in the libraries to improve the efficiency of the code, to provide specific data to the program, or fix possible errors while the executable programs are being built. Commonly used **update** modification sets (called *modsets*) are listed in Table A.3. Specific dataset names which contain these modsets are also indicated in the table. These modsets are delivered with the code.

A-1.4 Input/Output

The TranAir system uses a set of I/O manager routines for manipulating temporary datasets. These datasets are used to store data so that central memory can be freed for some other purpose. These datasets can reside on the SSD, the disk, or in main memory. Data stored on such datasets includes the oct-tree data structure, the boundary operators, GMRES search directions, and the decomposition of the Jacobian matrix.

TranAir also generates data required to communicate between the TranAir modules and databases needed to post process the solution. FORTRAN unit numbers 1 through 20 are reserved for this purpose. Typically, the user needs to interact with (save or access) only these datasets. Table A.4 provides a list of such datasets. (See also Appendix B for a description of ouput files created in the course of a coupled boundary layer analysis.)

Table A.2: Dependence of Programs on Libraries

Name	Required Libraries	Comments
fdinp	abtlb.a, gpulb.a, gpmlb.a, spmlb.a, fdlib.a	The driver and the associated routines for the input processor.
fdsol	gpulb.a, gpmbl.a, spmlb.a, spslb.a, grflb.a, fdlib.a, bdylb.a, callb.a	The driver and the associated routines for the solver.
fdout	gpulb.a, gpmlb.a, spmlb.a, fdlib.a	The driver and the associated routines for the output processor.
fdcic	gpulb.a	The driver and the associated routines for a program to convert a Cray binary dataset into an Iris binary dataset.

Table A.3: Commonly Used `update` Modsets

Modset	Library	Dataset	Comments
bigfdc	fdcic.ymp	bigfdc	Used to increase the main scratch array in the binary conversion program **fdcic**.
bigout	fdout.ymp	bigout	Used to increase the main scratch array in the output processor **fdout**.
bigsps	spslb.ymp	bigsps	Used to pack integer and real data into one word. This reduces the storage required to run the program by half without significantly affecting CPU cost.
bigcas	fdsol.ymp	bigsol	Used to increase the main scratch array in the solver**fdsol**.
bignes	fdsol.ymp	bigsol	Used to reduce memory requirements in the generation of the nested dissection ordering.
ssdsc2	fdsol.ymp	bigsol	Used to pack the real operator data (2 words into each storage location).
ssdsc3	fdsol.ymp	bigsol	Compacts the sparse solver datasets on the disk and puts them on the disk.

Table A.4: User Datasets Generated by TranAir

Name	Unit	Type	Purpose
fort.4	4	Binary	Grid and flow visualization database
fort.5	5	ASCII	Standard input
fort.6	6	ASCII	Standard output
fort.7	7	Binary	Database for communication between the input processor and the solver. The same dataset is used in a restart run.
fort.8	8	Binary	Database for communication between the solver and the output processor.
fort.10	10	ASCII	Database for surface corner point aerodynamic quantities
fort.11	11	ASCII	Database for sectional properties summary
fort.12	12	ASCII	Database for forces and moments summary
fort.14	14	ASCII	Database for sectional properties plot data
AELP3DG	15	Binary	Surface geometry visualization in PLOT3D.
AELP3DQ	16	Binary	Surface flow visualization in PLOT3D.
BLGGP	10	ASCII	Bounary Layer Solution Parameters On Network Points.

A–2 INSTALLING TranAir UNDER UNICOS

The following discussion also assumes that the user is in Bourne Shell in UNICOS. The discussion is limited to the applications at the NASA AMES ACF center and assumes the computer installation includes a Cray Y-MP (known as **eagle**) for numerical processing and a Cray X-MP (known as `columbia`) for file storage. Data and program files are assumed to reside on `columbia`. They are remotely accessed by a job running on **eagle**. It is assumed that the user's `.rhost` file has been properly configured on both systems to allow the remote copy command `rcp` to be correctly exectuted.

A–2.1 Installation

Parts List

The following files are delivered as a part of the TranAir system of programs and associated libraries.

- Program `PL` files (with extension `.ymp`). The `PL` files are delivered in (UNICOS) `PL` format. A `Makefile` has been furnished for installation and may be used to create executables with or without modifications from the program libaries. Table A.1 provides the list of the program libraries.
- Frequently used **update** modsets. These mods are developed to make TranAir run faster and in smaller amount of memory. See Table A.3 for a list of the program libraries.
- A special C library in file `mylibc.c` which allows certain system calls.
- `Makefile` (a file used by **make** called **makeTranAir**),
- Sample job deck (file `TranAir.yjb`) used to run TranAir on the Cray Y-MP system. This file contains the sample job that can be substituted to NQS and invokes appropriate **make** calls to build and execute the TranAir programs.
- Standard input deck (file `tstc.i.inp`).
- The TranAir graphics program `tgraf` (see next Appendix for instructions on installation and use).
- The boundary layer post processing tools, consisting of a number of separate Fortran source files bundled together as a Unix archive file called `Tblpproc.ar`.

Storing TranAir on `columbia`

The TranAir program libraries, compiled libraries, executable files and modsets should be stored in separate subdirectories in a directory which is commonly accesible to all who need to use the code. In what follows it will be assumed that the parent directory has been created on `columbia` and it will be referred to as **$Thome**. Under

$Thome there will be four subdirectories: **$Thome/pls**, **$Thome/lib**, **$Thome/mods**, and **$Thome/bin**. It is recommended that a Unix shell variable be created as a shorthand notation for the parent directory of the TranAir files. In what follows, it is assumed that such a shell variable has been defined and is called **$Thome**.

The update program library files are to be stored on the **$Thome/pls** directory. The libraries of linkable object code are stored on **$Thome/lib** and the executable files are stored on **$Thome/bin**.

The Makefile furnished with the TranAir system can be used to create the Fortran source code using the update program, compile it and build libraries of linkable object code and finally executable binary code. Copy all of the files included with the TranAir distribution in to the current working directory. Then run the make program as:

```
export Thome
make -f Makefile.install install
```

Approximately 1200 CPU seconds should be allowed for the job to compile and prepare the libraries for all of the TranAir code. If alternative executables are desired for running larger problems, the furnished modsets now located in **$Thome/mods** may be used to create a modified version of the exeuctables. For example, to create a modified version of the solver which will solve a larger problem more efficiently, the following procedure will automatically create the executable and store it on columbia.

```
rcp columbia:$Thome/mods/bigsol.mod fdsol.mod
rcp columbia:$Thome/mods/Makefile.mod Makefile.mod
make -f Makefile.mod fdsol
rcp fdsol columbia:$Thome/bin/bigfdsol
```

A–3 Running TranAir

After TranAir has been installed on the columbia machine it may be run directly from the executables in the **$Thome/bin** subdirectory. Input files describing the configuration and flow conditions may be defined for each of the four executable modules. In the examples that follow it is assumed that the files are available on columbia in the user's home directory and are named by a case name identified by the shell variable **$case**.

