NASA/CR—2006-213890

NPSS Multidisciplinary Integration and Analysis

Edward J. Hall, Joseph Rasche, Todd A. Simons, and Daniel Hoyniak
Rolls-Royce Corporation, Indianapolis, Indiana

March 2006

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2006-213890

NPSS Multidisciplinary Integration and Analysis

Edward J. Hall, Joseph Rasche, Todd A. Simons, and Daniel Hoyniak
Rolls-Royce Corporation, Indianapolis, Indiana

Prepared under Contract NAS3–98003, Task 5

National Aeronautics and
Space Administration

Glenn Research Center

March 2006

Acknowledgments

The authors would like to express their appreciation to the following people who contributed to this program: Vinod Nagpal, N&R Engineering, and Shantaram Pai, NASA Glenn Research Center, for their contributions related to the NESSUS code and probabilistics calculations in general; Al Magnusson and Diane Pourier, ICEM CFD Engineering, for their work regarding the CAPRI implementation and the CGNS testing; and John K. Lytle, Computational and Interdisciplinary Systems Office of the NASA Glenn Research Center, for continued support of this task under the Numerical Propulsion System Simulation (NPSS) project. Principal investigator and Rolls-Royce Program Manager for this program was Edward J. Hall. The NASA Project Manager was Donald Van Drei.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Preface

This report was prepared by Edward J. Hall, Joseph Rasche, Todd A. Simons, and Daniel Hoyniak of Rolls-Royce Corporation, Indianapolis, IN. The work was performed under NASA Contract NAS3-98003 from March 2000 to January 2002. Principal investigator and Rolls-Royce Program Manager for this program was Edward J. Hall. The NASA Project Manager was Donald Van Drei.

Contents

1 SUMMARY **1**

2 INTRODUCTION **3**

2.1 Role of Modeling in Engine Development 3

2.2 NASA High Performance Computing and Communications Program 3

2.3 NPSS Multidisciplinary Integration and Analysis 4

2.4 Multidisciplinary Analysis Approach 5

2.4.1 Aerodynamic Analysis 5

2.4.2 Structural Analysis 6

2.4.3 Multidisciplinary Analysis 6

3 MULTIDISCIPLINARY ANALYSIS **11**

3.1 Introduction 11

3.2 Aerodynamic Analysis with ADPAC 12

3.2.1 Grid Generation 15

3.2.2 CFD Wrapper Script 15

3.2.3 Airfoil Deflection Interpolation 17

3.2.4 Results 19

3.3 Introduction of NPSS_PHASE 1 AND 2 ANSYS macros 21

3.3.1 NPSS_PHASE1.MAC 25

3.4 NPSS_PHASE2.MAC 30

3.5 Automation of 3D CAD 32

3.6 Example Results 35

3.6.1 AE3007 Type III Fan 35

3.6.2 Low Cost Core Compressor (LCC) Rotor 35

3.6.3 Advanced Small Turboshaft Compressor (ASTC) Rotor 1 35

3.7 Evaluation of Reduction in Design Time 38

4 PROBABILISTIC ANALYSIS **41**

4.1 Introduction 41

4.2 Tip Gap Study 42

5 CAPRI Implementation **51**
5.1 Introduction to CAPRI . . . 51
5.2 CAPRI Implementation . . . 53

6 CGNS and ADPAC **55**
6.1 Description of CGNS . . . 55
6.2 Overview of CGNS for ADPAC . . . 55
6.3 ADPAC Data in a CGNS File . . . 56
6.3.1 Portability Across Platforms . . . 56
6.3.2 ADPAC Maximum Array Dimension . . . 57
6.3.3 ADPAC Standard Input File, *case*.input . . . 57
6.3.4 ADPAC Boundary Data File, *case*.boundata . . . 58
6.3.5 ADPAC Mesh File, *case*.mesh . . . 60
6.3.6 Body Force File, *case*.bf.# . . . 60
6.3.7 ADPAC Standard Output File, *case*.output . . . 61
6.3.8 ADPAC Plot Files . . . 61
6.3.9 ADPAC Restart Files . . . 62
6.3.10 ADPAC Convergence Files . . . 62
6.3.11 ADPAC Image Files . . . 63
6.4 Addition of iterative or time-accurate data, rigid body and arbitrary grid motion 63
6.4.1 The BaseIterativeData_t data structure under the CGNSBase_t data structure: . . . 63
6.4.2 The ZoneIterativeData_t data structure under the Zone_t data structure: 65
6.4.3 The RigidGridMotion_t data structure under the Zone_t data structure: 66
6.4.4 The ArbitraryGridMotion_t data structure under the Zone_t data structure: 67
6.5 Initial Proposal for Addition of the StructuredLevel_t node to CGNS . . . 70
6.6 Conclusion . . . 71

7 CONCLUSIONS **77**

A FILE SPECIFICATIONS FOR MULTIDISCIPLINARY ANALYSIS **83**
A.1 File Specification for Blade Point File . . . 83
A.2 File Specification for Blade Data File . . . 83
A.3 Example File: . . . 84

B NPSS PHASE 1 ANSYS MACRO **87**

C NPSS STAGE 2 ANSYS MACRO **127**

List of Figures

2.1 Conceptual illustration of Stage 1 (hot-to-cold coordinate conversion) of the NPSS multidisciplinary analysis system. 8

2.2 Conceptual illustration of Stage 2 (cold-to-warm coordinate conversion) of the NPSS multidisciplinary analysis system. 9

3.1 Stage 1 (hot-to-cold coordinate conversion) of the NPSS multidisciplinary analysis system. 13

3.2 Stage 2 (cold-to-warm coordinate conversion) of the NPSS multidisciplinary analysis system. 14

3.3 Midspan section of H-type mesh used in CFD-based aerodynamic analysis. . . . 16

3.4 ANSYS surface element with four nodes. 18

3.5 Predicted relative Mach number contours at the midspan location of the ASTC fan, LCC Rotor 9, and AE3007 Type III fan. 20

3.6 AE3007 Type III fan hot running and cold shape (near tip). 22

3.7 AE3007 Type III fan blade hot running shape at 80% speed. 23

3.8 Steady blade aerodynamic loading at 75% span of the AE3007 Type III fan. . . 24

3.9 Example of point file layout from aero design system. 27

3.10 Example of mesh using solid96 20 noded brick elements in hot running position. 28

3.11 Plot showing the cold morphed mesh, stressed hot running position and the lines representing the solid model location of the hot running airfoil. 31

3.12 Example of local permutation of stream section points. 33

3.13 Solid model from Pro/E via IGES and mesh created automatically on solid model. 34

3.14 Illustration of defelection analysis for the AE3007 Type III Fan Rotor 36

3.15 Illustration of delection analysis for LCC Rotor. 37

3.16 Illustration of deflection analysis for the ASTC Rotor 1. 38

4.1 Schematic outline of tip clearance probabilistic analysis. 45

4.2 AE3007 Type III fan blade predicted 100% corrected speed pressure ratio versus corrected mass flow operating performance curves based on varying tip gap. . . 46

4.3 AE3007 Type III fan blade predicted 100% corrected speed adiabatic efficiency versus corrected mass flow operatig performance curves based on varying tip gap. 47

4.4 AE3007 Type III fan blade predicted 100% corrected speed adiabatic efficiency versus corrected mass flow operating performance curves based on varying tip gap and intersections with constant pressure ratio over corrected flow line. . . . 48

4.5 Quadratic regression obtained from computation of effect of tip gap on predicted adiabatic efficiency. 49

4.6 Predicted probability density function for fan rotor adiabatic efficiency as a function of varying distribution scenarios for rotor tip gap. 50

6.1 ADF file mapping definition of the BaseIterativeData_t data structure. 72

6.2 ADF file mapping definition of the ZoneIterativeData_t data structure. 73

6.3 ADF file mapping definition of the RigidGridMotion_t data structure. 74

6.4 ADF file mapping definition of the ArbitraryGridMotion_t data structure. . . . 75

Nomenclature

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

Roman Symbols

i ... first grid index of numerical solution
j ... second grid index of numerical solution
k ... third grid index of numerical solution
p ... pressure
r ... radius or radial coordinate
t ... time
u ... velocity
x ... axial coordinate
y ... vertical or normal coordinate
y^+ ... boundary layer inner variable
z ... Cartesian coordinate normal to (x, y) plane

ADPAC ... ADvanced Propulsion Analysis Code
ANSYS... Commercial finite element solver code
CAS ... Computational Aero Sciences
CFD ... computational fluid dynamics
FEM ... Finite element method
FORTRAN ... Formula Translation coding language
HPCC ... High Performance Computing and Communication
M ... Mach number
MG ... levels of multi-grid
MULAC ... NASA mesh generation code
NESSUS ... NASA probabilistic analysis code
NPSS ... Numerical Propulsion System Simulation
PERL ... Practical Extraction and Report Language

PLOT3D ... post-processing 3-D visualization tool
R ... gas constant
R_c ... pressure ratio
Re ... Reynolds Number
S-A ... Spalart-Allmaras turbulence model
T ... temperature
U_{tip} ... rotor tip speed
V ... velocity
V_r ... velocity in the cylindrical coordinate system radial direction
V_x ... velocity in the cylindrical coordinate system axial direction
V_θ ... velocity in the cylindrical coordinate system circumferential direction

Greek Symbols

β ... flow angle
ζ ... third generalized coordinate
η ... efficiency
η ... second generalized coordinate
θ ... tangential coordinate
ξ ... first generalized coordinate
ρ ... density
ω ... rotational speed

Subscripts

$[\,]_{i,j,k}$ ... grid point index of variable
$[\,]_{max}$ ... maximum value
$[\,]_{min}$ ... minimum value
$[\,]_p$ ... pressure side
$[\,]_r$ ... pertaining to the radial (r) cylindrical coordinate
$[\,]_s$ ... suction side
$[\,]_t$ ... total (stagnation) value
$[\,]_x$ ... pertaining to the axial (x) cylindrical coordinate
$[\,]_\theta$ ... pertaining to the circumferential (θ) cylindrical coordinate
$[\,]_\infty$ ... freestream value

Chapter 1

SUMMARY

The objective of NASA Contract NAS3-98003 Task 5 was to enhance the Numerical Propulsion System Simulation (NPSS) core capabilities by expanding its reach into the high fidelity multidisciplinary analysis area. The intent was to investigate techniques to integrate structural and aerodynamic flow analyses, and provide a benchmark by which performance enhancements to NPSS can be baselined. The task order was based on the following 7 sub-tasks:

- Develop a high fidelity analysis to calculate the effects on performance of: variations in tip clearance, uncertainty in manufacturing tolerances, guide vane scheduling and the effects of rotational speed on the hot running geometry.
- Enable the numerical calculation of blade deformations between the Advanced Ducted Propfan Analysis (ADPAC) aero analysis and an ANSYS structural analysis.
- Evaluate the potential to use the CAPRI library for geometry analysis in a multidisciplinary simulation.
- Determine whether the CGNS standard can represent ADPAC I/O data. If it cannot, make recommendations on what else is needed.
- Incorporate probabilistic analysis based on ADPAC aerodynamic solutions. NASA provided an executable of the NESTEM/NESSUS probabilistic analysis program and user manuals. The necessary input/output data required to couple aerodynamic, structural, and probabilistic analysis programs were linked to provide the necessary interactions between codes.
- Report performance measurements on the HPCC testbeds. Expected measurements were analysis speedup and scalability.
- Estimate the impact of the new methods on the reduction in engine design or development time relative to a 1997 baseline when implemented in the design process. This improvement can be expressed in terms of a system, subsystem, component, and part.

Chapter 2

INTRODUCTION

Engineering design of advanced gas turbine aircraft propulsion systems and land-based power generation systems is at a crossroads. Gas turbine technology has matured sufficiently that without significant technology breakthroughs, the ultimate performance of these engines will not improve significantly. This forces manufacturers to focus on improvements in delivered value to their customer in the form of reduced acquisition and life-cycle costs. Major gas turbine manufacturers are now instituting processes that will allow them to design and deliver engine systems in relatively short time periods (2 years from concept to engine certification) to be more responsive to industry trends and demands. This represents a significant reduction in engine development time and cost.

2.1 Role of Modeling in Engine Development

A key element in the ability to accomplish a two year engine development program lies in the ability to accurately model and assess candidate engine designs. Modeling and simulation can replace, to a large extent, the extensive test and rebuild programs traditionally used to develop a successful engine. This strategy builds extensively on technology advancements afforded through modern computing architectures, as well as sophisticated simulation methodologies designed to capture the essential physics associated with the engine development goals.

2.2 NASA High Performance Computing and Communications Program

The NASA High Performance Computing and Communications (HPCC) Program is a dedicated task to advance the national computing infrastructure and computing technology capability. The mission of the HPCC program is to accelerate the development of high-performance computers and networks and encourage their use by government and private industry. NASA's Numerical Propulsion System Simulation (NPSS) is one of five tasks within the Computational Aerospace Sciences (CAS) project under the HPCC

program. NPSS is a concerted effort by NASA Glenn Research Center, the aerospace industry, and academia to develop an advanced engineering environment consisting of an integrated collection of software programs for the analysis and design of aircraft engines and, eventually, space transportation components. The purpose of NPSS is to dramatically reduce the time, effort and expense necessary to design and test propulsion systems. This feat is accomplished by generating sophisticated computer simulations of an aerospace object or system, thus permitting an engineer to "test" various design options without having to conduct costly and time-consuming real-life tests. The ultimate goal of NPSS is to create a "numerical test cell" that enables engineers to create complete engine simulations overnight on cost-effective computing platforms. Using NPSS, engine designers will be able to:

- Analyze different parts of the engine simultaneously
- Perform different types of analysis simultaneously (e.g. aerodynamic and structural)
- Perform analysis faster, better and cheaper

All of these items are consistent with the CAS goal of accelerating the development and availability of high-performance computing hardware and software to the United States aerospace community. In turn, NPSS addresses the growing need for simulation-based testing in the aerospace engine development process.

2.3 NPSS Multidisciplinary Integration and Analysis

The objective of this task order (NASA Contract NAS3-98003 Task 5) was to enhance the Numerical Propulsion System Simulation (NPSS) core capabilities by expanding its reach into the high fidelity multidisciplinary analysis area. The intent was to investigate techniques to integrate structural and aerodynamic flow analyses, and provide benchmark by which performance enhancements to NPSS can be baselined.

This report describes efforts under this task based on the following 7 sub-tasks:

- Develop a high fidelity analysis to calculate the effects on performance of: variations in tip clearance, uncertainty in manufacturing tolerances, guide vane scheduling and the effects of rotational speed on the hot running geometry.
- Enable the numerical calculation of blade deformations between the Advanced Propulsion Analysis CODE (ADPAC) aero analysis and an ANSYS structural analysis.
- Evaluate the potential to use the CAPRI library for geometry analysis in a multidisciplinary simulation.
- Determine whether the CGNS standard can represent ADPAC I/O data. If it cannot, make recommendations on what else is needed.

- Incorporate probabilistic analysis based on ADPAC aerodynamic solutions. NASA provided an executable of the NESTEM/NESSUS probabilistic analysis program and user manuals. The necessary input/output data required to couple aerodynamic, structural, and probabilistic analysis programs were linked to provide the necessary interactions between codes.

- Report performance measurements on the HPCC testbeds. Expected measurements were analysis speedup and scalability.

- Estimate the impact of the new methods on the reduction in engine design or development time relative to a 1997 baseline when implemented in the design process. This improvement can be expressed in terms of a system, subsystem, component, and part.

Together, these items combine to form a significant new capability for the NPSS system that can eventually be exploited to accomplish the stringent demands of modern propulsion system development programs.

2.4 Multidisciplinary Analysis Approach

In order to address the desired goals above, a simulation methodology was developed that couples high fidelity aerodynamic and structural dynamic analysis codes. The coupling methodology was developed such that structural forces resulting from aerodynamic loads were imported directly into the airfoil structural analysis. Conversely, deflections resulting from the combined aerodynamic and centrifugal loads on the structure were fed back into the aerodynamic simulation. Iterative update of the data elements results in a converged solution satisfying both aerodynamic and structural static equilibrium.

2.4.1 Aerodynamic Analysis

The aerodynamic analysis code selected for this study was the ADvanced Propulsion Analysis Code (ADPAC) [1] [2] [3] [4] [5]. The ADPAC code is the result of a twelve-year partnership between Rolls-Royce and NASA combining over 2.3 million dollars in development and applications experience. Aerodynamic, thermodynamic, and acoustic technology delivered through the ADPAC system have impacted gas turbine, aircraft, and propeller industries, including direct applications to the Lockheed C-130, Embraer RJ-145, Saab 2000, and Lockheed Joint Strike Fighter aircraft. ADPAC predictions of aerodynamic/acoustic data for the Saab 2000 aircraft guided product developments leading to a patent award for propeller noise reduction. The ADPAC code currently stands alone as the only code in the world with a demonstrated whole engine turbomachinery modeling capability.

ADPAC is intended to provide a highly adaptable computing framework for a wide variety of aerodynamic analysis tasks. The primary purpose of the ADPAC code is to

provide a rapid, three-dimensional flow analysis capability for multistage turbomachinery (fan, compressor, turbine) flows. ADPAC employs a flexible structured multiple-blocked mesh system, and proven numerics based on Runge-Kutta time-marching procedures.

ADPAC features a number of important aerodynamic modeling technologies including:

- The proven Spalart-Allmaras [6] turbulence model, known to predict adverse pressure gradient flows with high accuracy.
- Models representing the effects of deterministic unsteadiness in steady state analysis of multistage compressor/turbine flows.
- Modeling capabilities for flow injection/removal, useful for simulating the effects of compressor bleed flows, blade boundary layer control through flow aspiration, seal cavity/leakage flows, and endwall treatments such as slots, grooves, and endwall embedded flow turning devices.
- Multi-processor parallel computing based on the Message Passing Interface (MPI) interprocessor communication protocol. ADPAC is also fully compatible with all known parallel machine queuing systems.

ADPAC is an evolutionary product of computing technologies developed at Rolls-Royce (Indianapolis), and is therefore subject to US export restrictions.

2.4.2 Structural Analysis

Structural analysis for the multidisciplinary system was developed around the ANSYS analysis code. The ANSYS 5.7 Product Suite includes time-tested, industry-leading applications for structural, thermal, mechanical, and electromagnetic analyses, as well as solutions for transient impact analysis. ANSYS software solves for the combined effects of multiple forces, accurately modeling combined behaviors resulting from "multiphysics" interactions. The software also features advanced nonlinear material simulation capabilities.

2.4.3 Multidisciplinary Analysis

The approach taken for the multidisciplinary system was developed to address certain design requirements for turbommachinery blading. Specifically, the system is intended to provide a rapid means for the design engineer to assess both aerodynamic and structural performance simultaneously, including the combined effects resulting from airfoil static structure deflections.

The deflection analysis is divided into two stages. The overall multidisciplinary simulation methodology is flowcharted in Figure 2.1 and Figure 2.2. Figure 2.1 illustrates

what is referred to as the hot-to-cold coordinate conversion process (deflection analysis). This analysis takes the as-designed airfoil shape and backs out the hot running deflections to deduce the cold desired manufactured shape. The Stage 1 conversion does not require an iterative update of the aerodynamic loads on the blade since the hot running shape is fixed and is specified as input.

The second stage of the overall analysis provides the deflection analysis and aerodynamic prediction for off-design operation. This process, referred to as Stage 2, is illustrated in Figure 2.2. The Stage 2 analysis begins with the cold manufactured airfoil coordinates, and iteratively computes the desired running shape based on the specified operating parameters (rotational speed, inlet/exit aerodynamic boundary conditions, etc.). The Stage 2 analysis requires iterative updates of both the structural and aerodynamic analysis results, demanding a distinct coupling between disciplines.

Details of the multidisciplinary simulation and the assessment of probabilistic analysis in the simultion strategy are provided in the sections that follow. Chapter 3 describes the aerodynamic and structural simulation techniques in detail. Chapter 4 describes the probabilistic aerodynamic analysis and tip clearance assessment study. Chapter 5 describes the evaluation and investigation of the CAPRI CAD geometry handling package. Chapter 6 describes the evaluation of the CGNS standard for CFD analysis and the potential for implementing this library in the ADPAC code. Chapter 7 summarizes conclusions from this study and provides recommendations for future research.

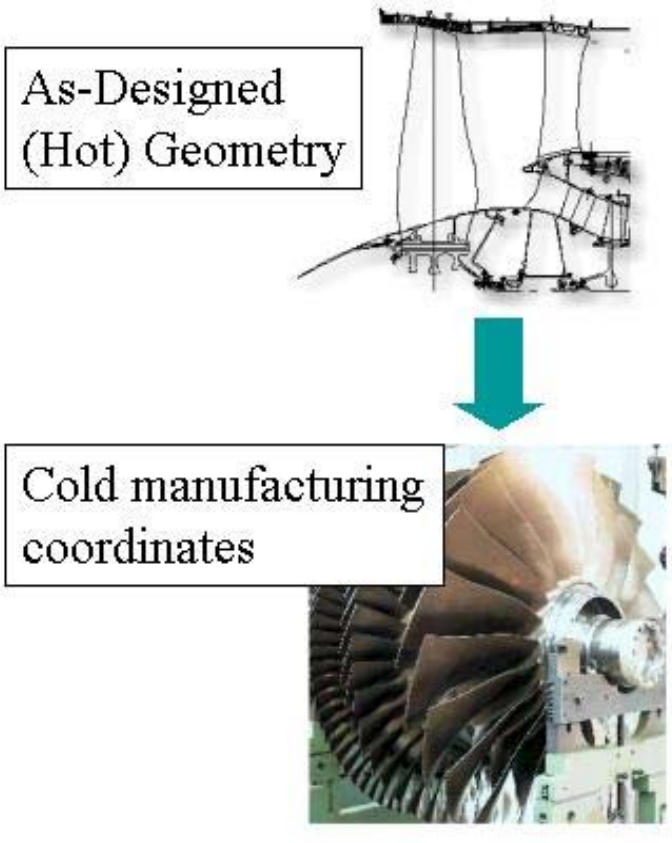

Figure 2.1: Conceptual illustration of Stage 1 (hot-to-cold coordinate conversion) of the NPSS multidisciplinary analysis system.

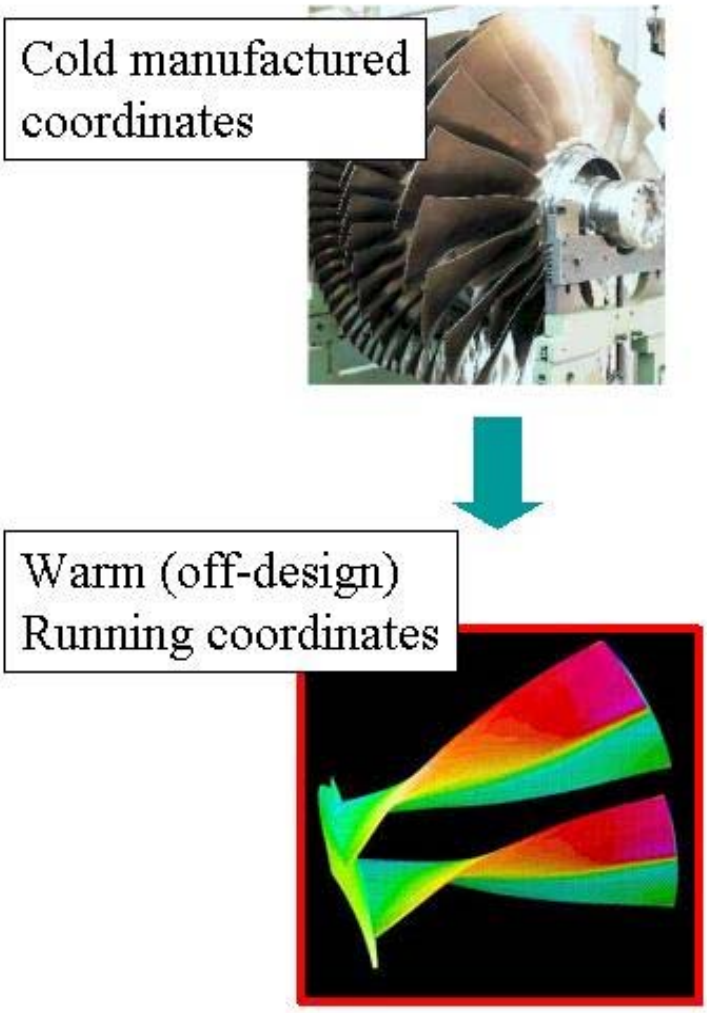

Figure 2.2: Conceptual illustration of Stage 2 (cold-to-warm coordinate conversion) of the NPSS multidisciplinary analysis system.

Chapter 3

MULTIDISCIPLINARY ANALYSIS

3.1 Introduction

In this chapter, a detailed description of the multidisciplinary analysis systems and the individual components that make up that system are described. Implicit in the definition of the multidisciplinary analysis system is the requirement for both a static structural and aerodynamic analysis capability. In this study, separate solvers were used to satisfy these two requirements. Aerodynamic predictions were based on the ADPAC [5] aerodynamic analysis code. Static structural analysis was accomplished by using the ANSYS commercial analysis code. Details of the implementation of these codes are provided in the sections that follow.

The approach taken for the multidisciplinary system was developed to address certain design requirements for turbomachinery blading. Specifically, the system is intended to provide a rapid means for the design engineer to assess both aerodynamic and structural performance simultaneously, including the combined effects resulting from airfoil static structure deflections. The objectives defined for the multidisciplinary analysis were addressed through the development of a two stage static deflection analysis capability.

The overall multidisciplinary simulation methodology is flowcharted in Figure 3.1 and Figure 3.2. Figure 3.1 illustrates what is referred to as the hot-to-cold coordinate conversion process (deflection analysis). This analysis takes the as-designed airfoil shape and backs out the hot running deflections to deduce the cold desired manufactured shape. The Stage 1 conversion does not require an iterative update of the aerodynamic loads on the blade since the hot running shape is specified as input.

The second stage of the overall analysis provides the deflection analysis and aerodynamic prediction for off-design operation. This process, referred to as Stage 2, is illustrated in Figure 3.2. The Stage 2 analysis begins with the cold manufactured airfoil coordinates, and iteratively computes the desired running shape based on the operating parameters (rotational speed, inlet/exit aerodynamic boundary conditions, etc.). The Stage 2 analysis requires iterative updates of both the structural and aerodynamic analysis results, demanding a distinct coupling between disciplines.

Both Stage 1 and Stage 2 analyses were controlled through an ANSYS macro. These macros assumed that an instance of ANSYS is running and acting as a global controller through the ANSYS utilities allowing external code execution. This functionality can eventually be replaced in NPSS via a JAVA or C++ manager with code interfaces controlled via CORBA interfaces. This task did not focus on the mechanisms by which the capability is implemented in NPSS. Rather, this task focused on the details of the multidisciplinary analysis itself.

Separate sections are provided below describing both the aerodynamic and static structural analysis elements. Descriptions are provided, where needed, to illustrate the software mechanisms used to automate the procedures. Very little analysis code development was required for this task. The bulk of the effort focused on integrating the various elements of the analysis under a cohesive controlled procedure.

3.2 Aerodynamic Analysis with ADPAC

For the purposes of the multidisciplinary simulation, the equations of fluid dynamics are solved by ADPAC (ADvanced Propulsion Analysis Code) [1] [2] [4] [5]. The ADPAC code is a general purpose turbomachinery aerodynamic design analysis tool, which has undergone extensive development, testing, and verification. A brief description of the theoretical basis for the ADPAC analysis follows.

The ADPAC analysis solves a time-dependent form of the three-dimensional Reynolds-averaged Navier-Stokes equations using a proven time-marching numerical formulation. The code employs proven numerics based on a finite-volume, explicit multi-grid Runge-Kutta time-marching solution algorithm. Steady-state flows are obtained as the time-independent limit of the time-marching procedure. Several steady-state convergence acceleration techniques (local time stepping, implicit residual smoothing, and multi-grid) are available to improve the overall computational efficiency of the analysis. The ADPAC code permits the use of a multiple-blocked mesh discretization, which provides extreme flexibility for analyzing complex geometries. ADPAC offers the choice of three different turbulence models: the algebraic Baldwin-Lomax turbulence model, the one-equation Spalart-Allmaras turbulence model, and a two-equation k-R turbulence model.

Automating the aerodynamic analysis with ADPAC involves three steps, pre-processing, solving the equations for fluid dynamics, and post-processing. Pre-processing involves grid generation and setting up boundary conditions and input decks. H-type grid topologies with an H-type tip clearance grid were used for the mesh. Boundary information for the analysis is stored in the blade_data.dat file and subsequently placed in the ADPAC boundary data (*.boundata*) file.

Post-processing involved reading the ADPAC output flow files and generating surface pressures and temperatures in a format suitable for use in the ANSYS structural

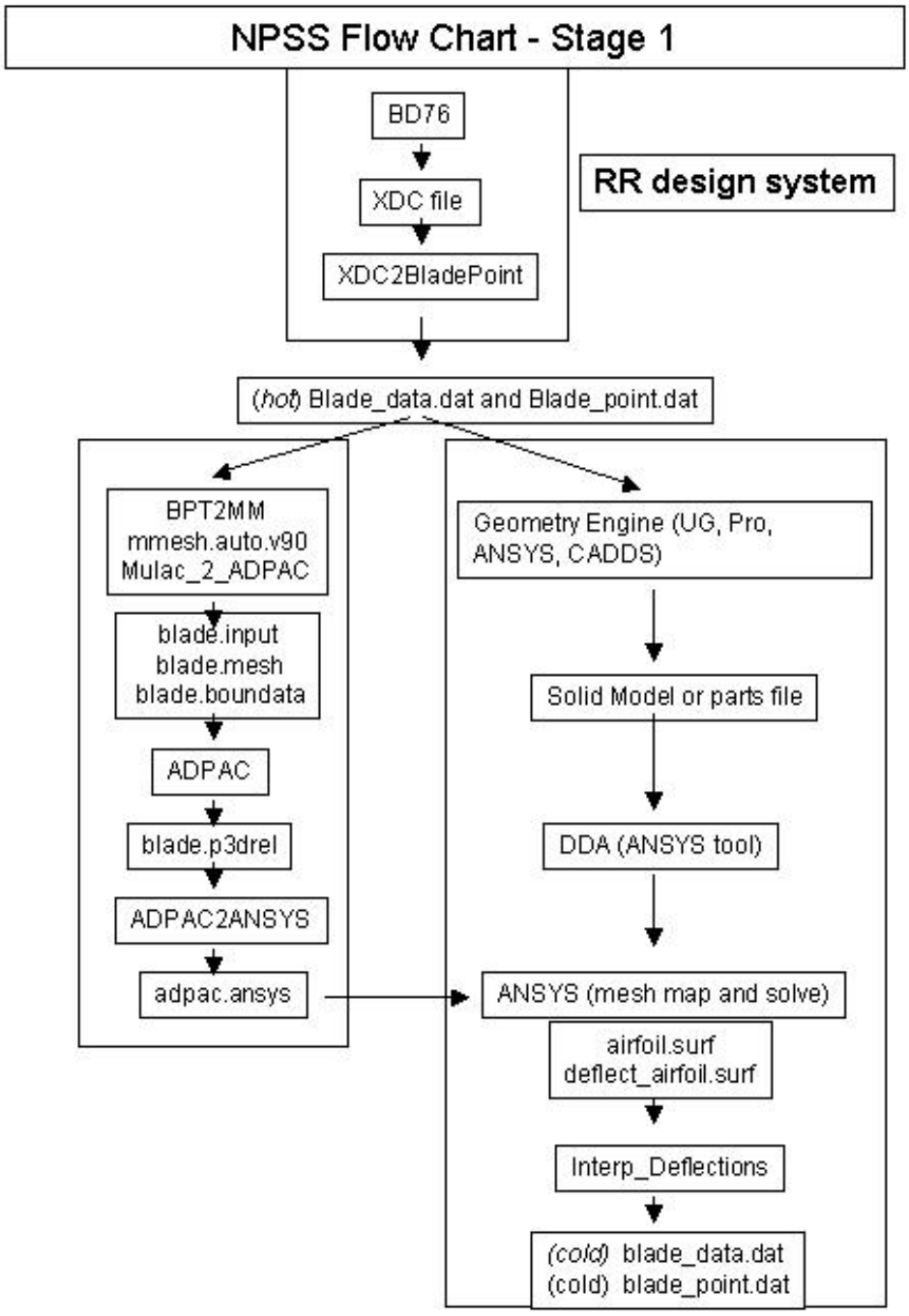

Figure 3.1: Stage 1 (hot-to-cold coordinate conversion) of the NPSS multidisciplinary analysis system.

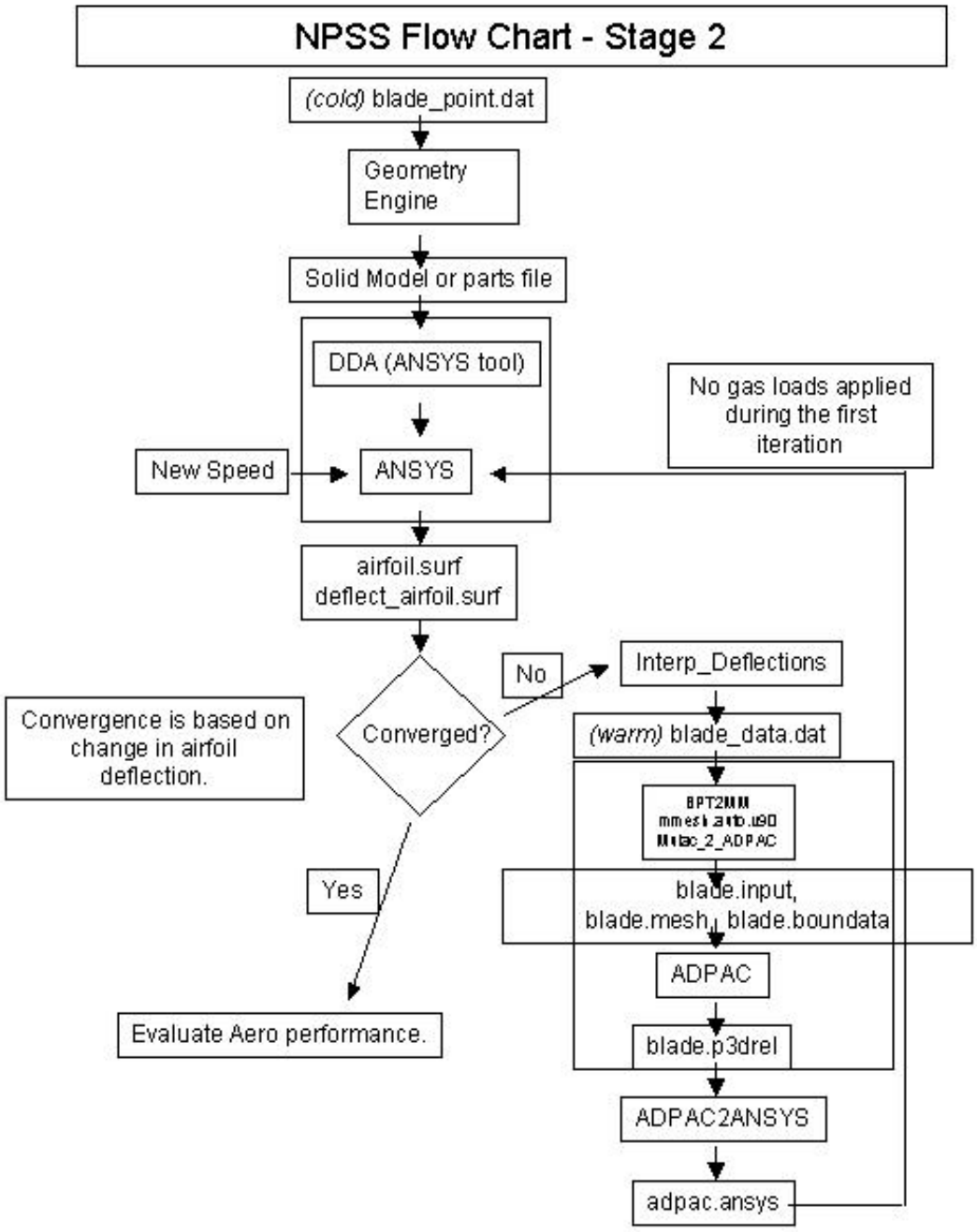

Figure 3.2: Stage 2 (cold-to-warm coordinate conversion) of the NPSS multidisciplinary analysis system.

analysis. The ANSYS code employs the predicted airfoil surface pressure and temperature to determine the blade structural and thermal loading. To generate the surface coordinates, the post processor must relocate the pressure side information in a periodic fashion because of the discontinuous airfoil representation present in the H-grid topology. The post-processor also computes the static pressure and static temperature from the conservative variables in the flow field.

3.2.1 Grid Generation

Since the multidisciplinary analysis was intended to be highly automated, mesh generation for the aerodynamic analysis was required to be simple and robust. Early mesh generation work involved a stand alone FORTRAN program to generate an H mesh with three-dimensional grid smoothing using the Poisson equations. Grid generation under this approach used an iterative scheme that was time consuming. This method was later replaced with the NASA grid generation program MULAC (version 9.0). MULAC is a three-dimensional algebraic grid generator used to generate grids for multistage turbomachinery. Both approaches used cubic splines to mesh the airfoil surface geometry, however the algebraic interior point specification approach in MULAC is considerably faster than the iterative scheme.

Two utilities were used in conjunction with the MULAC grid generator. The first utility converted the user-defined blade_data.dat file to a MULAC input deck. This utility, BPT2MM (Blade PoinT to Mulac Mesh), is a short FORTRAN code. The second utility converted the MULAC mesh to an ADPAC mesh file. This utility, Mulac_2_ADPAC, was also written in FORTRAN. The second utility does several additional tasks: an H-type tip clearance grid is added, the H grid is smoothed with elliptical equations, and the second utility also generates the boundary data input file for ADPAC.

A typical grid employs 300,000 grid points. The main block of the H grid has a grid resolution of 125 x 53 x 45. Figure 3.3 shows a midspan cross-section of the grid on a fan.

3.2.2 CFD Wrapper Script

All of the steps involved with the CFD analysis can be automated by using a single script. PERL was selected because it was the best tool suited to perform the functions required for pre- and post-processing. PERL stands for Practical Extraction and Report Language. PERL is designed for parsing text files and handling subprocesses involving input and output. The PERL script, cfd_wrapper.x, performs a number of functions in the CFD analysis. The script coordinates input and output for the grid generation and input and boundary input files for ADPAC. The script also runs ADPAC and post-processes the data as well.