At the end of the execution of each module in TranAir, a number of permanent files should be renamed and copied into either the file system on columbia or to a workstation networked with the eagle computer system. In the sample scripts which follow, a case name is defined and used to name the output files. Comments have been inserted to identify which output files are being transferred.

```
case=B747a
CASEPATH=columbia:/csf/ra/rac/bussolet/cases
Thome=columbia:/csf/ra/rac/bussolet/release
```

A-3

```
export case Thome
#
Transcr='tmpdir /scratch'
cd $Transcr
# working subdirectory is created if it does not exist.
if test ! -d $case
then
    mkdir ./$case
fi
cd $case
localdir='pwd'
rcp $Thome/bin/bigfdinp fdinp
rcp $Thome/bin/bigfdsol fdsol
rcp $Thome/bin/fdcic fdcic
rcp $Thome/bin/bigfdout
rcp $Thome/bin/Tsgpgl Tsgpgl
rcp $Thome/bin/Tcwngvp Tcwngvp
#
#  get input files
rcp columbia:$CASEPATH/$case.i.inp $case.i.inp
rcp columbia:$CASEPATH/$case.poi $case.poi
rcp columbia:$CASEPATH/$case.s.inp $case.s.inp
rcp columbia:$CASEPATH/$case.c.inp $case.c.inp
rcp columbia:$CASEPATH/$case.o.inp $case.o.inp
#
# Run the input processor
fdinp <$case.i.inp >$case.io 2>$case.ie
rcp MSPTS columbia:$CASEPATH/$case.msp
rcp $case.io columbia:$CASEPATH/$case.io
rcp $case.ie columbia:$CASEPATH/$case.ie
rcp fort.7 columbia:$CASEPATH/$case.tp7
#
# Run the solver
fdsol <$case.s.inp >$case.so 2>$case.se
rcp $case.so columbia:$CASEPATH/$case.so
rcp $case.se columbia:$CASEPATH/$case.se
rcp fort.7 columbia:$CASEPATH/$case.tp7
rcp fort.8 columbia:$CASEPATH/$case.tp8
rcp fort.4 columbia:$CASEPATH/$case.tp4
ar crul $case.blar BLGL* A411* NXZCP NA4* BLPR CREF* ATCHL*
rcp $case.blar columbia:$CASEPATH/$case.blar
#
# Run the binary converter
mv fort.4 fort.3
fdcic <$case.c.inp >$case.co 2>$case.ce
```

```
rcp $case.co columbia:$CASEPATH/$case.co
rcp $case.ce columbia:$CASEPATH/$case.ce
rcp fort.4 columbia:$CASEPATH/$case.bn
rcp fort.2 columbia:$CASEPATH/$case.ybn
#
# Run the post processor
fdout <$case.o.inp >$case.oo 2>$case.oe
rcp $case.oo columbia:$CASEPATH/$case.oo
rcp $case.oe columbia:$CASEPATH/$case.oe
rcp fort.10 columbia:$CASEPATH/$case.ggp
rcp fort.11 columbia:$CASEPATH/$case.sp.ggp
rcp fort.12 columbia:$CASEPATH/$case.fm.ggp
rcp fort.14 columbia:$CASEPATH/$case.fl.ggp
rcp AELP3DG columbia:$CASEPATH/$case.p3dg
rcp AELP3DQ columbia:$CASEPATH/$case.p3dq
rcp BLGGP columbia:$CASEPATH/$case.blggp
#
# Run boundary layer post processing
Tsgpgl >$case.blo 2>$case.ble
rcp $case.blo columbia:$CASEPATH/$case.blo
rcp $case.ble columbia:$CASEPATH/$case.ble
rcp BLCDF columbia:$CASEPATH/$case.blggp
#
```

Restarting a Run

In some cases it may be desirable to restart a run. For example, the original run did not converge or run to completion, or one may want to slightly perturb few of the controlling parameters or flow conditions. During a restart run the keywords **$ITE**, **$MAC**, **$ANG**, **$YAW**, or **$TOL** may be input as input to the solver module `fdsol`.

If a given case is to be restarted from an old one then it would be necessary to save certain files in the original run and then access those files in the restart run. The file that is to be saved is the FORTRAN unit 7 (`fort.7`). It is not necessary to run the input processor in this case. However the remaining three programs have to be run separately. A typical portion of the script used to restart a run is shown below:

```
case=mycase
Thome=columbia:/u/ra/madson/tranair
export case Thome
#
Transcr='tmpdir /scratch'
cd $Transcr
# working subdirectory is created if it does not exist.
if test ! -d $case
then
    mkdir ./$case
fi
cd $case
localdir='pwd'
rcp $Thome/bin/fdsol fdsol
rcp $Thome/bin/fdcic fdcic
rcp $Thome/bin/fdout
rcp $Thome/bin/Tsgpgl Tsgpgl
#
#  get input files
rcp columbia:$HOME/$case.tp7 fort.7
rcp columbia:$HOME/$case.s.inp $case.s.inp
rcp columbia:$HOME/$case.c.inp $case.c.inp
rcp columbia:$HOME/$case.o.inp $case.o.inp
#
# Re-Run the solver
fdsol <$case.s.inp >$case.so 2>$case.se
rcp $case.so columbia:$HOME/$case.so
rcp $case.se columbia:$HOME/$case.se
rcp fort.7 columbia:$HOME/$case.tp7
rcp fort.8 columbia:$HOME/$case.tp8
rcp fort.4 columbia:$HOME/$case.tp4
ar crul $case.blar BLGL* A411* NXZCP NA4* BLPR CREF* ATCHL*
rcp $case.blar columbia:$HOME/$case.blar
```

```
#
# Run the binary converter
mv fort.4 fort.3
fdcic <$case.c.inp >$case.co 2>$case.ce
rcp $case.co columbia:$HOME/$case.co
rcp $case.ce columbia:$HOME/$case.ce
rcp fort.4 columbia:$HOME/$case.bn
rcp fort.2 columbia:$HOME/$case.ybn
#
# Run the post processor
fdout <$case.o.inp >$case.oo 2>$case.oe
rcp $case.oo columbia:$HOME/$case.oo
rcp $case.oe columbia:$HOME/$case.oe
rcp fort.10 columbia:$HOME/$case.ggp
rcp fort.11 columbia:$HOME/$case.sp.ggp
rcp fort.12 columbia:$HOME/$case.fm.ggp
rcp fort.14 columbia:$HOME/$case.fl.ggp
rcp AELP3DG columbia:$HOME/$case.p3dg
rcp AELP3DQ columbia:$HOME/$case.p3dq
rcp BLGGP columbia:$HOME/$case.blggp
#
# Run boundary layer post processing
Tsgpgl >$case.blo 2>$case.ble
rcp $case.blo columbia:$HOME/$case.blo
rcp $case.ble columbia:$HOME/$case.ble
rcp BLCDF columbia:$HOME/$case.blggp
#
```

A similar restart run for the Output Processor can also be made. In that case it is necessary to fetch the file **$case.tp8** which was stored in the earlier run and the program to be executed is invoked in a similar fashion.