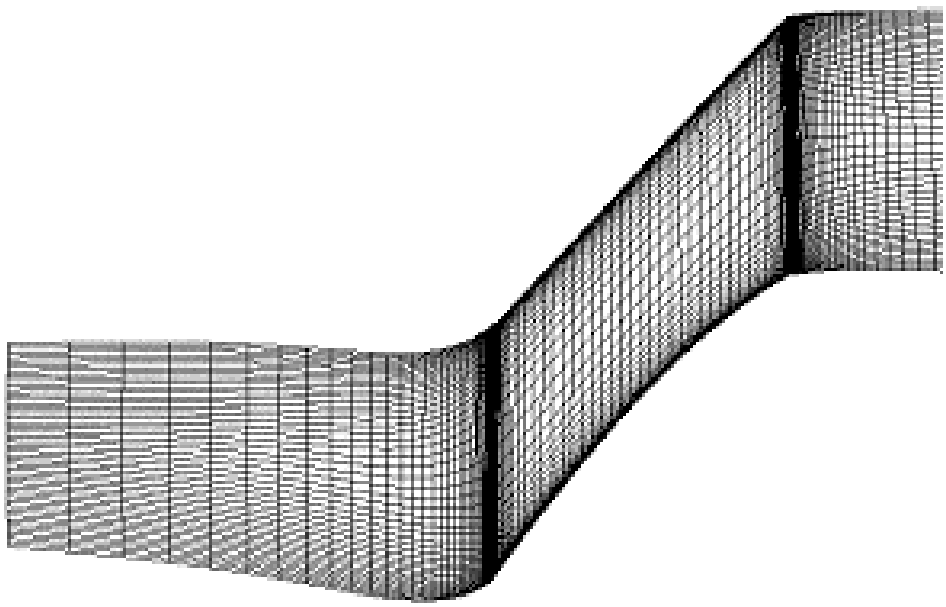

Figure 3.3: Midspan section of H-type mesh used in CFD-based aerodynamic analysis.

3.2.3 Airfoil Deflection Interpolation

The airfoil deflection data are interpolated onto the cloud of points defining the airfoil surface. Interpolation is performed using isoparametric shape functions. This approach is a flexible approach that can perform either linear or nonlinear interpolation. Currently the ANSYS mesh uses 20 node bricks to mesh the cold blade. ANSYS only writes out the corner nodes on the surface of the blade. The ANSYS output also contains the connectivity of the surface nodes, which form four noded shells. ANSYS also outputs the deflections for each node.

The interpolation program reads the surface quad mesh and deflections. A simple ANSYS surface element is illustrated in Figure 3.4. Deflections are interpolated for each point defining the airfoil using isoparametric shape functions. This approach is a flexible and robust method to interpolate deflections. By using different shape functions, deflections can be interpolated from shell meshes or solids. Any element shape, including hybrid elements, can be used in the interpolation without a significant change in the algorithm.

The global coordinate system (x,y,z) is related to a natural, or local coordinate system. The local coordinate system is used for each element, in this case (u,v) which range from (-1,+1). The relationships are made by

$$x = \sum_{i=1}^{4} h_i x_i \tag{3.1}$$

$$y = \sum_{i=1}^{4} h_i y_i \tag{3.2}$$

$$z = \sum_{i=1}^{4} h_i z_i \tag{3.3}$$

where for the two-dimensional, four noded quadrilateral cells,

$$h_1 = \frac{1}{4}(1-u)(1-v) \tag{3.4}$$

$$h_2 = \frac{1}{4}(1+u)(1-v) \tag{3.5}$$

$$h_3 = \frac{1}{4}(1+u)(1+v) \tag{3.6}$$

$$h_4 = \frac{1}{4}(1-u)(1+v) \tag{3.7}$$

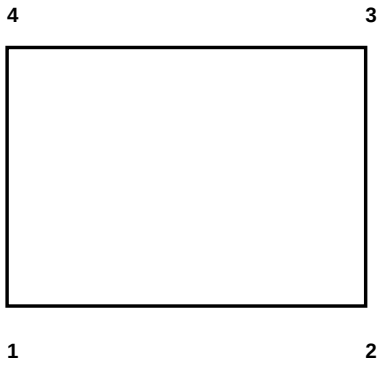

Figure 3.4: ANSYS surface element with four nodes.

A Downhill Simplex method due to Nelder and Mead is used to find the local coordinates of the point in the airfoil definition. The Downhill Simplex method is a multi-dimensional search routine that is accurate and robust.

Once the local coordinates (u,v) have been determined, the displacement is determined by

$$\xi = \sum_{i=1}^{4} h_i \xi_i \tag{3.8}$$

$$\eta = \sum_{i=1}^{4} h_i \eta_i \tag{3.9}$$

$$\zeta = \sum_{i=1}^{4} h_i \zeta_i \tag{3.10}$$

where ζ, η, ξ are the displacements in the x, y, and z directions, respectively.

3.2.4 Results

Several test cases were used to evaluate the CFD procedures developed for this work, as well as the method for computing cold shapes. The test cases include the first stage fan in the Advanced Small Turboshaft Compressor (ASTC), the last rotor in the Low Cost Compressor (LCC), the AE 3007 Type III fan, and the AE 3007 Type IV fan. Contours of predicted relative Mach number at the midspan location are shown for several cases in Figure 3.5.

Comparisons were made for the AE-3007 Type 3 fan blade as the cold and 80% part speed blade shapes were available from an earlier analysis. The method used to obtain the earlier solution was different from the current approach for generating cold and part speed blade shapes. The earlier method was based on an in-house finite element package called STRATA. The earlier model employed shell elements as opposed to the solid elements employed in the current approach. The airfoil aerodynamic loads were derived directly from the design system (results based on stream-line methods), not three-dimensional CFD solvers. In addition, the previous finite element method, while including non-linear effects, did not account for non-linearity caused by large deflections. More importantly, the earlier analysis modeled the platform and stalk as well as the airfoil. For moderately sized fan blades, applying the boundary conditions on the stalk correctly would have a more significant impact than the non-linearity or aerodynamic loading.

Figure 3.6 shows the tip section of the AE3007 Type 3 fan blade in the hot running shape at design speed (100% speed) , the cold shape from earlier design methods, and the cold shape computed by the current method. The agreement in the cold shape of the blade is very good, especially considering the differences in the detail of the analysis.

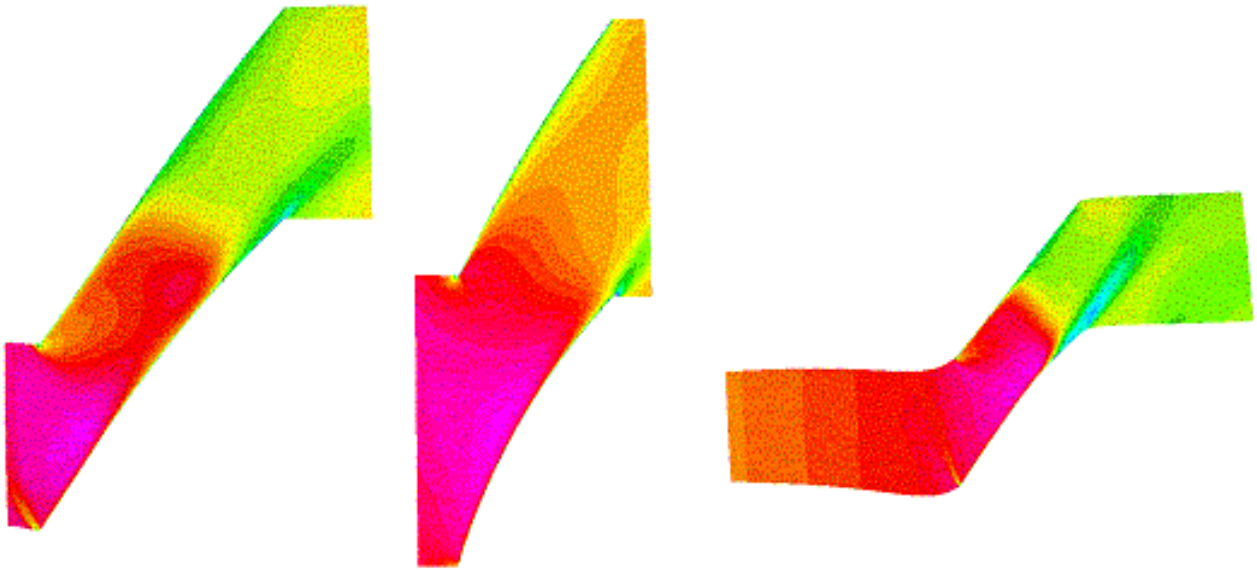

Figure 3.5: Predicted relative Mach number contours at the midspan location of the ASTC fan, LCC Rotor 9, and AE3007 Type III fan.

Some difference between the cold shapes were expected, primarily because of the blade root boundary condition used in the NPSS analysis. It is reasonable to fix the blade at the hub for compressors and small airfoils, but preferable to fix the boundary condition beneath the platform on larger airfoils or fan blades. The current work only generates the blade geometry with a fixed platform. Future work could include driving the geometry with Pro/E, UG or CADDS, which would also generate platform and stalk geometry. This would improve the structural boundary conditions as well.

Fan blades present the simplest boundary conditions for the CFD solver. The inlet CFD boundary conditions for the fan are uniform for different speeds, while the boundary conditions for rotors in a compressor or turbine are impacted by the stators both upstream and downstream of the rotor.

Figure 3.7 compares the tip section of the AE3007 Type III fan blade at 80% speed. At this lower rotational speed, the differences are significantly smaller and good agreement is achieved between methods.

To illustrate the importance of deflection on aerodynamic performance, Figure 3.8 shows a comparison of the steady blade loading at 80% with the correct airfoil shape as computed with the NPSS procedure, the steady loading with the 100% airfoil shape simply run at 80% speed. The steady blade loading is shown at the 75% radial span location. The x axis is the normalized axial location, while the y axis A is the static pressure normalized by the inlet total pressure. The variation in the incidence angle is visible in the steady loading. Blade loading differs near the aft end of the airfoil.

3.3 Introduction of NPSS_PHASE 1 AND 2 ANSYS macros

This section describes the driving system for the NPSS_PHASE 1 and NPSS_PHASE 2 ANSYS macros. These instructions include some general information covering the execution of the macros and the inputs/files required for successful execution.

The macros are compatible with ANSYS 5.7 and above and utilize the newly developed *moper command inside ANSYS which allows for efficient mapping of externally developed data, such as CFD data. The macros will run on any hardware platform that ANSYS 5.7 is installed on with little to no modifications to the macros. The calls to the external programs, inter_deflect.x and cfd_wrapper.x, are the only commands that will need to be modified. The attached copy of the macros (see Appenix B and Appendix C) show the hardwired location of the inter_deflect.x and cfd_wrapper.x calls, these commands can be shorted to just the inter_deflect.x and cfd_wrapper.x if the commands are made available on the unix/pc system.

It is the developer's intent to track the important variables with the ANSYS finite element program. Since there is no limit on the number of scalar parameters in ANSYS 5.7 and that the true gauge for the success of any airfoil is it structural/dynamic strength, the ANSYS database will be used to track the significant variables of the design, CFD or otherwise. The parameters from the CFD run are currently collected with the adpac.input

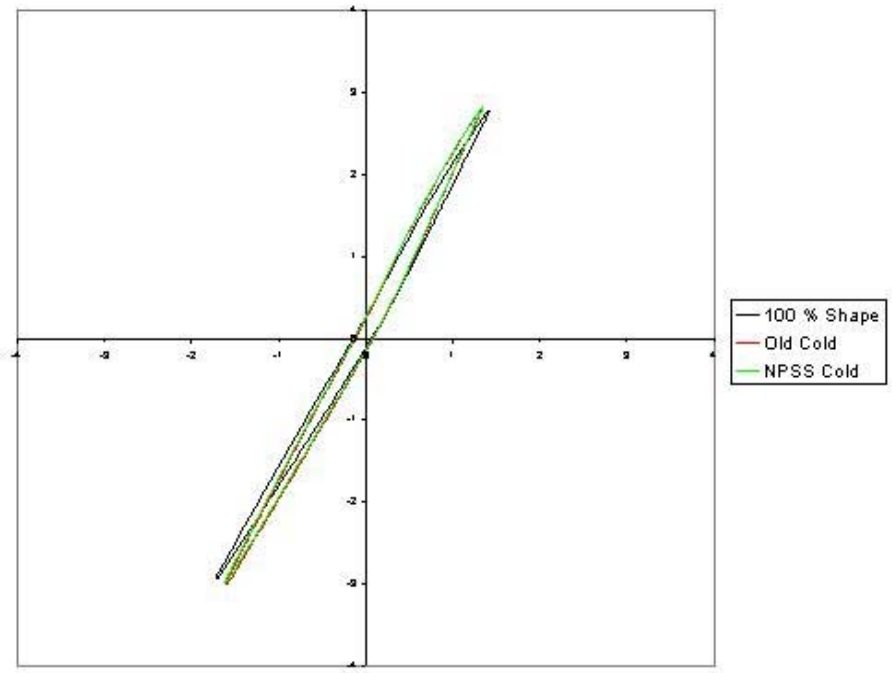

Figure 3.6: AE3007 Type III fan hot running and cold shape (near tip).

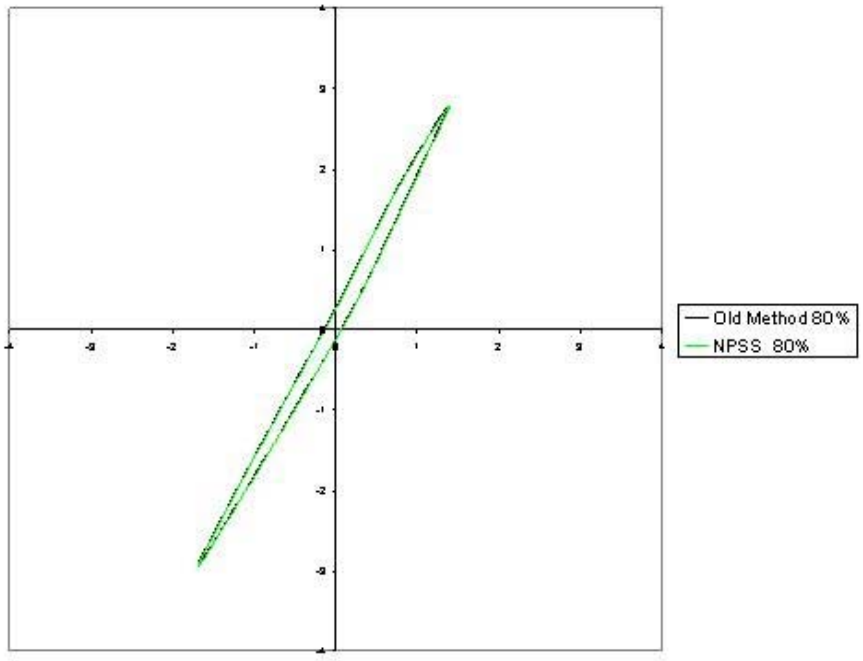

Figure 3.7: AE3007 Type III fan blade hot running shape at 80% speed.

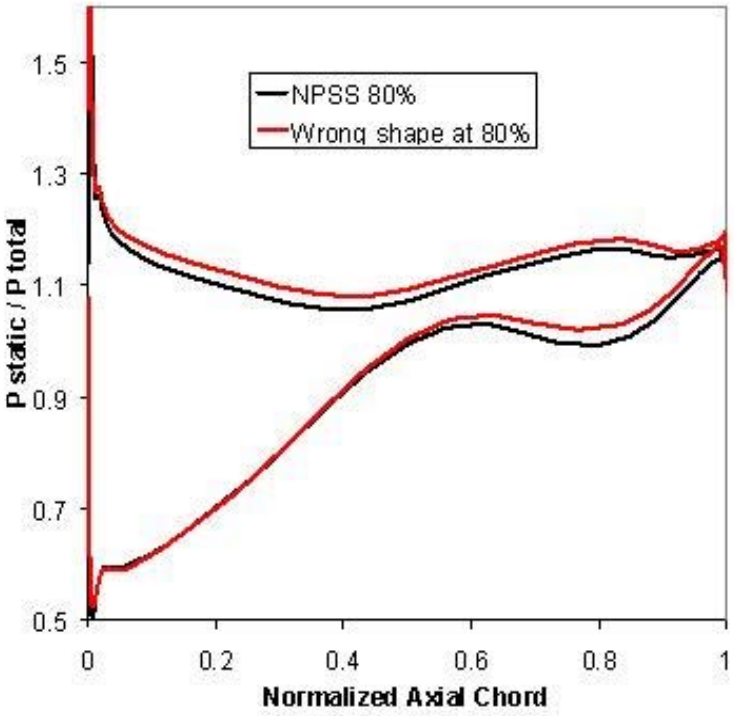

Figure 3.8: Steady blade aerodynamic loading at 75% span of the AE3007 Type III fan.

file generated by the cfd_wrapper.x. To input the parameters into ANSYS the external program should place the parameters in an ASCII file format with the following format:

```
par_a = 0.03
par_b= 4532
par_c = 3
```

etc.

The parameter names must be 8 characters or less with any duplication of parameter name replacing earlier stored values. The parameter file can be read into ANSYS with the /input,filename,ext command.

At the current state of development the solid model in phase 1 and phase 2 are created with the ANSYS program. Some exploration into driving a commercial 3D, parametrical, solid modelling package was covered under this contract. The CAD packages Pro/Engineer version 2000i and UG version 15 were investigated for possible front ends to the solid model creation. The technique necessary to access the 3D solid modelling technology will be discussed later in this chapter but will not go into CAD specific detail, only the technique will be discussed. It is left to the individual to set up their CAD system appropriately.

The point file definition of the airfoil was chosen as the format on which to pass design definition to the 3D solid modelling systems. The point file format was chosen in order to accommodate all known 3D solid modelling systems since all except point definition in the x, y, z format. This format along with the automation of CFD and physical analysis prediction allows for optimization and probabilistic design studies to be employed. Probabilistic design analysis investigation was covered by the current contract but optimization was not. Refer to the probabilistic analysis section (chapter 4, page 41) for more information.

The ANSYS macros for this report are set up to drive the process, driving the external codes such as interp_delfect.x and cfd_wrapper.x. It is recommended that this process not be changed, other than changing the /sys command format when incorporating into the NPSS Corba program.

3.3.1 NPSS_PHASE1.MAC

NPSS_PHASE1.mac takes the design points specified by the design system as well as the CFD flow data and creates a hot running position solid model inside the ANSYS finite element software. The gas pressures and metal temperatures from the CFD run are mapped onto the hot running position mesh and the mesh is morphed into its predicted cold shape through an iterative process.

The flow chart for phase 1 is shown in figure 3.1 in chapter 3 showing the process flow. The RR design system can be replaced with any design system as long as the system can produce the blade_data.dat and blade_point.dat file formats. The blade_data.dat is not used by the NPSS_PHASE1.mac or NPSS_PHASE2.mac macros and will not be discussed in this chapter, refer to chapter 6 for further discussion of the blade_data.dat format and content.

An example of the blade_point.dat file is shown below and has been edited to show relative features of the file:

```
    24     11    160
     -2.29470      -0.94252       7.24095
     -2.29614      -0.93870       7.24079
     -2.29680      -0.93472       7.24100
+++> Break in x,y and z coordinates to save on space
```

The first line of the blade_point.dat file contains three integers, the first is the number of blades in the stage, the second is the number of stream sections defining the airfoil and the third is the number of points per stream section. The number of points defining the airfoil can be derived from multiplying the second integer by the third integer to obtain the total number of points listed on the file. The order of the points must be consistent from section to section, the starting point for each section should start at the same tangency point from section to section and the number of points per section must stay constant.

The ANSYS formatting used to read the blade_point.dat file can be seen at the beginning of the NPSS_PHASE1.mac file shown in Appendix 2. (NPSS_PHASE1.MAC file, refer to the lines following the *vread commands for format used.)

The section of code in NPSS_PHASE1.mac from the first /prep7 command up to the vmesh,all command builds and meshes the solid model inside of ANSYS. This section of code can be replaced with coding that pulls in the solid model information from a CAD system. Either the commercial connection products offered by ANSYS, Inc could be used or IGES file format could be used to retrieve the solid model. The investigation into the automation of the 3D CAD systems Pro/E and UG will be discussed later in this chapter under the section entitled " Automation of 3D CAD".