A–4 EXECUTION STATISTICS

In this section data is provided which illustrates the computing resources required to run TranAir cases. The primary resources that are usually charged to the user include CPU time, central memory usage, and SSD usage.

These resources are generally functions of the case that is being run. At this time no formulas exist that will indicate the exact amount of resources required for a given case. The size of a case to be run in TranAir is typically designated by the following parameters.

- Number of Panels ($NPAN$)
- Number of unrefined boxes in the grid ($NRBXS$)
- Number of finite element regions used by the program ($NREG$)
- Number of unknowns (NEQ)
- number of equations solved in the Sparse Solver ($NRED$)

It should be noted that these parameters are not truly independent, e.g., $NREG$ and $NRBXS$ are two different measures of the number of boxes in the grid. Table A.5 summarizes the resources used in several sample cases. In the table, $NSSD$ is the number of SSD blocks used, CPU is the number of cpu seconds used, and CM is the number of blocks of central memory used when TranAir was applied to analyze some nontrivial cases. The statistics shown in this table reflect the sizes of cases for which results are shown in the Theory Document of this report.

Table A.5: Resource Requirements for TranAir Execution

Case	$NPAN$	$NRBXS$	$NREG$	$NRED$	$NSSD$	CPU	CM
Sphere	1,600	19,708	15,956	13,298	11,616	215	1.63MW
ONERA M6	3,921	358,635	312,855	237,471	163,000	4,475	3.87MW
F-16	7,142	238,526	215,954	182,818		2,967	4MW
B747-200	20,000	257,000	219,000	208,000	105,000	4,500	4MW

Appendix B- TRANAIR OUTPUT DATASETS

B-1 OVERVIEW

This chapter describes the datasets generated by the TRANAIR program and how to process information contained in them. The following table gives the datasets that have aerodynamic information either on the configuration surface or the grid.

Utility Name	Dataset/Unit	Type	Purpose
TGRAF	FT04 (4)	Binary	Grid and flow visualization.
PLOT3D	AELP3DG(15)	Binary	Surface geometry visualization.
	AELP3DQ(16)	Binary	Surface Flow visualization.
2D Plotting Package	FT10(10)	ASCII	2 Dimensional plots for surface geometry and aerodynamic properties.
	FT11(11)	ASCII	Sectional properties summary.
	FT12 (12)	ASCII	Forces and moments summary.
	FT14(14)	ASCII	Sectional properties plot data.

The dataset containing the grids and flow field properties, (FT04), is called the TRANAIR graphics dataset. This is a binary dataset in IRIS[1] [20] binary form created on the Cray. The primary utility used to visualize the grids and flowfield data is called *TGRAF* and is provided with the code. This code runs on the IRIS workstations.

TRANAIR produces two datasets which have the flow data at the surface corner points in the format acceptable to PLOT3D. These datasets can be used to postprocess the surface solution using PLOT3D.

TRANAIR also produces datasets that provide aerodynamics data for two dimensional plotting. These datasets are written in a specific ASCII format for particular

[1] IRIS is the registered trademark of the Silicon Graphics, Inc. of Mountain View California.

graphics utilities at Boeing. The formats of the data in these datasets has to be changed to suit the needs of any other plotting packages that may be used. This can be done after the fact or by changing the subroutines that write these data.

In the following sections the *TGRAF* utility is described in some detail. Appropriate manuals should be consulted to learn how to use PLOT3D or the 2D plotting packages.

B–2 TRANAIR GRAPHICS UTILITY

B–2.1 Overview

TGRAF is a utility which can be used to interrogate and display selected data from a **TRANAIR** graphics dataset. Volume subsets of grids and planar sections from the grids can be extracted and displayed. In addition, a limited capability to shade properties related to the aerodynamic or numerical behavior of the solution exists. Hardcopies can be made from the displayed data for both PostScript[2] and Tektronix inkjet printers.

The **TRANAIR** graphics dataset, FT04, contains some header information, network panel data, and special regions of interest (**LBO's**). For each grid used to find the solution, it also includes the oct-tree data and a series of records containing the flow field properties such as the density, Mach number, error indicators etc., (generically called **property** in the subsequent discussion).

It is noted that *TGRAF* is written specifically for the unstructured grids used in TRANAIR. It is also written for a specific workstation (it runs on the Silicon Graphics IRIS 3000 and 4D graphics workstations). It has been used considerably during code development and has been found to be fairly robust and reliable. However, the program has not been used in production environment and some of its input/output features are not totally polished. This utility is offered as a tool to help assimilate information and no claims are made regarding its robustness.

B–2.2 Starting TGRAF

To use *TGRAF* on the IRIS 3000's, the window manager **MEX** must be running on the workstation. If **MEX** is not running then it can be started using the following command.

% mex

The *TGRAF* utility is invoked using the following command:

% tgraf *case.bn*

where *case.bn* is the *TGRAF* dataset.

[2]PostScript is the registered trademark of the Adobe Systems, Inc.

When *TGRAF* is ready for user interaction, the workstation screen will contain several windows, including

- Graphics Work Area
- Transformations Menu
- Control Menu

The graphics work area window, named *TRANAIR Graphics*, is used to display graphical objects. It initially contains all networks and **LBO**'s used in the problem. Temporary menus are drawn over the same screen space during the execution of the program. All windows used in the *TGRAF* can be moved and resized using the **MEX** user interface [20].

Typically, a menu consists of **buttons** and/or **slider** areas (highlighted rectangular regions). A specific selection or an action is invoked by locating the cursor in the highlighted area and pressing the left mouse button.

B–2.3 Applying Transformations

The orientation of the display is controlled through the *Transformations* menu. This menu is always present on the screen while *TGRAF* is running. Selections from this menu allow the user to translate, rotate and scale the display in the work area. The *Transformations* menu is divided into three areas, one containing rotational controls, another containing scaling controls, and the last containing translation controls.

Rotation controls are located in the upper portion of the *Transformations* menu. Initially the controls consist of three **sliders** (one to rotate about each of the coordinate axes)and a **Reset** button that restores the rotations to a default position. Rotations occur about the center of the screen. Rotations made with the **slider** controls are composited (where rotations are applied to the current view). The **Fixed** button can be used to replace the rotation sliders with fixed view selectors. Pressing the **Fixed** button will cause objects in the work area to be redrawn in their default rotational orientation and will cause the rotation sliders to be replaced by a set of buttons which can be used to select a fixed view. The rotations made by the fixed view buttons are not composited and are applied exactly once to the x, the y, and the z coordinate axes in that order.

The sliding controls may be accessed by pressing the **Fixed** button which is then replaced by the **Slider** button. Pressing this button causes the fixed view buttons to be replaced by the sliding rotation controls, leaving the orientation in the *TRANAIR Graphics* window unchanged.

The scale of the display can be controlled with the **scale** slider. Non-positive scales are not allowed. The scale factor is applied equally to all three coordinate axes and scaling occurs about the center of the *TRANAIR Graphics* window. Unit scale can be restored by pressing the **Reset** button located near the **scale** slider.

The rate which scaling occurs can be changed by pressing the **Speed** button. Pressing the **Speed** button causes the *Set Scale Speed* control menu to appear. The speed control consists of a single slider. The value of the slider (displayed under the control) represents the rate at which scaling is applied by moving the **scale** slider in the *Transformations* menu from its extreme left position to its extreme right position. Increasing the rate of scaling allows large scale factors to be applied without excessive manipulation of the scale control. Decreasing the rate of scaling allows finer control over the scale factor.

Translations are performed by locating the cursor in the *TRANAIR Graphics* display window and pressing the left mouse button. While the mouse button is held down, the displayed objects will follow the movement of the cursor. Releasing the mouse button will cause the scene to stop translating.

As a user convenience, a "gunsight" object will appear at the center of the *TRANAIR Graphics* window while translations are occurring. Since rotations and scales are applied at the center of the display, the gunsight assists the user in positioning the scene properly. The gunsight will disappear when the mouse button is released.

The recommended translation method is to first locate the cursor over the part of the display that is to be the center of interest. Press the left mouse button to begin translating. While holding the left mouse button down, move the cursor to the center of the gunsight and release the left mouse button.

The translation can be reset by pressing the **Reset** button in the *Transformations* menu near the **Translation** label. This button positions displayed objects so that their center of mass is the center of rotation.

B–2.4 Control Panel

The second menu that appears provides various controls over the program. The functions provided on this control panel are

- **Extract**
- **Display**
- **Color**
- **Edit**
- **Script**
- **Exit**

These functions are described below:

B–2.5 Extractions

Selected volume subsets or planar sections in a given grid can be extracted using *TGRAF*. Extractions can be made by pressing the Extract button in the *Control Panel* menu. This will cause another menu named *Extractions* to appear. The *Extractions* menu provides controls for selection of a specific type of extraction.

In all types of extractions, the OK button must be pressed after all the relevant information for the extraction is specified. This causes the code to perform calculations necessary to obtain the required geometric information to create the graphical object to be displayed. It should be noted that some of these graphical objects can be large (indeed one can select the entire volume), and can take an appreciable amount of time to create. As the code is performing these actions the OK button is shaded dark blue indicating that the code is working. The Cancel button can be selected to cancel the particular extraction.

Most selections in the Extraction menu generate display objects which are added to a list of objects that can be displayed either individually or simultaneously. Each object extracted should be given a name so that it can be identified at a later time. No restrictions are made on the characters used in object names and names need not be unique. However, the practice of duplicating or omitting names can make object identification difficult and is not recommended.

When all extractions have been made, the Done button in the *Extractions* menu may be pressed to remove the menu from the screen.

Grid Selection

To select a specific grid from which the extractions are to be made the Grid button should be used in the *Extractions* menu. By default, the final grid in the dataset will be used if this selection is not made. A special selection menu named *Set Grid* will appear and display the names of grids that can be selected. A **scroll** control is provided within the menu. Should there be more selectable grids than can be concurrently displayed, non-visible grids can be brought into view for selection using the elevator control on the **scroll** control. A selection is made by pressing the left mouse button while the cursor is over the name of the desired grid.

Types of Extractions

The following types of specific extractions can be made:

- Plane
- Volume
- Point
- O-Box
- U-Box

- S-Reg
- OT-Reg

Planar Cut

These types of extractions are selected by pressing the **Plane** button in the *Extractions* menu. Selecting this item causes a plane selection menu named *Cut Plane* to appear. Planar cuts can be made through the selected grid by defining a plane using a normal vector and a point in the plane. The resulting object is the intersection of the plane with the entire grid, geometry, and **LBO**'s. The normal vector is selected by pressing the left mouse while the cursor is in the highlighted area under the "Type **Normal Vector: A B C**" button. The specification of the vector is made through three Cartesian coordinates of the normal vector along the x, y, and the z directions. These components need not be normalized. The numbers specifying the components must separated by at least one space. A point lying in the plane is similarly specified. The code will provide diagnostic messages if this input is provided improperly.

Volume Subset of a Grid

A rectangular subset **Volume** can be extracted out of the currently selected grid for display. **Volume** extractions can be made by pressing the **Volume** button in the *Extractions* menu. This selection causes a menu named *Cut Volume* to appear. The extent of the volume is defined by the coordinates of two diagonally opposite corner points of the desired rectangular region. The resulting object consists of all the grid, network geometry, and **LBO**'s contained within the specified box. Each corner is specified through its Cartesian coordinates in the x, y, and z directions. The coordinates are expected to be three numbers separated by white spaces. The code will provide diagnostic messages if this input is provided improperly. It should be noted that the display of volume grid can require large amounts of computing resources and can make the response time for the graphics package very slow. It is not advisable to cut large volume subsets from a fairly dense grids.

Other Extractions

The remaining types of extractions are variations of the **Volume** extraction which correspond to specific box types used in the TRANAIR code.

An **O-BOX** is **any** box in the TRANAIR grid. If an **O-BOX** is selected for extraction then all the refined grid boxes within that box at lower levels are also displayed as are any part of the configuration and **LBOs** contained within the volume.

A **U-BOX** is any **unrefined** box in the TRANAIR grid. If this option is selected then only one box and any configuration boundary and **LBOs** enclosed by this box will be displayed.

If it is desirable to cut a **U-BOX** which encloses a desired point the **Point** button should be used. The point is specified through its Cartesian coordinates in the x, y, and z directions. The coordinates are expected to be three numbers separated

by white spaces. *TGRAF* will issue a diagnostic message if the data is improperly specified.