The attached NPSS_PHASE1.MAC AND NPSS_PHASE2.MAC file only drive the ANSYS solid model engine. Figure A and Figure B show the point definition and solid model respectively.

The section of code in the NPSS_PHASE1.MAC file that goes from the vmesh,all command down to the " /com, SOLID MODEL AND MESH CREATION COMPLETED" command identify the pressure side, suction side, leading edge and trailing edge. The definition of this nodal components is necessary to facilitate the new *moper mapping command in ANSYS 5.7. If a CAD system is used to generate the solid

Figure 3.9: Example of point file layout from aero design system.

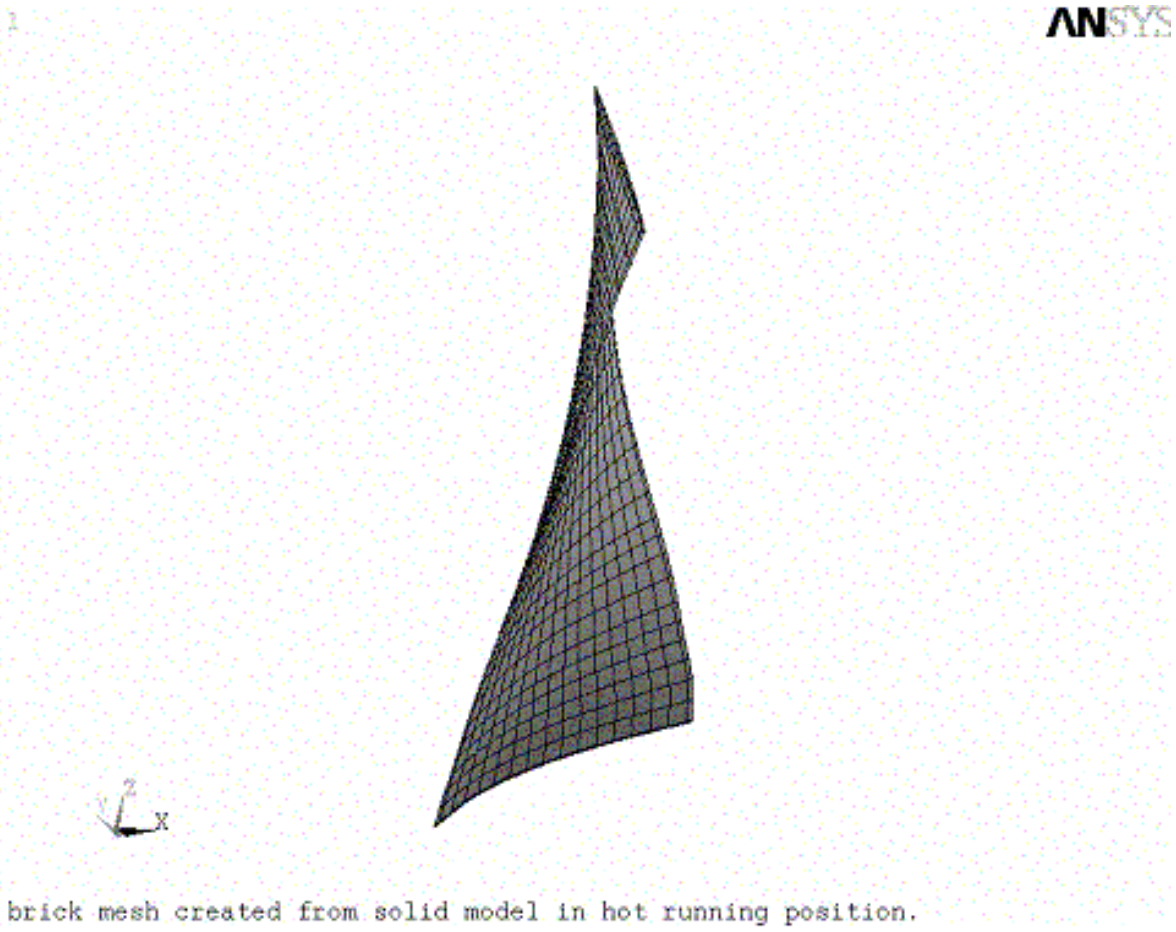

Figure 3.10: Example of mesh using solid96 20 noded brick elements in hot running position.

model and the CAD system does not allow tagging of surfaces, an alternate method of defining the nodal components will have to be employed.

After the solid model is meshed, a surface mesh of the airfoil is required in order to generate the airfoil.surf file. The airfoil.surf file is used in the interp_deflect.x program to define the original airfoil location. For NPSS_PHASE1.MAC the airfoil.surf represents the airfoil location in the hot running position. The section of code from the command et,2,22 down to *cfclose,airfoil,surf covers the surface meshing of the airfoil with ANSYS element type surf22 elements and the writing of the airfoil.surf file. The section of code from *cfopen to *cfclose covers the file format for airfoil.surf with the file format read statements following the *vwrite commands

The section of NPSS_PHASE1.MAC file from the *cfclose,airfoil,surf down to the solve command covers the mapping of temperatures and pressures from the adpac.ansys file which was generated previously with the first cfd_wrapper.x external call. Only the temperatures are applied to the finite element model at this point, so as to calculate the thermal growth at the hub section of the airfoil. The thermal growth calculation at the hub is required in order to calculate the appropriate displacements at the hub so as to eliminate the effect of a thermal pinch at the hub. The blade is simply supported at the hub with only the temperatures applied to the model. A structural analysis is performed to calculate the deflections induced by the applied temperatures, these deflections are then applied as boundary conditions, thus eliminating the effect of a thermal pinch at the hub. If a CAD system is used to model the entire blade, the same approach would be used at the blade/wheel attachment interface.

. The cfd_wrapper.x function is shown in Figure 2.1, "NPSS Flow chart - Stage 1" and encompasses BPT2MM, ADPAC and ADPAC2ANSYS utilities. Refer to the earlier sections of this chapter for more detail on the above-mentioned utilities.

The section of NPSS_PHASE1.MAC file from the second solve to the following solve command covers applying the calculated displacements for the thermal only solution, applying the calculated pressures and switching the analysis type from static to non-linear, large deflection effects included.

After the non-linear, large deflection analysis with the temperatures and pressures applied finishes; the macro calculates the deflected shape from the hot running position and takes the inverse of the deflected shape. The airfoil continues to untwist from the hot running position and it is the inverse of this continued untwisted that is used for the initial guess of the cold shape. The inverse deflections are used to morph the mesh to the initial guess of the cold geometry. The non-linear solutions is run again with its deflected shape compared to the original hot running position. If at this point, the deflected initial morph mesh resides within the tolerance specified the macro generates the deflect_airfoil.surf file that contains the deflections necessary to calculate the cold_blade_point.dat file.

If the initial guessed morphed mesh does not fall within the tolerance an iterative *DO loop is started with a maximum of 20 iterations being allowed. The nodal locations continue to be modified until the tolerance has been met. The parameter bltol is the convergence tolerance that is currently hardwired into the macro and is set to 0.005

inches. The bltol and the number of iterations can be modified by editing the NPSS_PHASE1.MAC file. The Appendix B "NPSS_PHASE1.MAC MACRO" lists the ANSYS commands.

The analysis time of 65 minutes is contributed to using the *do loop process to modify the nodal location from iteration to iteration. Utilizing the *vput command in ANSYS 5.7 could dramatically decrease the analysis time. The *vput command was not utilized for either phase of the npss program due to the risk of corrupting the ANSYS database with the *vput command. The *vput command will be investigated in future development of this process.

The files cold_blade_point.dat and cold_blade_data.dat files are created at the completion of the NPSS_PHASE1.mac macro. These files are used at the start of NPSS_PHASE2.mac macro.

The files needed to start the NPSS_PHASE1. MAC process are the following:

1) blade_data.dat for the hot running condition of interest. 2) blade_point.dat for the hot running condition of interest. 3) mater.mp file containing the ANSYS commands necessary to define the material inside of ANSYS. (An example is shown in appendix blah blah, "Example of material input deck" 4) adpac.input for rpm speed. 5) adpac.ansys for cfd point cloud x,y,z locations and pressure and temperature for each point.

Four blades designs were used to test the NPSS_PHASE1.MAC macro; LCC3, S42, Type 3 and Type 4. All four produced reasonable results, the Type 3 design was used to gauge the accuracy of the hot to cold calculations because of existing results for the Type 3 design. The comparison is shown in chapter 3.

3.4 NPSS_PHASE2.MAC

The NPSS_PHASE2.mac macro assumes that the phase 1 portion has been completed and there is a cold_blade_point.dat and cold_blade_data.dat file in the current working directory. The cfd_wrapper.x script is run to calculate the temperatures and pressures off of the cold_blade_data.dat file. The cfd_wrapper.x creates the adpac.ansys file which is read in following the completion of the cfd_wrapper.x scripts. The file offdesign_adpac.input contains the off design point rpm.

The commands from the "/input,offdesign_adpac,input" down to "vmesh,all" build the cold shape solid model inside of the ANSYS program. The same steps are completed for phase 2 as were completed for phase 1 to calculate the boundary conditions that negate the effect of thermal pinch at the hub of the airfoil. The commands from "vmesh,all" down to "ITN=20" cover setting up the boundary conditions and running the initial analysis run.

The commands from "ITN=20" down to "*ENDDO" run a *do loop that runs the cold mesh up to speed and applies the calculated pressures and temperatures until CFD and deformed mesh converge.

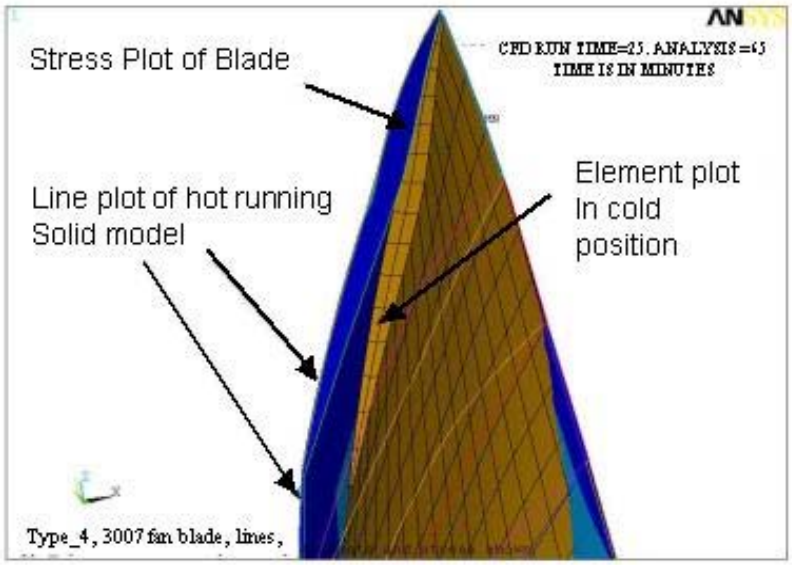

Figure 3.11: Plot showing the cold morphed mesh, stressed hot running position and the lines representing the solid model location of the hot running airfoil.

The commands from "*ENDDO" to the end of the macro generate the new_blade_data.dat and new_blade_point.dat which represent the airfoil in the off design point condition. Refer to Appendix B, "NPSS_PHASE2.MAC for command sequence.

3.5 Automation of 3D CAD

Pro/Engineer and UG were the only two parametric CAD packages investigated under the NPSS contract for possible automation of solid model creatios. It was deemed necessary to investigate these two common parametric CAD packages due to their strength in solid model creation and their robust parametric language. It should be noted that any CAD system that can run a history file or log file in batch mode and can create a solid model that is importable to ANSYS can be used.

The history file or what is called the trail file by Pro/Engineer was created by importing the point files created by ANSYS that split the points into at a pressure side and suction side. The points were read in to create datum points. These datum points were used to create datum curves which were used to create a lofted surface. The top and bottom of the lofted surface were closed off to create a solid model inside Pro/E. The rest of the geometry was created using standard Pro/E commands.

At the completion of the solid model, the trail file was copied over to a filed called runfile. The runfile could then be used to build up the solid model automatically inside of Pro/E as long as the number of points did not change or their relative position did not move excessively. Local permutations of the pressure side were possible but large scale movement of the stream sections, such as a cold shape, were not possible, refer to Figure 3.12 for an example of a rough local permutation of pressure side.

This problem is due to the fact that Pro/E records graphical pick location and not named component. By moving the stream section to the cold location, it was far enough away from the original graphical pick that it caused the trail file to error on the solid model creation of the airfoil.

There is a possible work around, it might be possible to replace the initial set of points after the model has been created with a different set of points and have the trial file work. This approach was not investigated due to time and budget constraints but will be investigated and documented if NPSS work is continued.

The local permutations of the points was possible as well as the automation of the conversion of the Pro/E solid model into an ANSYS database which could be meshed and analysed. See Figure E: Local permutation of stream sections, solid model from Pro/E via IGES and mesh shown.

The UG CAD system was not investigated to the same extent as Pro/E due to Rolls-Royce's working knowledge of the UG system. The UG system has a "grep" language that is similar to ANSYS macro language, this allows for the automation of the solid model creation. The automation and documentation of the UG grep process will be implemented and documented if further NPSS work is funded.

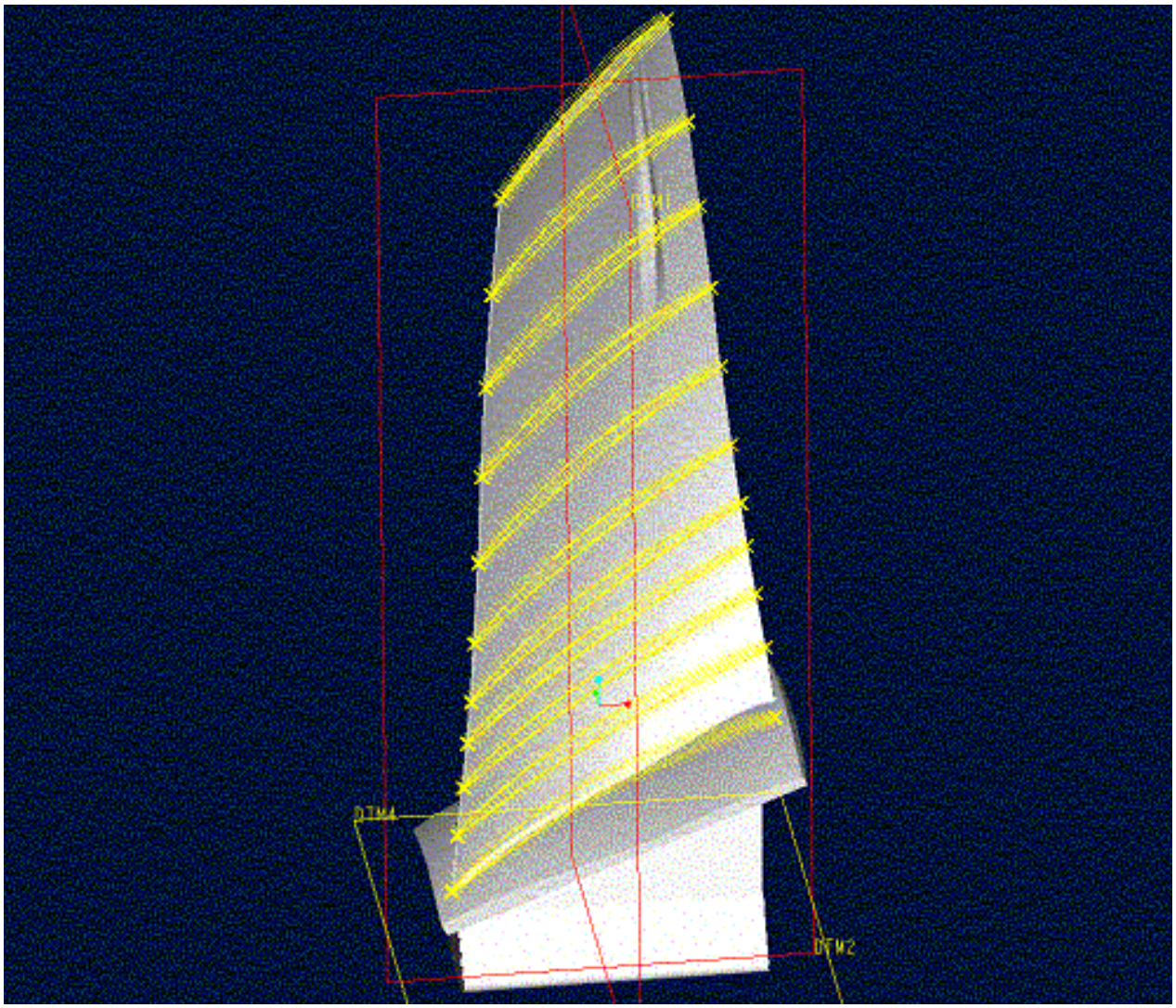

Figure 3.12: Example of local permutation of stream section points.

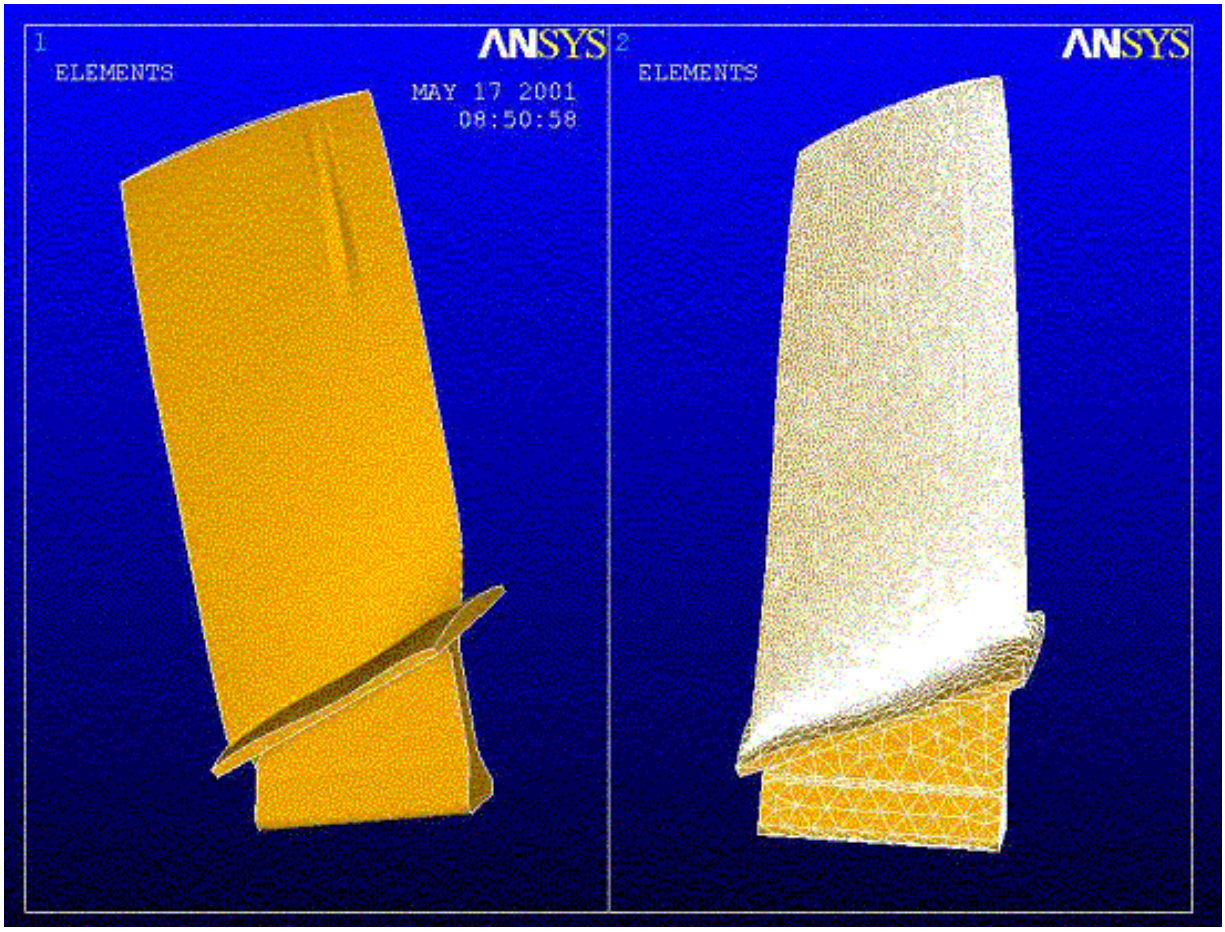

Figure 3.13: Solid model from Pro/E via IGES and mesh created automatically on solid model.