An S-REGION is any **solver** region in the TRANAIR grid. Every finite element region in the problem is individually identified. All the regions not cut by the boundary are ordered first followed by those regions which are formed by the intersection of the boundary and the grid. Note that it is possible to have several regions located within one rectangular grid box if the geometry of the configuration divides that rectangular grid box. Each such subregion (dubbed a **D-region** in the Theory Document[2]) has a separate identity and can be extracted separately. However, the extraction process used in the code displays all the D-regions in the rectangular grid box. An **OT-REGION** is any **oct-tree** region in the TRANAIR grid. In identity, the **S-REGION** and the **OT-REGIONS** are equivalent except that their index within the code differs.

These extraction mechanisms are provided as a convenience to TRANAIR code developers. They are of little interest to TRANAIR users and use of these extraction types is not encouraged. Each of these types of boxes or regions is numbered in the TRANAIR code. To be able to display them in the *TGRAF* utility it is necessary to specify the index of the item. **Volume** extractions are the preferred method of cutting rectangular regions.

B–2.6 Display Manipulations

The objects drawn in the *TRANAIR Graphics* window can be controlled through the display panel which is accessed from the *Control Panel* button marked Display. Invoking this function will cause a window named *Edit Display* to appear containing user interface controls that change the visibility of the extracted objects.

One interface lists the names of all known objects. It always contains the network and **LBO** objects and it contains the names of objects that have been extracted. The vertical scrolling elevator can be used to move through the list of names. An object can be **selected** by pressing the mouse on the name representing the object. An object must be selected before its visibility can be changed, except when the visibility of all objects is being changed.

Each object consists of several sub-objects that can be individually included or omitted from the display. These sub-objects consist of the following:

- BLUE: This sub-object is the boundary of the extracted object. In the user specified **Plane** and **Volume** extractions these are the extents of those planes and volumes. In case of the other types of extractions these are boundaries of the equivalent rectangular regions in space. This sub-object is drawn with color blue.

- GREEN: This sub-object includes all the parts of the configuration boundary enclosed within the extracted objects. This sub-object is separate from the other objects identified with the names of the networks and can be displayed separately or together with the networks. These sub-objects are displayed in color green.

- RED : This sub-object is the intersection of the grid boxes with the cutting region. These sub-objects are displayed in red color.
- LBO : This sub-object includes all the parts of the **LBO**'s enclosed within the extracted objects. These sub-objects are displayed in magenta color.
- If the selected extraction is a planar cut through the grid it is possible to paint particular flow-field properties over this extracted plane. A set of flow properties is associated with each closed polygon in the RED sub-object. These polygons can be shaded based on a color that indicates the value of the selected property using a particular scale. Both FLAT and SMOOTH shadings can be applied.

It should be noted that sub-objects can be empty. Enabling visibility of an empty sub-object in an object has no visible effect in the work area.

Objects are drawn in the manner described by their visibility patterns. To alter the visibility of objects selected for display, the object is located in the list of object names and selected by pressing the left mouse button while pointing at the object's name. The information in the panel displaying the visibility structure (selected sub-objects) will change to reflect the current drawing state of the selected object. The visibility buttons can be altered as required and the ITEM button is pressed to set the visibility pattern for the selected object.

To change the state of all objects (e.g. make all objects invisible) the visibility buttons are changed to the desired pattern and the ALL button is pressed. This will cause all objects to be drawn in the manner described by the buttons.

Editing Objects

The Edit button is used to see details about a selected object, to enable painting properties on a **Plane** object, to delete an object, or to alter display of numerical information about an extraction. This function is typically used to enable a painting property on a plane. Selecting the Edit button causes a menu named *Edit Object* to appear.

A property to be painted on a **Plane** extraction is selected by locating its name in the list of paintable properties and pressing the left mouse button while the cursor is over the name. The property is applied to the selected object when the Prop button is pressed. Loading a property can take an appreciable amount of time. The Flat and Smooth visibility patterns in the *Edit Display* menu are used to cause the property to be painted on the object.

Pressing the Delete button will cause the selected item to be deleted. *TGRAF* will require verification before it deletes any object. The Delete function is used primarily to recover memory when an extracted object is no longer needed.

The Obox, Ubox, and Tbox buttons are used to display numerical information about intersections. These functions cause text to appear in the *TRANAIR Graphics* window showing the TRANAIR code indices of various boxes involved in

the selected object. This capability is provided as a convenience to TRANAIR code developers and its use is not recommended.

The Done button causes the *Edit Object* window to be removed from the screen.

B–2.7 Controlling Legends

To change the mapping of properties to colormap entries for shaded planes and to control legend visibility, the COLOR button in the *Control Panel* is used. To change visibility of a legend, select the legend name from the list presented. Toggle the legend visibility by pressing the Legend button. The slider on the right can be used to change the mapping bounds for the selected property. The display in the *TRANAIR Graphics* window will update when the mouse button is released.

B–2.8 Making Hardcopies

A limited output capability is provided to produce raster and PostScript image files. To create a hardcopy file, the Print button in the *Control Panel* menu is used. This action causes a hardcopy selection menu named *Print* to appear. A title for the hardcopy can be specified by pressing the Title button and typing in the desired text. A default title will be appended to any title specified. The Laser button is pressed to specify that a PostScript file is to be created. A filename may be specified for the PostScript file. If no filename is provided a default name is generated.

The Color button is used to specify that a Textronics raster file be created. A filename can be specified for the raster image file. If no file name is provided a default filename is used. The colormap data is stored in a second file with the suffix .map appended to it. The raster image can be edited with the **PIX** pixel editor utility.

The OK button is used to cause the display data to be written in the appropriate file(s). The Cancel button is used to return to the *Control Panel* without creating the print files.

B–2.9 Script Capability

The Script button in the *Control Panel* window creates a menu named *Script*. This menu provides a mechanism for specifying a file containing transformation, extraction, and printing commands that are to be executed. This feature is useful when an analysis procedure is applied in the same way to several (possibly) different TRANAIR graphics datasets and interactivity is not required.

Format of the Script File

The script file is a simple file containing extraction, positioning, display, and hardcopy commands. Commands must lie on one text line. However, long commands may be continued on another line by placing the *escape* character '\' as the last character of the line to be continued. The maximum length of a command line is 1023 characters.

The command interpreter is case dependent. All commands must be entered in lower case.

Commands may require arguments to qualify the commands. The arguments are values separated by white-spaces. No command requires more than nine arguments. Extra arguments are silently ignored. Spaces can be inserted into arguments by using the *escape* character or by quoting the argument with double quotes. The *escape* character causes the next character to be treated as simple text. The escaped character loses any particular "special" meaning it may have (e.g. a quoted space does not separate arguments).

Comments can be placed in the script file by placing the *comment* symbol '#' in the first position on a line. When the comment symbol is seen as the first character on a line, the remainder of that line is discarded.

The script file recognizes several commands whose functionality parallels that of the interactive menus in the *TGRAF* program. The commands are:

- **grid**: Selects a grid from database from which subsequent extractions are made. The **grid** command requires a single integer argument that is the grid number to load.

- **plane**: Specifies that a plane is to be extracted from the current grid. The **plane** command requires seven arguments. The first three are floating point values that describe a normal vector of the plane being defined. The next three arguments are floating point values that describe a point lying in the plane. The last argument is the name of the object that is to be created.

- **volume**: Specifies that a rectangular volume is to be extracted from the current grid. The **volume** command requires seven arguments. The first three describe the coordinates of the lower-left-near corner of the volume, the next three describe the coordinates of the upper-right-far corner of the volume. The last argument is the name of the object that is to be created.

- **clear**: Specifies that no objects are to be visible. This command ensures that the "screen" is clear when running a script. The **clear** command requires no arguments.

- **display**: Controls the visibility of objects. The **display** command requires at least one argument and can take up to five. The required argument is the name of the object whose visibility is to be altered. If no other arguments are given, the named object is made invisible. The other four arguments, if specified, must be one of the **red**, **green**, **blue**, or **LBO** keywords. The keywords can be specified in any order. Each keyword controls the visibility of one of the parts (*red*, *blue*, *green*, or *LBO*) of the named object. If more than one object share the same name, one object is selected by the program to be altered.

- **reset**: Resets the view to the default orientation. No rotations are applied, the center of mass of displayed objects is positioned at the center of the screen, and

the objects are scaled so that no part of any displayed object can clipped by appling a rotation. The **reset** command takes no arguments.

- **lookat**: Defines the geometric coordinate that is to be the center of rotation. The **lookat** command requires three arguments. The arguments are floating point values describing the coordinates of the point that is to be the center of rotation. The translations described by the **lookat** command are not composited.

- **scale**: Scales displayed objects. The **scale** command requires one argument. The argument is a positive floating point value describing the amount of scale to be applied to displayed objects. Scale values are not composited.

- **rotate**: Rotates displayed objects about the center of rotation. The **rotate** command requires two arguments. The first is a floating point value describing the number of degrees to rotate the scene. The second argument must be one of x, X, y, Y, z, or Z. This argument describes the axis about which the rotation is to occur. Rotations are composited.

- **laser**: Creates a file containing PostScript commands needed to draw displayed objects with a PostScript printer. The **laser** command requires two arguments. The first is the name of a PostScript file to be created. The second is a title to be applied to the PostScript picture.

An example script file is shown in Figure B.1.

B–2.10 Ending a Session

The Exit button in the *Control Panel* is used to exit the *TGRAF* program. It is necessary to confirm this request. After confirmation, all of the menus and windows associated with this *TGRAF* session are removed from the screen and the user is returned to the UNIX shell. On the 3000 series IRISen, **IT IS NECESSARY TO ALSO EXIT MEX AT THIS POINT** unless the user wants to continue doing other things within **MEX**.

B–2.11 Known Problems

Known problems and restrictions include:

- Polygons are not smooth shaded properly.

- Some inputs to **OBOX**, **UBOX**, **POINT**, and region cutters can cause program to fail.

- Memory allocation failure is not handled gracefully.

- Incorrect datasets are not handled gracefully.

```
# this is a script file for TRANAIR GRAPHICS
# Comments begin with a '#' in the first column
# Empty lines (as below) are ignored.

# turn all objects invisible

clear

# Reset SCALE, TRANSLATE, and ROTATIONS

reset

# Enable Grid #1

grid 1

# Cut a plane with normal (0, 1, 0) through point (0, .1, 0 )
# give the object a favorite name (e.g. "plane@grid1")

plane 0 1 0 0 .1 0 plane@grid1

# turn on red+green+lbo parts of cut
# Note that the blue part is NOT turned on.

display plane@grid1 red green sbo

# Position the display

lookat 0 0.1 0
scale 2
rotate -90 x

laser 1.ps "A view looking at (0,.1,0)"
#
# End of the script
#
```

Figure B.1: An Example *TGRAF* Script File

- Deleted **$LBO** and Network objects cannot be recreated from within the program.

The primary improvement to *TGRAF* will be to improve performance so that extractions and painting are done within reasonable time. The flow field properties could be displayed in various ways including shaded, vector displays, and combinations of these. Hidden line and surfaces could be eliminated and iso-surface display, and other advanced rendering techniques could be implemented.

B-2.12 Batch Mode TGRAF

A non-graphics variant of the *TGRAF* program has been written to provide a batch equivalent of the *TGRAF* graphics application. The non-graphics application, dubbed *TNOGRAF* uses the script file capability of *TGRAF* to produce PostScript plot files. The *TNOGRAF* program is run with the command:

% **tnograf** *case.bn scriptfile*

where *case.bn* is a dataset containing the *TGRAF* dataset and *scriptfile* is a file containing a *TGRAF* script program.

B-3 PLOT3D DATASETS

The datasets to be used for **PLOT3D** post processing are AELP3DG and AELP3DQ. Appropriate manuals should be consulted for further instructions on how to use PLOT3D.

B-4 DATA FOR 2D PLOTTING

There are many datasets created with data that is expected to make two dimensional graphs of relevant information. These files include FT10 which contains data for surface corner point aerodynamic quantities; FT11 which contains sectional properties summary; FT12 which contains information for forces and moments summary; and FT14 which contains information for sectional properties plot data. All these files are in ASCII format and can be printed or edited to suit the particular utility that is used to make 2D plots. Again the user should refer to the appropriate manual for those utilities available to him to make 2D plots.

Appendix C– BOUNDARY LAYER OUTPUT FILES

This appendix describes the output files produced in conjunction with a coupled boundary layer full potential solution invoked by the **$BOU** input parameter (inputs can be found in section 4–5.6). This keyword invokes a boundary layer analysis using Boeing's `A411` differential boundary layer analysis code in the *infinite swept tapered wing approximation.* Surface velocities computed by the inviscid TranAir solution are used to drive a boundary layer analysis on a conical surface boundary layer grid. The boundary layer analysis provides transpiration terms h to a TranAir boundary condition of the form:

$$\vec{W} \cdot \hat{n} = h \tag{C.1}$$

on every network corner point which has been defined to be part of a boundary layer "rib."