3.6 Example Results

In this section, results from the multidisciplinary analysis system are presented as a form of validation and demonstration. The system was applied to three compression system airfoils. In each case, both the Stage 1 and Stage 2 analyses were applied to the airfoil. To the extent possible, both the aerodynamic predictions and the deflection predictions were examined and compared to existing data and/or other trusted engineering calculations.

3.6.1 AE3007 Type III Fan

The AE3007 Type III fan test case was previously discussed and portions of the results from the multidisciplinary analysis were presented earlier in this chapter. Figure 3.14 serves as a summary of this test case. Illustrated on Figure 3.14 are representations of the AE3007 engine, a full rotation representation of the 3D CFD model, and comparisons between the various deflection analyses resulting from the application of the Stage 1 and Stage 2 analysis macros. One verification of the system that can be performed easily is to first predict the cold blade coordinates using the hot-to-cold analysis (Stage 1) and then attempt to redelfect the airfoil to the 100% speed operating condition using the cold-to-warm (Stage 2) analysis. Properly coded, the original hot shape and the post-Stage 2 blade shape at the same operating condition at 100% speed should be identical. This desired result was achieved in all test cases, and is illustrated in Figure 3.14.

3.6.2 Low Cost Core Compressor (LCC) Rotor

The low cost core compressor represents one of the new breed of highly loaded compressor designs intended to reduce cost in multistage aircraft gas turbine compressor through reduced stage count. The test airfoil used in this exercise was the rotor from the final stage of the LCC design. This geometry represents the smallest scale extreme of the airfoils tested. The LCC rotor is approximately 0.5 inches in span.

The LCC rotor was also tested using both the Stage 1 and Stage 2 analyses, and once again the deflection results were consistent with expectations. Figure 3.15 illustrates the results from the analysis for this rotor. The original hot running shape was reestablished following the Stage 1 and Stage 2 analyses at the 100% speed operating condition.

3.6.3 Advanced Small Turboshaft Compressor (ASTC) Rotor 1

Finally, the Stage 1 and Stage 2 analyses were applied to the first rotor from the Advanced Small Turboshaft Compressor (ASTC). The ASTC rotor represnts and extreme high Mach number design where airfoil shape is critical in terms of controlling shock structure aerodynamics. The ASTC rotor displays some unusua;l characteristics due to the high hub ramp angle and swept design. The airfoil delfects in the opposite sense

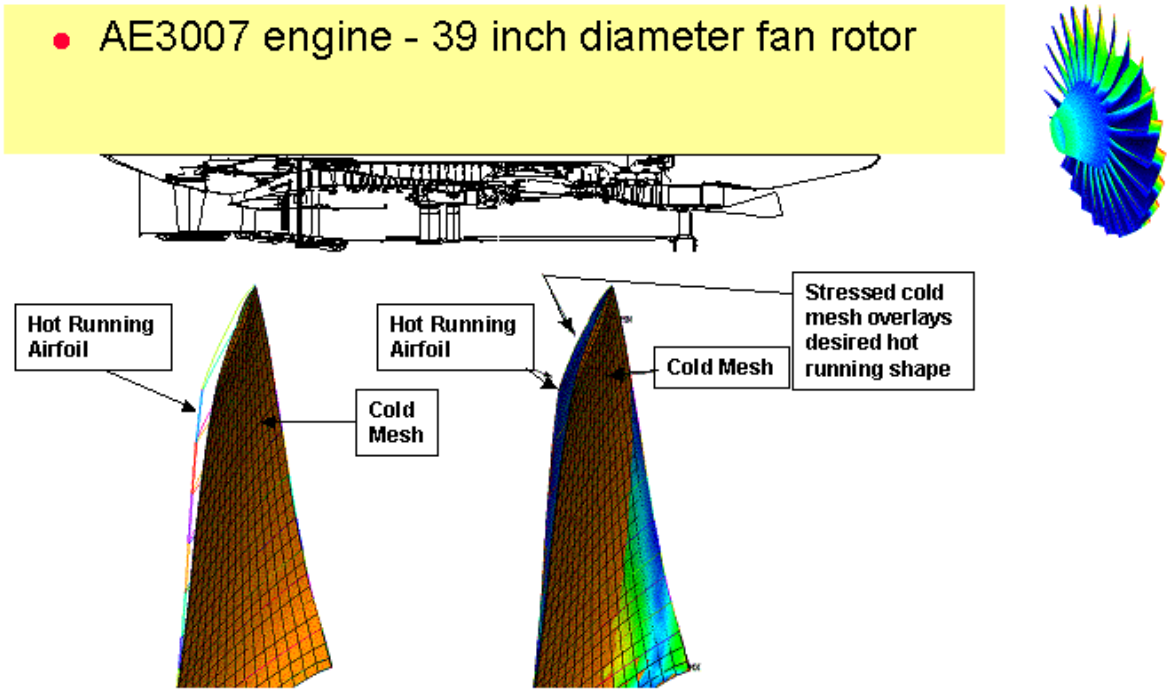

Figure 3.14: Illustration of defelection analysis for the AE3007 Type III Fan Rotor.

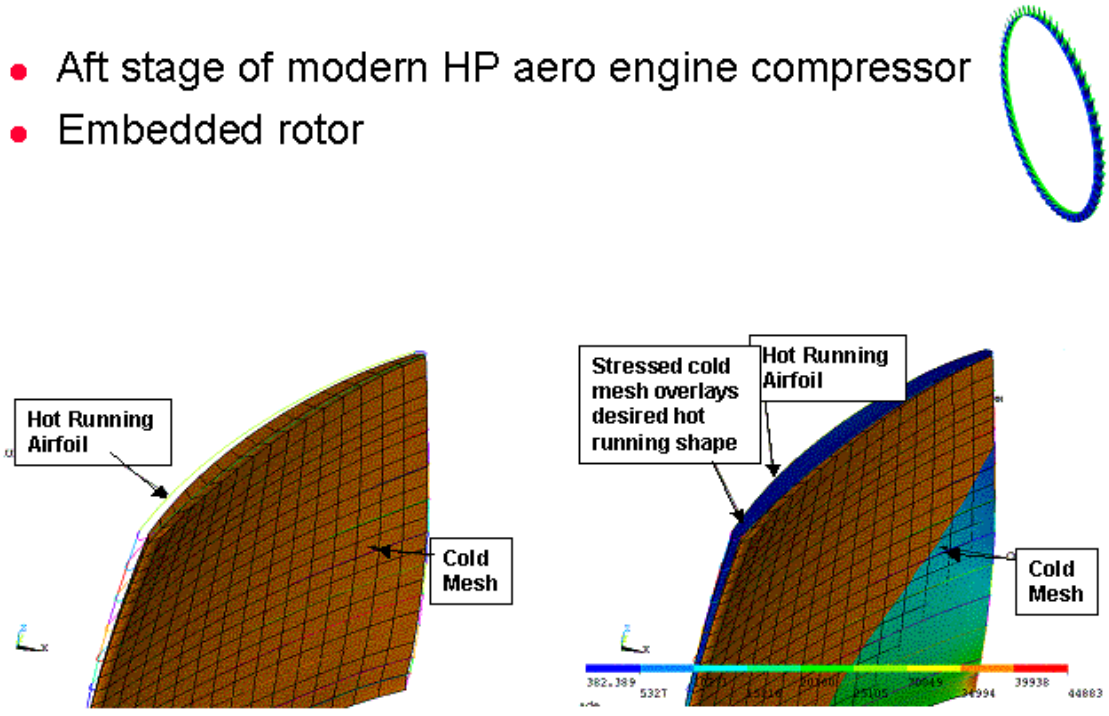

Figure 3.15: Illustration of delection analysis for LCC Rotor.

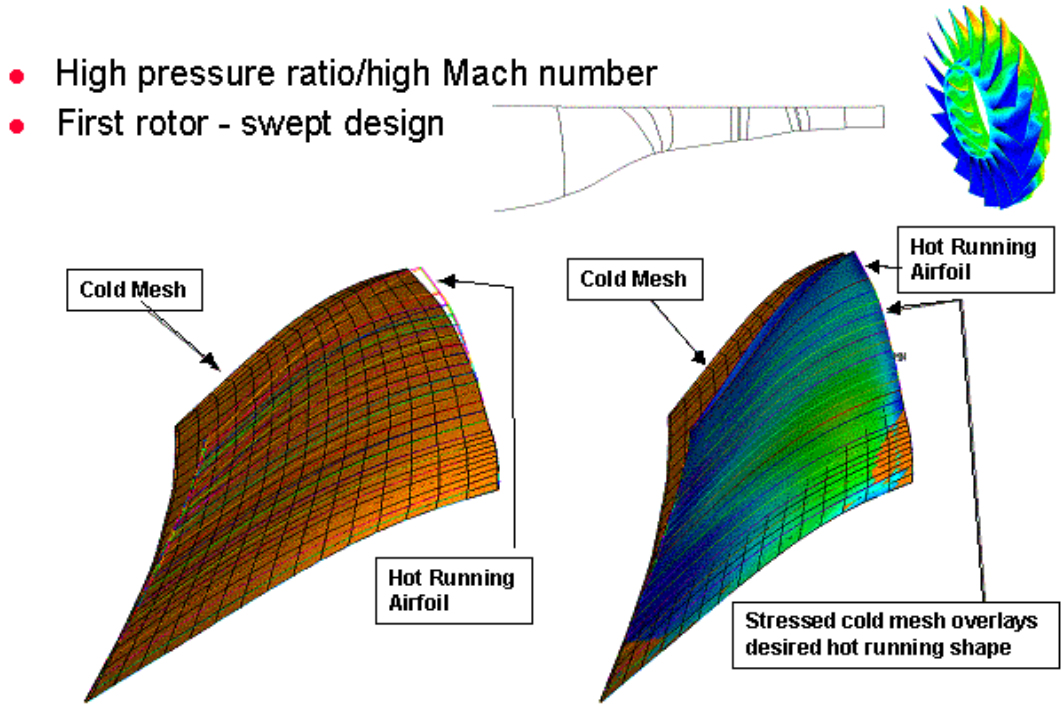

Figure 3.16: Illustration of deflection analysis for the ASTC Rotor 1.

compared to the previous two airfoils examined. Centrifugal loading on the airfoil causes the airfoil to "stand up" resulting in an unusual untwist behavior. Accurate prediction of this behavior is critical to maintain clearance control and obviously, balde shape. Obviously, very high levels of aerodynamic loading also contribute to this behavior, and as airfoil loading increases, accurate prediction of the combined aerodynamic and structural deflection effects will be crucial.

3.7 Evaluation of Reduction in Design Time

An obvious benefit of the combined multidisciplinary analysis is that the aerodynamic and structural engineering analysis tasks can be completed in an automated, simultaneous fashion. This feature results in a significant reduction in time between design iterations. In order to quantify this benefit, a comparison was made between desgn/analysis procedures

in use in the gas turbine industry with the equivalent capability provided by the multidisciplinary analysis. Specifically, the comparison was developed under a conceptual problem involving the design of a modern fan blade for a regional aircraft turbofan engine.

1997 level technology fan design was based on separate discipline analyses for stress, life, aerodynamics, aeromechanics, and noise. Aerodynamics and stress were the predominant analyses, as the remaining disciplines were generally not analyzed in detail until a satisfactory baseline design was achieved. Iterative design procedures in use in 1997 at the Rolls-Royce Indianapolis Engineering facility were based on earlier versions of the ADPAC and ANSYS (as well as other) analysis codes. Engineering key systems were developed up to the point where communication between discipline-specific analyses was based on separate files in proprietary formats, often requiring filters when passing from one analysis type to another. In addition, a significant knowledge and experience base was required to properly perform the discipline specific analyses. Thus, data passing from one discipline to another also passed from one user to another. This feature, in itself, can cause significant time delays as schedules are often not aligned between discipline teams.

The multidisciplinary system described in this report provides savings in design time for two main reasons. First, a single user can develop both the aerodynamic and stress/deflection analysis simultaneously as part of the iterative design process. Second, essential engineering analysis tasks such as the hot-to-cold and cold-to-warm deflection analyses are automated, enabling a significant reduction in user task load and analysis time. Finally, the simple fact that engineering computing hardware has essentially tripled in capability since 1997 represents yet another improvement in reducing design/analysis time.

Based on the assertions listed above, the following conclusions were drawn regarding the benefit of the NPSS multidisciplinary analysis relative to a 1997 capability baseline:

- **Iterative design process can be reduced from two weeks to four days.** This estimate was based on know iterative cycles for the prescribed design task. Rolls-Royce has had onging fan design projects over the past 10 years on a nearly continual basis. Experience suggests that in the absence of any unusual design requirement, a satisfactory baseline design could be achieved in approximately 2 weeks using the "pass by hand" data transfer procedure. The multidisciplinary system enables the evaluation of several design iterations per day, such that the iterative process could be completed in 4 days.

- **Deflection analysis time reduced from 2 days to 1/2 day** In general, 1997 technology did not employ high fidelity data in determining airfoil deflection analysis. Aerodynamic pressure forces were based on simplified schemes based on simple design estimates (if they were represented at all). The deflection task implied in this element is the ability to convert from hot-to-cold coordinates, and then subsequently convert from cold-to-warm (off design) operating points. The automated procedures developed under this task provide complete control of this process and enable rapid turnaround of this task.

Properly coupled with a suitable design system, the multidisciplinary analysis enables a new realm of design studies in rapid fashion. Exploratory designs outside the typical design space that were too risky based on an assumed high level cost can now be evaluated inexpensively by a single engineer. Future advances in coding and computing technology will contribute to further improvements of this nature.

Chapter 4

PROBABILISTIC ANALYSIS

4.1 Introduction

The objective of this portion of the NPSS Multidisciplinary Integration and Analysis task was to adopt a probabilistic solution approach using elements of the multidisciplinary analysis system. Specifically, the probabilistic analysis was invoked in an aerodynamic simulation environment, representing a relatively new application area for the science of probabilistics. To supplement this effort, NASA provided a probabilistic analysis code referred to as NESSUS for this exercise. NESSUS is a modular computer program employing state-of-the-art methods for performing probabilistic analysis of structural/mechanical systems. NESSUS can be used to assess component/system reliability, to identify important random variables, to provide information for risk-based decision analysis and reliability-based optimization, and develop designs that are more cost-effective and reliable than those developed using traditional deterministic design methods.

The traditional application of NESSUS is for structural analysis and reliability predictions. The present study was dedicated to employing the NESSUS probabilistic analysis technique in an aerodynamic prediction framework. The initial approach used was to couple ADPAC to the NESSUS program using the standard ASCII format input file required by NESSUS. The longer term plan was to perform a more integral coupling between ADPAC and NESSUS. Unfortunately this approach was not possible due to the lack of availability of a UNIX version of the NESSUS code. (The supplied version of NESSUS was compiled for use on a Windows NT operating system only.)

Figure 4.1 provides an ideal schematic illustration of the probabilistic analysis concept in the multidisciplinary analysis framework. The desired coupled result would include both aerodynamic factors as well as structural stress assessments. For the purposes of this demonstration, however, only the aerodynamic assessment was performed.

To demonstrate the coupling between ADPAC and NESSUS, a study was performed to predict the effect of uncertainty in the tip gap dimension on the predicted adiabatic

efficiency of the AE3007 Type IV fan blade. The AE3007 turbofan engine is a high bypass (5:1), two spool, axial-flow engine, flat rated at 7,580 pounds of thrust. The design features include: wide-chord single-stage direct-drive fan; 14-stage axial-flow compressor with inlet guide vanes (IGVs) and five variable-geometry-stator stages; full annular type combustion liner with 16 fuel nozzles and two high tension igniters; two-stage HP turbine driving the HP compressor; three-stage LP turbine to drive the fan; dual aircraft mounted, fully-redundant FADECs; accessory-drive gearbox for engine-driven accessories and air system for aircraft pressurisation, engine starting, and cross bleed starts. The AE3007 powers the Cessna Citation X business jet, for which the engine was certificated in early 1996; and the 50-seat Embraer RJ 145 and 37-seat ERJ 135 twin-turbofan regional transports, which are now in service. A growth version, the AE3007H developed for the Northrop Grumman Ryan Aeronautical Centre Global Hawk unmanned surveillance aircraft, is in flight test with the US Air Force.

As part of the development of the AE3007 engine, several fan designs were evaluated for a number of important design criteria including aerodynamic and aeromechanical performance, structural integrity, and resistance to damage due to bird strike. The Type IV fan is the design employed for the EMB-145 regional aircraft application.

4.2 Tip Gap Study

The affect of uncertainty in the tip gap dimension on the predicted adiabatic efficiency was investigated computationally using both ADPAC and NESSUS analysis codes. The tip gap was initially assumed to be a random variable having a normal probability distribution with a mean value of 0.035 inches (35 mils) and a standard deviation of 0.0033 inches (3.3 mils). The tip gap was assumed to be constant from blade to blade, and hence the assumption of spatial periodicity is valid. The analysis roughly approximates variability resulting from the tip dimension grinding operation in an environment where the rotor is assumed to be perfectly centered, and the tip dimension varies only as a constant based on the final tip grinding operation.

The ADPAC CFD code was used to generate constant speed operating curves for both pressure ratio and adiabatic efficiency for tip gap values ranging from 0.025 inches (25 mils) to 0.080 inches (80 mils). The predicted 100% speed total pressure ratio versus corrected mass flow rate operating curves for each tip gap are illustrated in Figure 4.2. The curves were generated by imposing constant, radially varying inflow total pressure, total temperature, swirl and radial flow angle profiles, and then running the code to convergence for a range of fixed hub exit static pressures. The solutions were deemed valid as long as the integrated mass flow, total pressure, and total temperature converged to uniform values over a large number of iterations near the end of the run (a typical run required 2000 iterations). In some cases, for very high pressure ratios, the integrated exit mass flow showed a slow, or occasionally sudden decrease over the tail end of the solution iterative process. This behavior was interpreted as a numerical indicator of stall and these solutions were deemed invalid. Figure 4.3 illustrates the predicted adiabatic efficiency versus corrected mass flow for the range of tip clearance values. The results clearly

illustrate the expected trend that in general mass flow decreases as tip gap increases, and peak efficiency decreases as tip gap increases.

In order to compare results from each curve for the probabilistic analysis, the predicted performance data was interpolated to intersect a separate curve representing a line of constant pressure ratio over corrected mass flow. The design intent value of pressure ratio over corrected mass flow was chosen as the appropriate value and the intersections with the predicted performance data were determined. Once computed, these intersection data formed the basis of the deterministic effect of the tip gap dimension on the predicted adiabatic efficiency. A graphical representation of the intersections are provided in Figure 4.4.

In order to further assess the behavior of this data over a wider range and distribution of tip gap dimensions, a response surface was created from the baseline intersection data. The values for tip gap and efficiency were input into NESSUS and used to define a quadratic response function. The resulting response function is compared to the baseline ADPAC predictions in Figure 4.5.

The response function was used by NESSUS to predict the behavior of adiabatic efficiency using two different probability prediction methods. The first method is referred to as the First Order Reliability Method (FORM), and the second was a standard Monte Carlo procedure. Cumulative distribution functions for the adiabatic efficiency obtained from both methods were compared and showed little difference.

A comparison of several predicted cumulative distribution function curves based on varying tip gap distributions is provided in Figure 4.6. This plot illustrates the change in variability of the system as a function of different levels of variability in the tip gap dimension. Separate curves are provided for the following distributions:

- 35 mils +/- 3.33 mils
- 45 mils +/- 3.33 mils
- 45 mils +/- 6.67 mils
- 55 mils +/- 3.33 mils
- 65 mils +/- 3.33 mils

Several interesting features are evident from these results. First, it is obvious that the results display a wider range of performance variability as the tip clearance increases. This is due to the change in the slope of the predicted response surface for larger tip gap dimensions (see Figure 4.5). The results for the mean value of 35 mils result in a nearly vertical curve, while the results for the 65 mils mean value result in a distribution that is less steep. A second major observation is that the standard deviation can have a significant impact on the shape of the curve. This is illustrated in the comparison of the two curves for the mean value of 45 mils. Obviously, a larger standard deviation implies greater variability, and results in a significant spreading of the cumulative distribution

function for adiabatic efficiency. The results for the 35 mils clearance suggest that over the 1% to 99% range of probability distribution, a total variability of 0.1% in adiabatic efficiency can be expected. For the 45 mils case with a standard deviation of 6.67 mils, the expected variability grows to roughly 0.4% variability in adibatic efficiency.

From these curves, a number of interesting scenarios can be explored. If the design requirement is that a minimum adiabatic efficiency must be maintained, the results suggest that it may be more economical to run a larger clearance with a smaller standard deviation, as opposed to a smaller clearance, with a larger standard deviation. This point is illustrated by comparing the results for 45 mils +/- 6.67 mils standard deviation with the results for 55 mils +/- 3.33 mils standard deviation. Near the lower end of the cumulative probability distribution curves, there is a small region where the curves cross over, indicating a greater likelihood of failing to maintain a minimum performance level for the case with the smaller clearance.

Obviously, these results represent only a small demonstration of the capability provided through the combination of high fidelity aerodynamic analysis and probabilistic analysis. Although this is a relatively simplistic example, it does represent one of the first applications of probabilistics to three-dimensional CFD analysis. This represents a technology area that is ripe for future exploration.

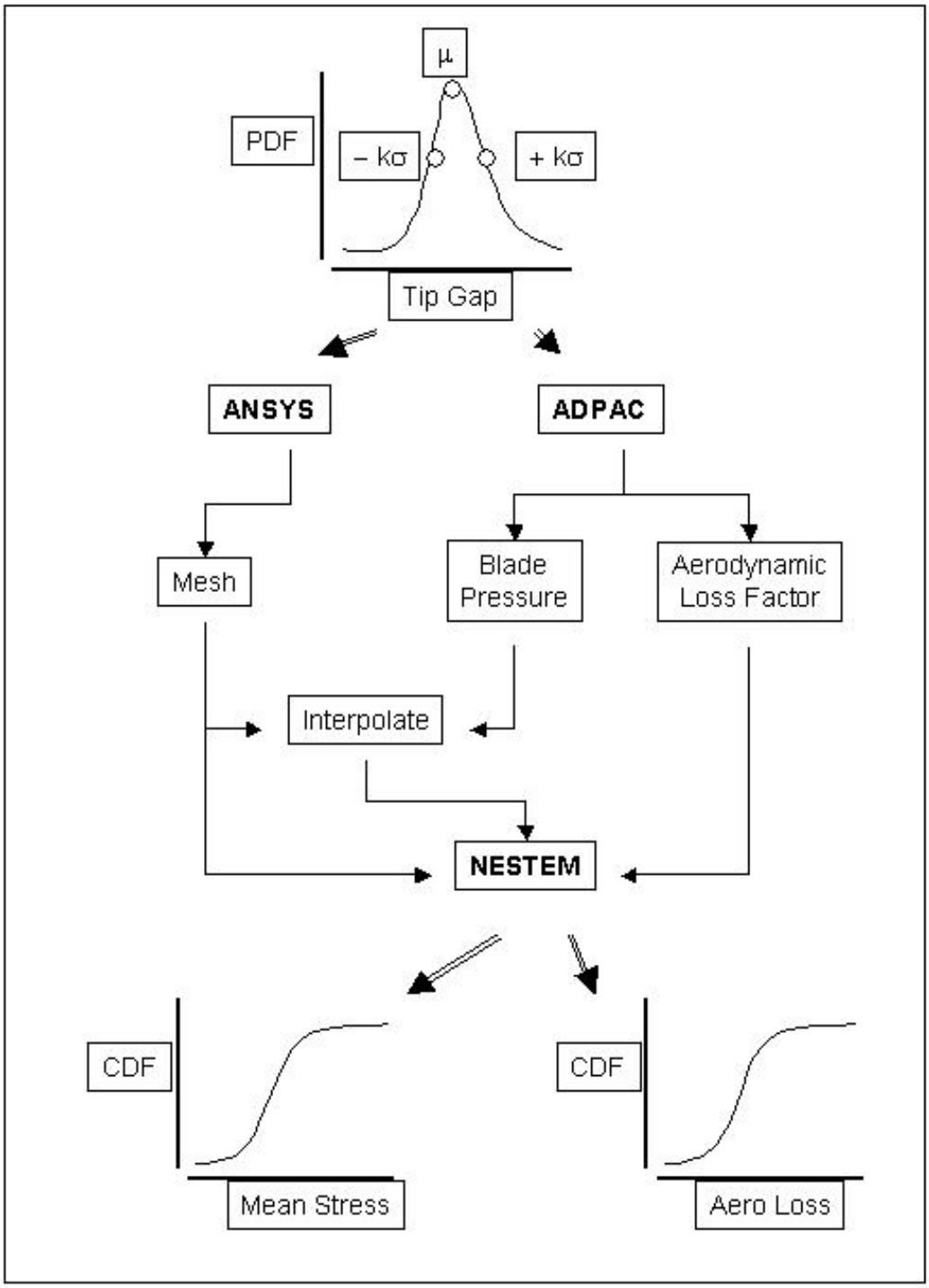

Figure 4.1: Schematic outline of tip clearance probabilistic analysis.

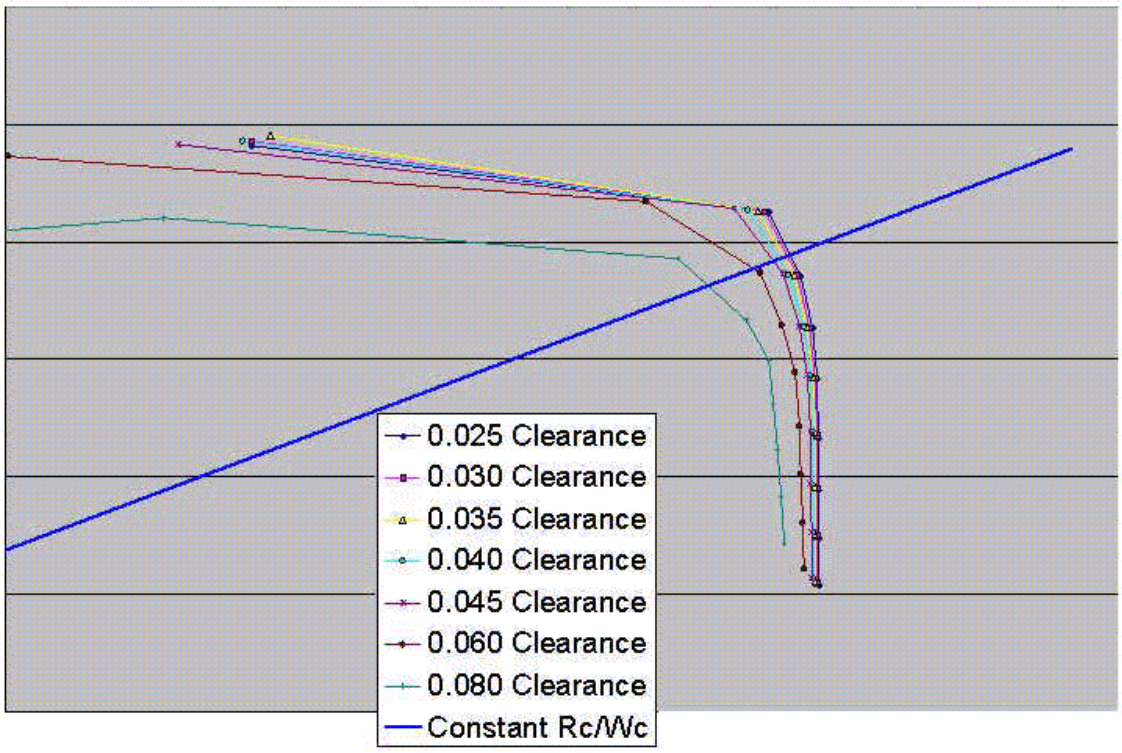

Figure 4.2: AE3007 Type III fan blade predicted 100% corrected speed pressure ratio versus corrected mass flow operating performance curves based on varying tip gap.

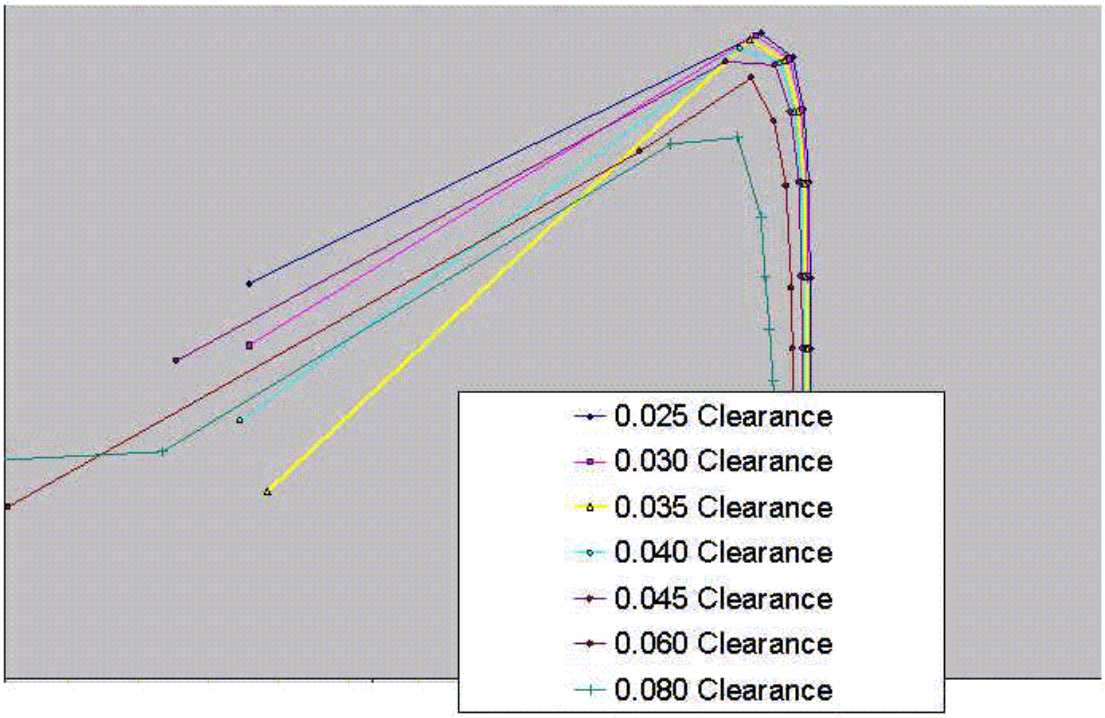

Figure 4.3: AE3007 Type III fan blade predicted 100% corrected speed adiabatic efficiency versus corrected mass flow operatig performance curves based on varying tip gap.

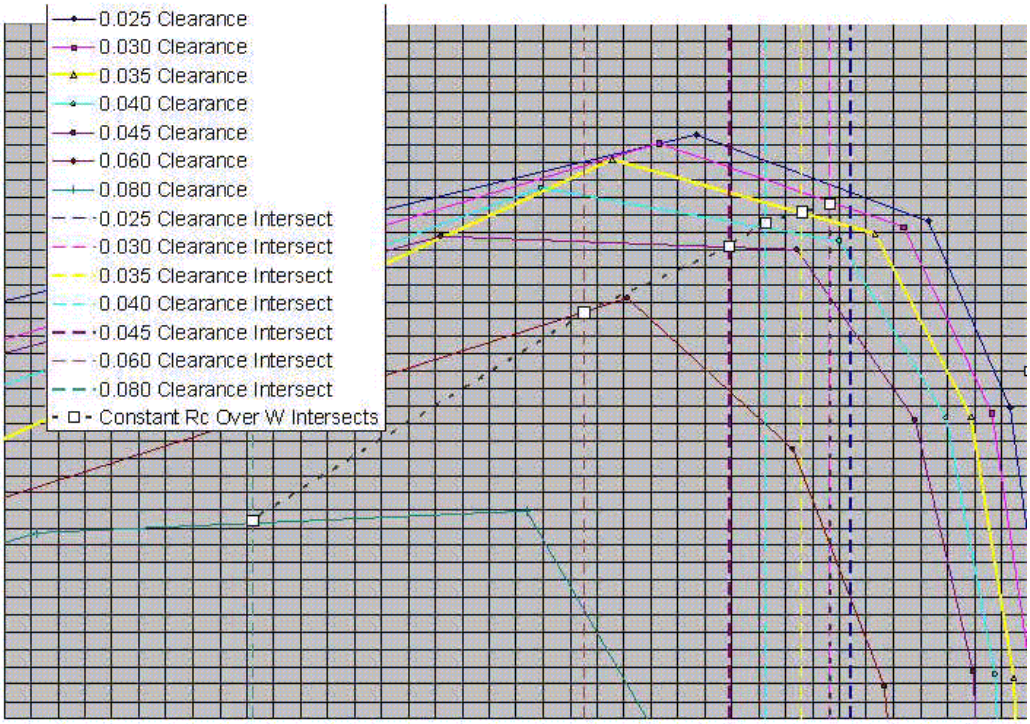

Figure 4.4: AE3007 Type III fan blade predicted 100% corrected speed adiabatic efficiency versus corrected mass flow operating performance curves based on varying tip gap and intersections with constant pressure ratio over corrected flow line.

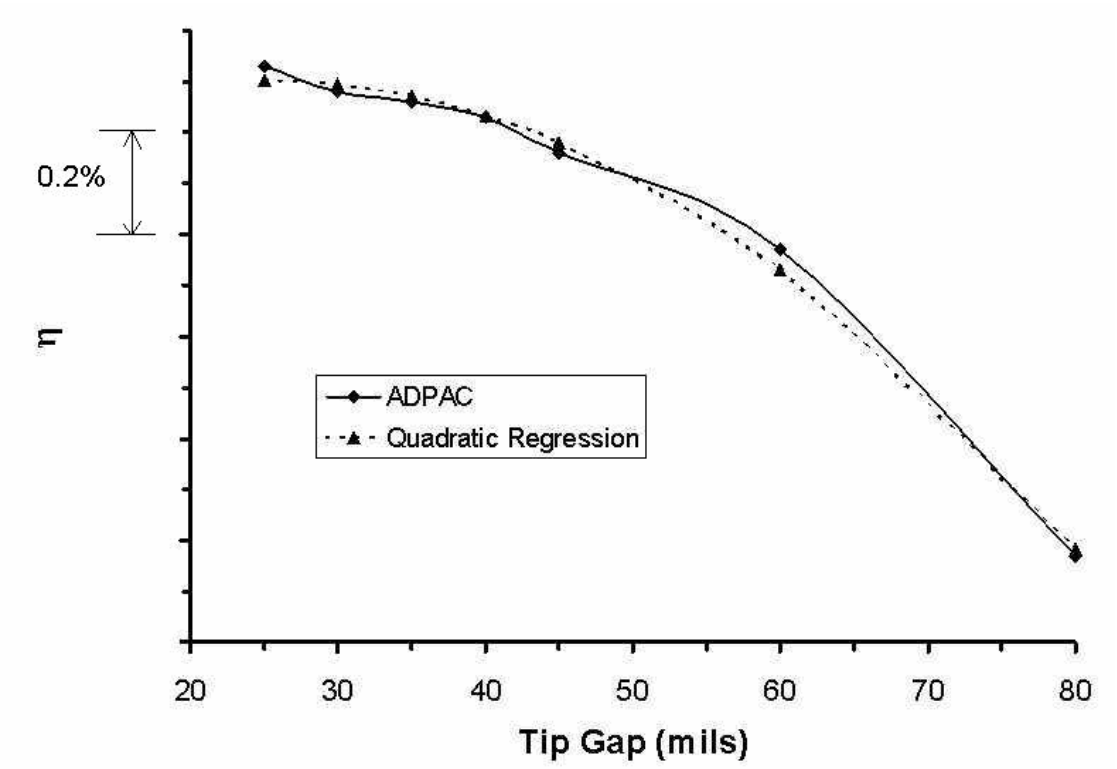

Figure 4.5: Quadratic regression obtained from computation of effect of tip gap on predicted adiabatic efficiency.

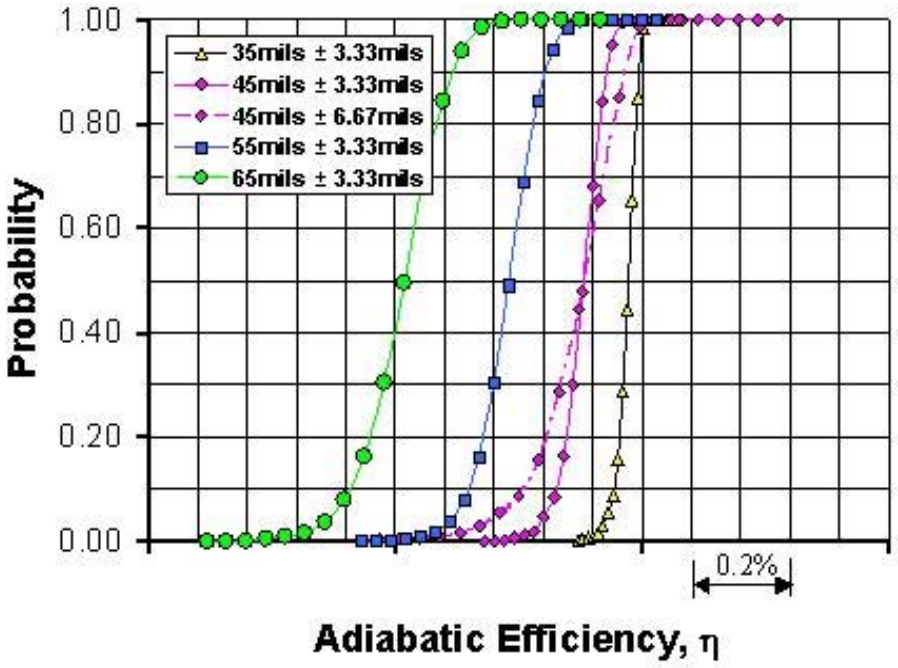

Figure 4.6: Predicted probability density function for fan rotor adiabatic efficiency as a function of varying distribution scenarios for rotor tip gap.