The TranAir post-processor `fdout` will generate an additional file called `BLGGP` containing boundary layer solution parameters at network corner points which are included in the boundary layer analysis. The data provided in this file includes boundary layer thickness, δ^*, momentum thicknesses, $\theta_{11}, \theta_{13}, \theta_{31}$, and θ_{33}, skin friction magnitudes and vectors *etc.* These are the boundary layer solution quantities which have been interpolated from the boundary layer solution grid to the network corner points and have been appropriately scaled according to the infinite swept tapered wing approximations. This file should be brought back to your workstation by using `rcp` to copy it from the Y-MP to the workstation.

In addition to these files, two additional boundary layer post processing programs may be run which use files created by `fdsol` and `A411` as input. These programs create plot files containing boundary layer solution information, boundary layer profile data and estimates of integrated skin friction and profile drag. The post-processors and files they create include:

- `Tsgpgl`: Integrates skin friction drags and estimates profile drags using a Squire Young drag formula. Creates a ggp file `BLCDF` with corresponding drag information. Also writes such information to standard out.

- `Tcwngvp`: Creates a ggp file containing velocity profile information on a file called `BLPRF` for plotting. This file may be *HUGE* if you have a lot of boundary layer ribs.

Files Used Between TranAir and Boundary Layer Analysis (BLGL)

The coupling of TranAir to the boundary analysis code is very loose and most communication between TranAir and the boundary layer analysis is accomplished through a number of boundary layer files. It is recommended that these be collected together under the file name `$CASE.blar` (for boundary layer archive) in your job script. These files are typically quite large and most of them are Cray unformatted FORTRAN files. The archive format is a convenient way to work with a group of files. Thus, they should generally remain on the Cray Y-MP.

To get a table of contents of file names stored in the archive, log onto the Y-MP and go to a directory where the `$CASE.blar` file resides and type:

```
ar t $CASE.blar
```

To extract all files from the archive file, type:

```
ar x $CASE.blar
```

To extract one or more files, type:

```
ar x $CASE.blar filename filename etc.
```

The list of files which should be contained in `$CASE.blar` is described in table C.1 and table C.2. The basic communications process consists of feeding information about surface velocities from TranAir to the Infinite Swept Tapered Wing Preprocessor `BLGL`. Then `BLGL` creates input files for the Boundary Layer Analysis code `A411`. `A411` solves the boundary layer equations and stores solution information on a database file. TranAir reads some files created by `BLGL` as well as the boundary layer database files and computes transpirations. Most users will only be concerned with the files described in the table C.1. (In particular, the first four files.) The other files are described for completeness.

File Name	File Type	Contents
BLGLSH	ASCII	Listing of Shock Locations On Rib.
A411OU	ASCII	Boundary Layer Solver Output File, Upper Surface.
A411OL	ASCII	Boundary Layer Solver Output File, Lower Surface.
BLGLOUT	ASCII	Boundary Layer Infinite Swept Wing Preprocessor Output.
A411BLU	UNFORMATTED	Upper Surface Boundary Layer Solution Database.
A411BLL	UNFORMATTED	Lower Surface Boundary Layer Solution Database.
BLGLDAU	UNFORMATTED	Input Parameters Controlling Boundary Layer Analysis for Upper Surface.
BLGLDAL	UNFORMATTED	Input Parameters Controlling Boundary Layer Analysis for Lower Surface.
BLGLDE	UNFORMATTED	Input Parameters Controlling Boundary Layer Grid Generation.

Table C.1: Files Contained In The `$CASE.blar` Archive

File Name	File Type	Contents
NXZCP	UNFORMATTED	Communications File From TranAir to BLGL.
A411INU	UNFORMATTED	Communications File from BLGL to A411 For Boundary Layer Analysis on the Upper Surface.
A411INL	UNFORMATTED	Communications File from the BLGL to A411 For Boundary Layer Analysis on the Lower Surface.
CREFU	UNFORMATTED	Communications File From BLGL to TranAir For the Upper Surface.
CREFL	UNFORMATTED	Communications File From BLGL to TranAir For the Lower Surface.
NA4T9U	UNFORMATTED	Reserved by A411.
NA4T9L	UNFORMATTED	Reserved by A411.
NA4T8U	UNFORMATTED	Reserved by A411.
NA4T8L	UNFORMATTED	Reserved by A411.
ATCHLU	UNFORMATTED	Attachment Line Solution Database For The Upper Surface.
ATCHLL	UNFORMATTED	Attachment Line Solution Database For The Lower Surface.
NA4T0U	UNFORMATTED	Velocity Profiles On Upper Surface, If Requested.
NA4T0L	UNFORMATTED	Velocity Profiles On Lower Surface, If Requested.
BLPF	UNFORMATTED	TranAir Boundary Layer Solution on Network Corner Points.

Table C.2: Binary Files Contained In The `$CASE.blar` Archive

References

[1] A502 User's Manual – PAN AIR Technology Program for Solving Potential Flow about Arbitrary Configurations, Volumes I and II, D6-49930-TN, 1982.

[2] Ruppert, P. B.; Johnson, F.; Bussoletti, J.; Samant, S.; and Young, D.: TRANAIR Computer Code (Theory Document). NASA Contract Report NAS2-12513, The Boeing Company, 1989.

[3] Ruppert, P. B.; Bussoletti, J. E.; Johnson, F. T.; Sidwell, K. W.; Rowe, W. S.; Samant, S. S.; SenGupta, G.; Weatherill, W. H.; Burkhart, R. H.; Everson, B. L.; Young, D. P.; and Woo, A. C.: A New Approach to the Solution of Boundary Value Problems Involving Complex Configurations. Computational Mechanics – Advances and Trends, Ahmed K. Noor, ed., The American Society of Mechanical Engineers, New York, 1986, p. 49.

[4] Samant, S. S.; Bussoletti, J. E.; Johnson, F. T.; Burkhart, R. H.; Everson, B. L.; Melvin, R. G.; Young, D. P.; Erickson, L. L.; Madson, M. D.; and Woo, A. C.: TRANAIR: A Computer Code for Transonic Analyses of Arbitrary Configuration. AIAA Paper 87-0034, Jan. 1987.

[5] Everson, B. L.; Bussoletti, J. E.; Johnson, F. T.; Samant, S. S.; Erickson, L. L.; and Madson, M. D.: TRANAIR and its NAS implementation. Paper Presented at the NASA Conference on Supercomputing in Aerospace. NASA Ames Research Center, March 1987.

[6] TRANAIR Computer Code (Theory Document). NASA Contract Report NAS2-11851, Boeing Military Airplane Company, 1987.