Chapter 5

CAPRI Implementation

5.1 Introduction to CAPRI

CAPRI [8] is a CAD-vendor neutral application programming interface designed for the construction of analysis suites and design systems. By allowing access to the geometry from within all modules (grid generators, solvers and post-processors) such tasks as meshing on the actual surfaces, node enrichment by solvers and defining which mesh faces are boundaries (for the solver and visualization system) become simpler. The overall reliance on file 'standards' is minimized.

This 'Geometry Centric' approach makes multi-physics (multi-disciplinary) analysis codes much easier to build. By using the shared (coupled) surface as the foundation, CAPRI provides a single call to interpolate grid-node based data from the surface discretization in one volume to another. Finally, design systems are possible where the results can be brought back into the CAD system (and therefore manufactured) because all geometry construction and modification are performed using the CAD system's geometry kernel.

The computational steps traditionally taken for Computational Fluid Dynamics (CFD), Structural Analysis, and other simulation disciplines (or when these are used in design) are:

- **Surface Generation** The surfaces of the object(s) are generated usually from a CAD system. This creates the starting point for the analysis and is what is used for manufacturing.
- **Grid Generation** These surfaces are used (with possibly a bounded outer domain) to create the volume of interest. Usually for the analysis of external aerodynamics, the aircraft is surrounded by a domain that extends many body lengths away from the surfaces. This enclosed volume is then discretized (subdivided) in one of many different ways. Unstructured meshes are built by having the subdivisions usually comprised of tetrahedral elements. Another technique breaks up the domain into

sub-domains that are hexahedral. These sub-domains are further divided in a regular manner so that individual cells in the block can be indexed via an integer triad.

- **Flow Solver or Simulation** The solver takes as input the grid generated by the second step (and information about how to apply conditions at the bounds of the domain). Because of the different styles of gridding, the solver is usually written with ability to use only one of the volume discretization methods. In fact there are no standard file types, so most solvers are written in close cooperation with the grid generator. For fluids, the solver usually simulates either the Euler or Navier-Stokes equations in an iterative manner, storing the results either at the nodes in the mesh or in the element centers. The output of the solver is a file that contains the solution.
- **Post-processing Visualization** After the solution procedure has successfully completed, the output from the grid generator and the simulation are displayed and examined in a graphical manner by the fourth step in this process. Usually a workstation with a 3D graphics adapter is used to quickly render the output from data extraction techniques. The tools (such as iso-surfacing, geometric cuts and streamlines) allow the examination of the volumetric data produced by the solver. Even for steady-state solutions, much time is usually required to scan, poke and probe the data in order to understand the physics in the flow field.

These steps have worked well in the past for simple steady-state simulations at the expense of much user interaction. The data was typically transmitted between phases via files. In most cases, the output from a CAD system could go to IGES files. The output from Grid Generators and solvers do not have agreed standards, although the CFD General Notation SYstem (CGNS) described in the next chapter is a proposed standard. PLOT3D data formats served as a long-standing informal standard for a subset of the problem space. The user would have to patch up the data or translate from one format to another to move to the next step. Sometimes this could take days. Specifically, the problems with this procedure are:

- **File based** Information flows from one step to the next via data files with formats specified for that procedure. Historically, this allows individuals or groups to work in isolation on the construction of one of these components; unfortunately the user (or team) suffers greatly because of the lack of integration. In many cases the files that get used do not contain all the information required to couple all components so that the user can be removed from the mechanics of running the suite.
- **'Good' Geometry** A bottleneck in getting results from a solver is the construction of proper geometry to be fed to the grid generator. With 'good' geometry a grid can be constructed in tens of minutes (even with a complex configuration) using unstructured techniques. Adroit multi-block methods are not far behind. This means that a million node CFD steady-state solution can be computed on the order of hours (using current high performance computers) starting from this 'good' geometry. Unfortunately, geometry from CAD systems (especially when transmitted via IGES files) is not 'good' in the grid generator sense. The data is usually defined

as disjoint and unconnected surfaces and curves (as well as masses of other information of no interest for the mesh). The grid generator needs smooth closed solid geometry. It can take a week (or more) of interaction with the CAD output (sometimes by hand) before the process can begin. This is particularly onerous if the CAD system is based on solid modeling. The part was a proper solid with topology and in the translation to IGES has lost these important characteristics.

- **One-Way Comminucation** All information travels on from one phase to the next. This makes procedures like node adaptation difficult when attempting to add or move nodes that sit on bounding surfaces (when the actual surface data has been lost after the grid generation phase). In fact, the information passed from phase to phase is not enriched but is filtered.

Until this process can be automated, more complex problems such as Multi-disciplinary analysis or using the above procedure for design becomes prohibitive. There is also no way to easily deal with this system in a modular manner. One can only replace the grid generator, for example, if the software reads and writes the same files. Procedures like zooming, defined within the Numerical Propulsion System Simulation (NPSS), are difficult to acheive when the surface definition for the coupling between the simulations is lost.

Instead of the serial approach to analysis as described above, CAPRI uses a geometry centric approach. This makes the actual geometry (not a discretized version) accessible to all phases of the analysis. The connection to the geometry is made through an Application Programming Interface (API) and NOT a file system. This API isolates the top level applications (grid generators, solvers and visualization components) from the geometry engine. Also this allows the replacement of one geometry kernel with another, without effecting the top level applications. For example, if UniGraphics is used as the CAD package then Parasolid (UG's geometry engine) can be used for all geometric queries so that no solid geometry information is lost in a translation. If Pro/E is used then Pro/Toolkit is accessed when geometric information is required.

5.2 CAPRI Implementation

The purpose of the CAPRI sub-task in the multidisciplinary development was to establish CAPRI in a commercial grid generation package, examine the functionality, and provide feedback for the CAPRI developers to enable improvements in the library. The ultimate goal is obviously to enable CPARI as the programming interface for all geometry-related data access within the multidisciplinary analysis, and ultimately the NPSS system.

These objectives were satisfied through a subcontracted effort to ICEM CFD, Inc. ICEM CFD provides a range of CFD-related products featuring inherent ties to CAD geometry database access. A direct demonstration of the CAPRI interface capability was provided through the implementation of CAPRI to enable access to a new CAD geometry

database. Since there was no direct interface to ComputerVision geometry in the original ICEM CFD tool set, the CAPRI library was employed to provide this functionality.

The CAD integration effort with Computer Vision has focused on three areas:

- **A "traditional" ICEM CFD direct interface to CV.** This is a non-interactive application that currently provides read-only access to CV geometry through the CV-DORS package. It writes an ICEM CFD native file, called a TETIN file, that is used to transfer geometry to the ICEM CFD meshing utilities. Exact geometry is transferred. Invariance of the mesh definition under model updates is accomplished using the native grouping capabilities of CV. The CV interface is currently ported to SGI and Solaris 6 for CV11 and Solaris 2.51 for CV8.The interface will be available with ICEM CFD 4.2.

- **A binding of CAPRI to CV.** CAPRI is a CAD-vendor neutral application programming interface designed for the construction of analysis suites and design systems. Currently the CAPRI API has been implemented over SDRC, Pro-Engineer, ACIS, Parasolid, Catia, and now Computer Vision. ICEM CFD has written a simple CAPRI based application to access a CAD model and import to ICEM CFD using a facetted representation. This application has been written for the CV, SDRC, and Pro Engineer CAPRI bindings and will be included with the 4.2 version of ICEM CFD. It is expected to be particularly useful for large models that are intractable when imported using an exact b-spline representation. Future plans with CAPRI Master Modeling CAPRI's creator, Bob Haimes, are in the process of extending CAPRI's abilities to manipulate the CAD model. The current version of CAPRI permits the interrogation and manipulation of part parameters. The version of CAPRI currently under development will extend this capability to include operations on the CAD model's feature tree. By manipulating the feature tree, an analysis application can defeature a model for more effective analysis. ICEM CFD plans to expose this functionality as a "model manipulation" feature inside its mesh editor application.

- **Integration with CGNS** The CFD General Notation System (CGNS) is a system of conventions and software for storage and retrieval of CFD data. It is designed to facilitate the exchange of aerodynamic data between sites and applications. The functionality of CGNS is highly complementary to the functionality of CAPRI. ICEM CFD is in the early stages of discussions with Bob Haimes to identify a problem or application that would simultaneously exercise the capabilities of both CAPRI and CGNS. It is possible that CAPRI could ultimately form the basis for the geometry storage elements of the CGNS standard.

Chapter 6

CGNS and ADPAC

6.1 Description of CGNS

The CFD General Notation System (CGNS) consists of a collection of conventions, and software implementing those conventions, for the storage and retrieval of CFD (computational fluid dynamics) data. The system consists of two parts: (1) a standard format for recording the data, and (2) software that reads, writes, and modifies data in that format. The format is a conceptual entity established by the documentation; the software is a physical product supplied to enable developers to access and produce data recorded in that format.

The CGNS system is designed to facilitate the exchange of data between sites and applications, and to help stabilize the archiving of aerodynamic data. The data are stored in a compact, binary format and are accessible through a complete and extensible library of functions. The API (Application Program Interface) is platform independent and can be easily implementd in C, C++, Fortran and Fortran90 applications.

6.2 Overview of CGNS for ADPAC

The CFD General Notation System was developed to provide a standard for the communication of information between structured-grid multiblock Navier-Stokes analysis codes from surface definition through grid generation, flow solution and post-processing.

The ADPAC program solves a discretized form of the Navier-Stokes equations based on a flexible multiple-block grid discretization scheme permitting coupled 2D/3D mesh block solutions. Steady flow predictions are accelerated by a multi-grid procedure. The code is capable of executing in either a serial or parallel computing mode from a single source code.

This chapter studies the possibility of using the CFD General Notation System for archival and retrieval of data for use with the ADPAC v1.0 solver.

This report shows that most of the data recorded by ADPAC can be easily stored in the CGNS system. In the next section, the ADPAC data is reviewed, compared with the CGNS standard, and suggestions are made to facilitate the implementation of CGNS in ADPAC. All of the data recorded in the ADPAC multiple input and output files can be recorded in a single CGNS file, allowing to keep all the case data in a unique file. For highly standard data such as grid coordinates, flow solutions, convergence history, etc., existing CGNS data structures can be used directly. For data specific to ADPAC, such as those found in the ADPAC input file, it is recommended to use the general CGNS node Descriptor_t.

Although most of the data recorded and used by the ADPAC solver can be stored using CGNS, three key elements need to be added to CGNS to allow a full support of the ADPAC capabilities. The first major element that was missing from the CGNS standard was the support of time dependant and iterative data, with or without rigid or arbitrary grid motion.

Most of the effort spent over the past year was concentrated on adding new data structures to the CFD General Notation System to support time dependant data, as used by ADPAC and other similar codes. The development of these new data structures, the supporting documentation and complete software capability to perform the input/output of these data, are completed and available for use. Section 6.5 of this report describes in detail the addition of time dependant/iterative data in the CGNS system.

The second crucial missing element is the support of multi-grid data. The CGNS steering committee is currently reviewing a proposal to add support to multi-grid mesh data. This proposal is outlined in section 6.6 of this report.

The last element is the parallelization of CGNS. Currently, the CGNS Committee is discussing options to facilitate parallel implementations of the CGNS system, but no concrete effort has yet been undertaken.

6.3 ADPAC Data in a CGNS File

The main advantage of using CGNS for ADPAC is that an entire case could be recorded under a single database file, rather than the current collections of 20 or more files for a single case. This section reviews the data contain in each ADPAC file and suggests how this information could be recorded under a single CGNS file.

6.3.1 Portability Across Platforms

The CGNS system may be implemented in any CFD applications through the use of a complete and self-describing set of functions. The API is accessible through C or Fortran77 function calls allowing implementation in C, C++, Fortran77 and Fortran90 applications. The databases themselves are stored in compact C binary files. They are made machine independent through internal byte ordering translations, performed as

needed and invisible to the user or application. The API performs extensive error checking on the database and informs the user of any irregularities via precise error diagnosis messages.

Currently the CGNS system is available on most architectures commonly used for CFD analysis: Cray/Unicos, SUN/Solaris, SGI/IRIX, IBM/AIX, HP/UX, DEC-Alpha/OSF, Windows NT, Linux, as well as the NEC.

6.3.2 ADPAC Maximum Array Dimension

Before compiling and running the ADPAC program, the array size for the particular case in study must be defined. This information is currently kept in the file "parameter.inc" which is then included in the FORTRAN subroutines for compilation.

This information could be recorded and saved in the CGNS file, directly under the CGNSBase_t node, for future reference. A Descriptor_t node named "parameter.inc" could be created under the CGNSBase_t node to include the list of all parameters name and size.

The computational tool ADSTAT could create a new CGNS file and store parameter size automatically in the Descriptor_t parameter.inc. At time of compilation, this information could be read by a small utility that would create a temporary parameter.inc file for inclusion in the FORTRAN subroutines.

This would have the advantage to keep the array dimensions setting together in a same file with all the other data relevant to a particular case.

6.3.3 ADPAC Standard Input File, *case*.input

The user-specified parameters controlling the operation of the ADPAC program are kept in the case.input file. These are represented by a series of keywords and associated values. Version 1.0 of ADPAC lists 85 different keywords that can be specified in the input file. Each keyword represents an input variable for which a default value is defined. Therefore only the variables requiring a different value than the default need to be set in the input file.

It is suggested to replace the input file by a Descriptor_t node recorded directly under the CGNSBase_t node in the CGNS case file and to name this Descriptor_t node "input". There is no limit in the length of the descriptor recorded in the Descriptor_t node, and it may also include carriage returns or special characters. Therefore, the same format currently use for the input file could be also used for the descriptor. Each line would include a keyword followed by an equal sign and the value of the input variable. Each line that does not contain a recognizable keyword would be treated as a comment line, allowing users to place any number of comments in the descriptor node. At execution, the ADPAC program would read the input variables directly from the CGNS case file.

6.3.4 ADPAC Boundary Data File, *case*.boundata

The ADPAC boundary data file describes both the interblock connectivity for patch located at interface between blocks and the patch boundary condition data. In the CGNS file, the same information is divided under two different sub-structures, one for the interblock mesh connectivity and one for the boundary conditions. These sub-structures are named the ZoneGridConnectivity_t and the ZoneBC_t respectively.

ZoneGridConnectivity_t

The zone connectivity information may be recorded for each zone under a general data structure called ZoneGridConnectivity_t. This data structure holds three sub-structures responsible for the grid connectivity data, GridConnectivity_t, GridConnectivity1to1_t, and OversetHoles_t. It may also include some documentation recorded in the sub-structure Descriptor_t.

All three types of multi-block connectivity may be defined using the general connectivity sub-structure called GridConnectivity_t. It contains the list or range of indices defining the interface in the current zone (receiver), the name of the adjacent zone (donor), the list of points on the donor side and a parameter specifying the type of connectivity.

If the interface is constituted of points having consecutive indices, the patch may be defined by simply referencing the first and last indices of the range. This patch definition method is called PointRange. When the points do not have consecutive indices numbering, they are all recorded in a structure called PointList. For example, an abutting one-to-one interface between two structured blocks represents a rectangular sub-range in the computational domain. It can be simply defined using a PointRange.

The list of points on the donor side, PointListDonor, contains the images of the receiver zone interface points in the donor zone. These are real values identifying the bi- or tri-linear interpolation factors used to define the location of each receiver point in the donor zone grid. For the particular case where the points on the donor side coincide with those on the receiver side, the donor points correspond to the indices of the nodes on the donor side. The types of connectivity are identified using the standardized names Overset, Abutting and Abutting1to1.

A special sub-structure is defined in the SIDS for the recording of an abutting interface patch with one-to-one point matching between two adjacent structured zones. In this particular case the connectivity may be entirely defined by simply identifying the range of points delimiting the interface in both adjacent zones, and a transformation matrix describing the relative indices orientation of the zones. This information is recorded in GridConnectivity1to1_t, a special data structure created for this particular case.

The ADPAC boundary condition type PATCH could be directly translated into the GridConnectivity1to1_t data structure. The mismatched abutting interblock connectivity, as defined by the boundary type BCINT1, BCINTM, BCPRM, BCPRR and PINT would

require the more general data structure GridConnectivity_t with the grid connectivity type set to "Abutting".

ZoneBC_t

In CGNS, the boundary conditions can be defined either on mesh patches or on geometrical entities. Associating boundary conditions to mesh patches permits specification of local data sets at the vertices or face centers of the boundary condition patch. The zonal boundary condition structure, ZoneBC_t, also provides a means to define boundary conditions without requiring geometric data in the file.

Within the hierarchy, the ZoneBC_t structure is located directly under each zone and contains the list of boundary condition structures (BC_t) pertaining to the zone. Each BC_t sub-structure contains the boundary condition information for a single patch of the zone. It provides for the recording of a boundary condition type (BCType_t) as well as one or more sets of boundary condition data (BCDataSet_t). The BC_t data structure also contains information describing the patch itself, such as its location (PointRange or PointList) and its normal vector definition

The first item identifying the boundary condition equations to be enforced at a given boundary location is the boundary condition type, BCType_t. The boundary condition types are identified using standardized names such as BCDirichlet, BCWallViscous and BCInflowSubsonic. The second item used in the definition of a boundary condition is the boundary condition data set, BCDataSet_t. It holds a list of variables defining the boundary condition. Each variable may be given as global data or local data defined at each grid point of the boundary condition patch.

Some of the boundary condition types defined in the CGNS standard have a direct equivalent in ADPAC. However, many ADPAC boundary condition types are specific to ADPAC and have no equivalence in the CGNS standard. Therefore, for a first implementation of ADPAC, it is suggested to keep the current ADPAC boundary type names and record them as UsedDefined boundary condition patches. The specific information required by ADPAC for a particular ADPAC boundary condition type could be recorded in the Descriptor_t node directly under the BC_t node of the patch.

For example, a boundary condition type INLETA could be defined as follow:

```
BC\_t  {
   Descriptor\_t  INLETA {PTOT=1.0 TTOT=1.0 ALPHA=20.0 CHI/AKIN=1.00 ARIN=0.0001}
   BCType\_t  UserDefined
   GridLocation\_t Vertex
   IndexRange\_t PointRange
   int[IndexDimension]  InwardNormalIndex
 } ;
```

The CGNS PointRange is equivalent to the ADPAC definition using LFACE1, L1LIM, M1LIM1, M1LIM2, N1LIM1 and N1LIM2. The InwardNormalIndex has the same function as the ADPAC variable LDIR1.

6.3.5 ADPAC Mesh File, *case*.mesh

The physical coordinates of the computational grid are defined by the grid coordinates data structure. This structure contains a list of data arrays representing the individual components of the position vector. If necessary, the nondimensionalization information and dimensional unit sub-structures may also be defined.

The SIDS support coordinate definition in Cartesian, cylindrical and spherical coordinate systems. A series of standardized names supplied by the SIDS unambiguously identifies the content of each coordinate data array. These data-name identifiers are self-describing, for example CoordinateX and CoordinatePhi.

ADPAC mesh data would be recorded for each block in the GridCoordinates_t node named GridCoordinates under each Zone_t node. The Cartesian data arrays would be identified CoordinateX, CoordinateY and CoordinateZ. The dimensional scaling factor DIAM can also be recorded for each data array in the DataConversion_t data structure.

6.3.6 Body Force File, *case*.bf.#

The flow solution data structure, FlowSolution_t, is used to record one solution data set under a Zone_t node. There is no limit on the number of solutions sets contained in a zone data structure. Each solution set, in term, may include one to several solution vectors. A flow solution data structure could be used, for example, to hold the initial or restart solution, while a second one could serve to record the computed solution after some number of iterations.