[7] Bussoletti, J. E.; Johnson, F. T.; Sidwell, K. W.; Everson, B. L.; and Young, D. P.: EM-TRANAIR: A Computer Program for the Solution of Maxwell's Equations in Three Dimensions: Volume 1, Theory Manual. Boeing Military Airplane Development AFWAL-TR-87-3082-vol.-1, 1987.

[8] Samant, S. S.; Bussoletti, J. E.; Johnson, F. T.; Melvin, R. G.; and Young, D. P.: Transonic Analysis of Arbitrary Configurations Using Locally Refined Grids. Proceedings of the 11th International Conference on Numerical Method in Fluid Dynamics, 1988.

[9] Young, D. P.; Melvin, R. G.; Bieterman, M. B.; Johnson, F. T.; Samant, S. S.; and Bussoletti, J. E.: A Locally Refined Rectangular Grid Finite Element Method. Applied Mathematics Technical Report SCA-TR-108-R1, Boeing Computer Services, Seattle, Washington, 1989.

[10] Young, D. P.; Melvin, R. G.; Johnson, F. T.; Bussoletti, J. E.; Wigton, L. B.; and Samant, S. S.: Application of Sparse Matrix Solvers as Effective Preconditioners. Boeing Computer Services Engineering and Scientific Services Technical Report ETA-TR-99, 1988. Also in SIAM Journal on Scientific and Statistical Computing, vol. 10, no. 6, 1989, pp. 1186-1199.

[11] Bussoletti, J. E.; Johnson, F. T.; Young, D. P.; Melvin, R. G.; Burkhart, R. H.; Bieterman, M. B.; Samant, S. S.; and SenGupta, G.: TRANAIR Technology: Solutions for Large PDE Problems. Solution of Super Large Problems in Computational Mechanics, J. H. Kane and A. D. Carlson, eds., Plenum Press, New York, 1989, pp. 95-124.

[12] Johnson, F. T.; Samant, S. S.; Bieterman, M. B.; Melvin, R. G.; Young, D. P.; Bussoletti, J. E.; and Madson, M. D.: Application of TRANAIR Rectangular Grid Approach to Aerodynamic Analysis of Complex Configurations. AGARD-CP-464, 1989, pp. 21.1-21.12.

[13] Melvin, R. G.; Bieterman, M. B.; Young, D. P.; Johnson, F. T.; Samant, S. S.; and Bussoletti, J. E.: Local Grid Refinement for Transonic Flow Problems. Proceedings of the Sixth International Conference on Numerical Methods in Laminar and Turbulent Flow, C. Taylor, P. Gresho, R. L. Sani, and J. Häuser, eds., Pineridge Press, vol. 6, part 1, 1989, pp. 939-950.

[14] Bieterman, M. B.; Bussoletti, J. E.; Hilmes, C. L.; Johnson, F. T.; Melvin, R. G.; Samant, S. S.; and Young, D. P.: Solution Adaptive Local Rectangular Grid Refinement for Transonic Aerodynamic Flow Problems. Report ECA-TR 126, Boeing Computer Services, Seattle, Washington, 1989. Proc. 1989 GAMM Conference on Numerical Methods in Fluid Mechanics, Delft, Netherlands, September 1989 in Notes on Numerical Fluid Mechanics, vol. 29, Vieweg Verlag, 1990.

[15] Young, D. P.; Melvin, R. G.; Bieterman, M. B.; Johnson, F. T.; and Samant, S. S.: Global Convergence of Inexact Newton Methods for Transonic Flow. Report ECA-TR-124-R1, Boeing Computer Services, Seattle, Washington, 1989.

[16] Chen, A. W.; Curtin, M. M.; Carlson, R. B.; and Tinoco, E. N.: TRANAIR Applications to Engine/Airframe Integration. AIAA Paper 89-2165, July 1989.

[17] Goodsell, A. M.; Madson, M. D.; and Melton, J. E.: TRANAIR and Euler Computations of a Generic Fighter Including Comparisons with Experimental Data. AIAA Paper 89-0263, Jan. 1989.

[18] Madson, M. D.; Carmichael, R. L.; and Mendoza, J. P.: Aerodynamic Analyses of Three Advanced Configurations Using the TRANAIR Full-Potential Code. NASA CP-3020, vol. 1, part 2, 1989, pp. 437-452.

[19] Tseng, W.; and Cenko, A.: TRANAIR Applications to Fighter Configurations. AIAA Paper 89-2220, July 1989.

[20] Walatka, P. P.; Buning, P. G.; Pierce, L.; and Elson, P. A.: PLOT3D User's Manual. NASA TM-101067, 1990.

[21] Snepp, D. K.; and Pomeroy, R. C.: A Geometry System for Aerodynamic Design. AIAA Paper 87-2902, Sept. 1987.

[22] Silicon Graphics, Inc., IRIS User's Guide, Volume I, Programming Guide Version 4.0, Document 007-1101-040, Mountain View, California, 1987.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Contractor Report

4. TITLE AND SUBTITLE
TranAir: A Full-Potential, Solution-Adaptive, Rectangular Grid Code for Predicting Subsonic, Transonic, and Supersonic Flows About Arbitrary Configurations—User's Manual

5. FUNDING NUMBERS
C NAS2-12513
WU 505-61-21

6. AUTHOR(S)
F. T. Johnson, S. S. Samant, M. B. Bieterman, R. G. Melvin, D. P. Young, J. E. Bussoletti, and C. L. Hilmes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Boeing Military Airplane Company
P. O. Box 3707, M/S 7K-06
Seattle, WA 98124-2207

8. PERFORMING ORGANIZATION REPORT NUMBER
A-90092

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Ames Research Center
Moffett Field, CA 94035-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-4349

11. SUPPLEMENTARY NOTES
Point of Contact: M. Madson, Ames Research Center, MS 227-2, Moffett Field, CA 94035-1000
(415) 604-3621

12a. DISTRIBUTION/AVAILABILITY STATEMENT
[redacted]
Subject Category – 02

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The TranAir computer program calculates transonic flow about arbitrary configurations at subsonic, transonic and supersonic freestream Mach numbers. TranAir solves the nonlinear full potential equations subject to a variety of boundary conditions modeling wakes, inlets, exhausts, porous walls, and impermeable surfaces. Regions with different total temperature and pressure can be represented. The user's manual describes how to run the TranAir program and its graphical support programs.

14. SUBJECT TERMS
Subsonic flow, Transonic flow, Supersonic flow, Full potential, Three-dimensional compressible flow, Arbitrary configuration, User's documentation, Computer program

15. NUMBER OF PAGES
204

16. PRICE CODE

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

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

NASA-Langley, 1992