The flow solution data structure contains all the sub-structures found in the grid coordinate structure (data arrays, dimensional information, rind-data and documentation), with additionally the grid location parameter. Unlike the grid coordinates, which always coincide with the vertices, the flow solutions may be defined at the vertices, or at cell, face or edge centers. This extra feature necessitates the creation of two different data structures to hold grid coordinates and flow solutions. Once again, the SIDS specification regulates the variable names in order to facilitate a standardized data exchange. A list of data-name identifiers for typical Navier-Stokes solution variables was established, and can be easily extended if needed. It contains identifiers such as Pressure, VelocityX, SkinFrictionY, MassFlow, etc. It is also possible to record any type of site specific variables, even if they are not included in the SIDS nomenclature.

The ADPAC body force variables BFR, BFRU, BFRV, BFRW, BFRE and BL may therefore be used directly in the CGNS file for identifying the different solution arrays. The SIDS data name identifiers includes the names "Density", "MomentumX",

"MomentumY", "MomentumZ", and "EnergyInternal" for flow solution quantities. However, it does not have specific names for axial, radial and circumferential momentum, as well as for the blockage term (BL). These variables could eventually be added to the list of data name identifiers supported by the SIDS, if it is shown that they are widely used.

For a first implementation, the body force for each block could be recorded as follow:

```
Zone\_t block N
  FlowSolution\_t body\_force
    DataArray\_t BFR
    DataArray\_t BFRU
    DataArray\_t BFRV
    DataArray\_t BFRW
    DataArray\_t BFRE
    DataArray\_t BL
```

Dimensional units and data conversion factor could also be recorded for each solution vector.

6.3.7 ADPAC Standard Output File, *case*.output

The convergence history data recorded in the case.output file could be recorded under the ConvergenHistory_t data structure in the CGNS file. Globally relevant convergence history information is contained in "ConvergenceHistory_t GlobalConvergenceHistory", located directly under the base node. This convergence information may include total configuration forces, moments, and global residual and solution-change norms taken over all the zones. Convergence history information applicable to the zone is contained in "ConvergenceHistory_t ZoneConvergenceHistory", located under each zone.

The remaining information contained in the output file could be recorded under a Descriptor_t node named "output", directly under the base node.

6.3.8 ADPAC Plot Files

Flow Data Plot Files, *case*.p3dabs and *case*.p3drel

Each of the 5 flow field variables, R, RU, RV, RW and RE, has a standard name, defined in the SIDS. They are, respectively: Density, MomentumX, MomentumY, MomentumZ, and EnergyStagnationDensity. Two FlowSolution_t nodes can be added under each zone to hold the absolute and relative reference frames flow data values. Since these data are defined at the vertices, it is not necessary to specify the grid location (GridLocation_t), which by default, is defined to be "Vertex".

To output nondimensional data with known normalizations, specify the data class directly under the base node, such as DataClass_t = NormalizedByDimensional. This

indicates that the data has been nondimensionalized by known reference values, which are specified in the CGNS file. Then specify the reference state values using the ReferenceState_t data structured, also directly under the base node. The reference state values should include the Mach and Reynolds numbers, as well as the velocity in x, y, and z, if an angle of attack of the reference state is needed. The SIDS standard names for these variables are respectively Mach, Reynolds, VelocityX, VelocityY, and VelocityZ. The dimensional reference values of Density and VelocitySound should also be included to allow to recover the raw data from the nondimensional data recorded under FlowSolution_t.

Turbulence Parameter Data Plot Files, *case*.p3d1eq or *case*.p3d2eq

In CGNS, the turbulence model data structure, TurbulenceModel_t, describes the equation set used to model the turbulence quantities. It is recorded under the flow equation set data structure, FlowEquationSet_t/TurbulenceModel_t, directly under the base or under each zone. The turbulence model data discretized over the entire mesh should be recorded under a FlowSolution_t node. For example:

```
FlowSolution\_t p321eq {
  DataArray\_t  TurbulentSANuTilde

}
```

As for the flow solution data, the turbulence model variables may be recorded as nondimensional data, with the reference values specified under the ReferenceState_t node, under the base node.

6.3.9 ADPAC Restart Files

All restart files should be recorded under different FlowSolution_t nodes, and could be named "restart.new", "restart.old", "restart_SA.new", "restart_SA.old", "restart_KR.new" and "restart_KR.old". Since these solutions contain cell-centered flow variables, the node GridLocation_t must be defined with location set to "CellCenter".

With the recent addition of time dependant support in CGNS, the restart files could be recorded at each N iterations cycles. Then at restart, the latest flow solution could be automatically used. Section 4 of this report gives more details on the new data structures created for time dependant and iterative data.

6.3.10 ADPAC Convergence Files

In CGNS, flow solver convergence history information is described by the ConvergenceHistory_t structure. This structure contains the number of iterations and a list of data arrays containing convergence information at each iteration.

Measures used to record convergence vary greatly among current flow-solver implementations. Convergence information typically includes global forces, norms of equation residuals, and norms of solution changes. Attempts to systematically define a set of convergence measures within the CGNS project have been futile. For global parameters, such as forces and moments, a set of standardized data-array identifiers is used. For equations residuals and solution changes, no such standard list exists. It is suggested that data-array identifiers for norms of equations residuals begin with RSD, and those for solution changes begin with CHG. For example, RSDMassRMS could be used for the L2-norm (RMS) of mass conservation residuals. It is also strongly recommended that NormDefinitions be utilized to describe the convergence information recorded in the data arrays. The format used to describe the convergence norms in NormDefinitions is currently unregulated.

6.3.11 ADPAC Image Files

CGNS does not provide any means for recording images. However, a list of image file associated with the case in study could be recorded under a Descriptor_t node located directly under the base. Such list could include the file name or the URL of the images as well as a desciption and/or analysis of each image.

6.4 Addition of iterative or time-accurate data, rigid body and arbitrary grid motion

In order to keep a record of time dependent or iterative data, two new data structures called BaseIterativeData_t and ZoneIterativeData_t were added to the CFD General Notation system. The BaseIterativeData_t data structure is recorded directly under the CGNSBase_t node. It contains information about the number of time steps or iterations being recorded, and the time and/or iteration values at each step. In addition, it may include the list of zones and families for each step of the simulation, if these vary throughout the simulation.

The ZoneIterativeData_t data structure is located under the Zone_t node. It allows to record pointers to zone data for each recorded step of the simulation.

6.4.1 The BaseIterativeData_t data structure under the CGNSBase_t data structure:

```
CGNSBase\_t:=
  {
  BaseIterativeData\_t   BaseIterativeData                    (o)
    {
    int NumberOfSteps                                  (r)
    DataArray\_t<real,1,NumberOfSteps> TimeValues ;              (o/r)
```

```
    DataArray\_t<real,1,NumberOfSteps> IterationValues ;          (r/o)

    DataArray\_t<int,1,NumberOfSteps> NumberOfZones ;                 (o)
    DataArray\_t<int,1,NumberOfSteps> NumberOfFamilies ;          (o)
    DataArray\_t<char,3,[32,MaxNumberOfZones,NumberOfSteps]>
ZonePointers ;                                (o)
    DataArray\_t<char,3,[32,MaxNumberOfFamilies,NumberOfSteps]>
FamilyPointers;                               (o)

    List(DataArray\_t<> DataArray1 ... DataArrayN)                (o)
    List( Descriptor\_t Descriptor1 ... DescriptorN ) ;            (o)
    DataClass\_t DataClass ;                                   (o)
    DimensionalUnits\_t DimensionalUnits ;                     (o)
    }
  ...
  }
```

Notes:

1. The NumberOfSteps is a required element of the BaseIterativeData_t data structure. It holds either the number of time steps or the number of iterations.

2. TimeValues or IterationValues must be defined. If both are used, there must be a one-to-one correspondence between them.

3. The fields NumberOfZones and ZonePointers are optional. They are used if different zone data structures apply to different steps of the simulation. Similarly, if the geometry varies with time or iteration, then the fields NumberOfFamilies and FamilyPointers are used to record which Family_t data structure(s) correspond(s) to which step.

4. The DataArray_t for ZonePointers and FamilyPointers are defined as tri-dimensional arrays. For each recorded step, the name of all zones and families being used for this step may be recorded. Note that the names are limited to 32 characters, as usual in the SIDS. The variables MaxNumberOfZones and MaxNumberOfFamilies represent the maximum number of zones and families that apply to one step. So if NumberOfSteps=5 and NumberOfZones=2,2,3,4,3, then MaxNumberOfZones equals 4.

5. When NumberOfZones and NumberOfFamilies vary for different stored time step, the name "Null" is used in ZonePointers and FamilyPointers as appropriate for steps in which the NumberOfZones or NumberOfFamilies is less that the MaxNumerOfZones or MaxNumberOfFamilies.

6. The DataClass_t, DimensionalUnits_t and Descriptor_t nodes may optionally be specified under the ZoneIterativeData_t node.

7. Any numbers of extra DataArray_t nodes are allowed. These should be used to record data not covered by this specification.

The mapping of the BaseIterativeData_t data structure onto the ADF database is illustrated in Figure 6.1.

6.4.2 The ZoneIterativeData_t data structure under the Zone_t data structure:

```
Zone\_t<int CellDimension, int PhysicalDimension > :=
  {
  ZoneIterativeData\_t ZoneIterativeData <NumberOfSteps> :=          (o)
    {
    DataArray\_t<char,2,[32,NumberOfSteps]> RigidGridMotionPointers ;    (o)
    DataArray\_t<char,2,[32,NumberOfSteps]> ArbitraryGridMotionPointers ;(o)
    DataArray\_t<char,2,[32,NumberOfSteps]> GridCoordinatesPointers ;    (o)
    DataArray\_t<char,2,[32,NumberOfSteps]> FlowSolutionsPointers ;  (o)

    List(DataArray\_t<> DataArray1 ... DataArrayN)               (o)
    List( Descriptor\_t Descriptor1 ... DescriptorN ) ;           (o)
    DataClass\_t DataClass ;                                  (o)
    DimensionalUnits\_t DimensionalUnits ;                    (o)
    }
  ...
  }
```

1. The data arrays xxxPointers contain lists of associated data structures for each time step or iteration. They refer by name to data structures within the current zone. The name "Null" is used when a particular time or iteration does not have a corresponding data structure to point to.

2. The DataClass_t, DimensionalUnits_t and Descriptor_t nodes may optionally be specified under the ZoneIterativeData_t node.

3. Any numbers of extra DataArray_t nodes are allowed. These should be used to record data not covered by this specification.

4. The ZoneIterativeData_t data structure may not exist without the BaseIterativeData_t under the CGNSBase_t node. However BaseIterativeData_t may exist without ZoneIterativeData_t.

The mapping of the ZoneIterativeData_t data structure onto the ADF database is illustrated in Figure 6.2.

6.4.3 The RigidGridMotion_t data structure under the Zone_t data structure:

Adding the rigid body motion information to the CGNS file enables an application code to determine the mesh location without the need to alter the original mesh definition recorded under GridCoordinates_t. A new data structure named RigidGridMotion_t is created to record the necessary data defining a rigid translation and/or rotation of the grid coordinates.

The rigid grid motion is recorded independently for each zone of the CGNS base. Therefore the RigidGridMotion_t data structure is added under each the zone data structure (Zone_t). There may be zero to several RigidGridMotion_t nodes under a Zone_t node. The multiple rigid grid motion definitions may be associated to different iteration or time step in the computation. This association is recorded under the ZoneIterativeData_t data structure.

SIDS definition of the RegionGridMotion_t data structure:

The RigidGridMotion_t under the Zone_t data structure:

```
Zone\_t<int CellDimension, int PhysicalDimension > :=
 {
 List(RigidGridMotion\_t RigidGridMotion1...
RigidGridMotionN) ;                (o)
 ...
 }
```

The RigidGridMotion_t data structure:

```
RigidGridMotion\_t :=
  {
  List( Descriptor\_t Descriptor1 ... DescriptorN ) ;            (o)
  RigidGridMotionType\_t RigidGridMotionType ;               (r)
  DataArray\_t<real,2,PhysicalDimension, 2>OriginLocation ;      (r)
  DataArray\_t<real,1,PhysicalDimension>RigidRotationAngle ;     (o/d)
  DataArray\_t<real,1,PhysicalDimension>RigidVelocity ;      (o)
  DataArray\_t<real,1,PhysicalDimension>RigidRotationRate ;      (o)
  List(DataArray\_t DataArray1 ... DataArrayN ) ;            (o)
  DataClass\_t DataClass ;                          (o)
  DimensionalUnits\_t DimensionalUnits ;                     (o)
  }
```

Definitions:

- RigidGridMotionType_t: enumeration type that describes the type of rigid grid motion.
- RigidGridMotionType_t := Enumeration(None, ConstantRate, VariableRate)
- OriginLocation: Physical coordinates of the origin before and after the rigid body motion.
- RigidRotationAngle: Rotation angles of about each axis of the translated coordinate system.
- RigidVelocity: Grid velocity vector of the origin translation.
- RigidRotationRate: Rotation rate vector about the about the axis of the translated coordinate system.

Notes:

- The DataClass_t, DimensionalUnits_t and Descriptor_t nodes may optionally be specified under the RigidGridMotion_t nodes.
- RigidGridMotionType, OriginLocation and RigidRotationAngle are the only three required data under RigidGridMotion_t. All other elements are optional.
- Any numbers of DataArray_t nodes are allowed. These should be used to record data not covered by this specification.
- RigidGridMotion_t implies relative motion of grid zones or blocks. However, no attempt is made here to require that the ZoneGridConnectivity_t information be updated to be consistent with the new grid locations. The user is responsible to ensure that any ZoneGridConnectivity_t information is kept up to date.

The mapping of the RigidGridMotion_t data structure onto the ADF database is illustrated in Figure 6.3.

6.4.4 The ArbitraryGridMotion_t data structure under the Zone_t data structure:

When a grid is in motion, it is often necessary to account for the position of each grid point as it deforms. When all grid points move at the same velocity, the grid keeps its original shape. This particular case of grid motion may be recorded under the RigidGridMotion_t data structure. On the other hand, if the grid points have different velocity, the mesh is deforming. The ArbitraryGridMotion_t data structure allows the CGNS file to contain information about arbitrary grid deformations. If not present, the grid is assumed to be rigid.

In addition to the creation of the ArbitraryGridMotion_t data structure to record the velocity of each grid point, CGNS has been enhanced to allow multiple GridCoordinates_t nodes under a Zone_t. This enables the storage of the instantaneous grid locations at different time steps or iterations. The original grid coordinates definition, as currently defined in the SIDS, remains unchanged with the name "GridCoordinates".

The arbitrary grid motion is recorded independently for each zone of the CGNS base. Therefore the ArbitraryGridMotion_t data structure is added under each the zone data structure (Zone_t). There may be zero to several ArbitraryGridMotion_t nodes under a Zone_t node. The multiple arbitrary grid motion definition may be associated to different iteration or time step in the computation. This association is recorded under the ZoneIterativeData_t data structure.

SIDS definition of the ArbitraryGridMotion_t data structure:

The ArbitraryGridMotion_t data structure under the Zone_t data structure:

```
Zone\_t<int CellDimension, int PhysicalDimension > :=
  {
  List( ArbitraryGridMotion\_t ArbitraryGridMotion1, ...,
ArbitraryGridMotionN ) ;        (o)
  List( GridCoordinates\_t<IndexDimension, VertexSize>
               GridCoordinates, GridCoordinates1, ...,
GridCoordinatesN ) ;                (o)
  ...
  }
```

The ArbitraryGridMotion_t data structure:

```
 ArbitraryGridMotion\_t :=
  {
  ArbitraryGridMotionType\_t ArbitraryGridMotionType ;           (r)
  List(DataArray\_t<real,1,VertexSize>
        GridVelocityX, GridVelocityY, ... ) ;           (o)
  List( Descriptor\_t Descriptor1 ... DescriptorN ) ;            (o)
  DataClass\_t DataClass ;                            (o)
  DimensionalUnits\_t DimensionalUnits ;                      (o)
  GridLocation\_t GridLocation ;                      (o/d)
  Rind\_t<IndexDimension> Rind                            (o/d)

  }
```

The DataArray_t nodes are used to store the components of the grid velocity vector. The table below lists the new data-name identifiers created to record these vectors in the cartesian, cylindrical and spherical coordinate systems.

Data-Name Identifier	Description	Units
GridVelocityX	x-component of grid velocity at a grid point	L/t
GridVelocityY	y-component of grid velocity at a grid point	L/t
GridVelocityZ	z-component of grid velocity at a grid point	L/t
GridVelocityR	R-component of grid velocity at a grid point	L/t
GridVelocityTheta	Theta-component of grid velocity at a grid point	σ/t
GridVelocityPhi	Phi-component of grid velocity at a grid point	σ/t
GridVelocityXi	Xi-component of grid velocity at a grid point	L/t
GridVelocityEta	Eta-component of grid velocity at a grid point	L/t
GridVelocityZeta	Zeta-component of grid velocity at a grid point	L/t

Definitions:

- ArbitraryGridMotionType_t is an enumeration type that describes the type of arbitrary grid motion. The type is either NonDeformingGrid or DeformingGrid. ArbitraryGridMotionType_t := Enumeration(NonDeformingGrid, DeformingGrid) ;

Notes:

- The only required element of the ArbitraryGridMotion_t data structure is the ArbitraryGridMotionType. Thus, even if a deforming grid application does not require the storage of grid velocity data, the ArbitraryGridMotion_t node must exist (with ArbitraryGridMotionType=DeformingGrid) to indicate that deformed grid points (GridCoordinate_t) exist for this zone.
- The DataClass_t, DimensionalUnits_t and Descriptor_t nodes may optionally be specified under the RigidGridMotion_t nodes.
- Point by point grid velocity implies a deformation (or potentially only motion) of the grid points relative to each other. Because the original grid coordinates definition is to remain unchanged with the name "GridCoordinates", any deformed coordinates are to be written with a different name (e.g., "GridCoordinates1" or another used-defined name) and are to be pointed to using GridCoordinatesPointers in the data structure ZoneIterativeData_t.
- Point by point grid velocity may also lead to relative motion of grid zones or blocks, or movement of grid along abutting interfaces. However, no attempt is made here to require that the ZoneGridConnectivity_t information be updated to be consistent with the new grid locations. The user is responsible to ensure that any ZoneGridConnectivity_t information is kept up to date.
- Rind is an optional field that indicates the number of rind planes included in the grid velocity data. It only applies to structured zones.

- The GridLocation specifies the location of the velocity data with respect to the grid; if absent, the data is assumed to coincide with grid vertices (i.e. GridLocation = Vertex).

The mapping of the ArbitraryGridMotion_t data structure onto the ADF database is illustrated in Figure 6.4.

6.5 Initial Proposal for Addition of the StructuredLevel_t node to CGNS

In order to support multi-grid, a proposal written by Dr. Christopher Rumsey was presented to the CGNS Steering Committee for evaluation. At the time of this writing, this proposal is considered ready for implementation into the CGNS documentation and software library.

It is proposed to create a new data structure named "StructuredLevel_t". This data structure would be optional and could be recorded under three different nodes, FlowSolution_t, DiscreteData_t and ZoneGridConnectivity_t. Its usage would be as follow:

```
Label =         StructuredLevel\_t Dimensions = 1 Dimension Values
=  IndexDimension Data =  Array of indices (integer)
```

indicating the grid level under the fine (GridCoordinates_t) level that the solution is being given, for each of the index dimensions. In 3D, this would be an array of 3 integers: indexi, indexj and indexk. In 2D this would be an array of 2 integers, indexi and indexj, and finally in 1D, only one index, indexi, would be required.

1. The number 1 indicates no change (fine level)
2. The number 2 indicates second level or every second point
3. The number 3 indicates third level or every fourth point
4. The number 4 indicates fourth level or every eighth point
5. Etc.

The values of indexi, indexj and indexk affect the size of the data arrays (VertexSize and CellSize) passed into and expected by the FlowSolution_t and DiscreteData_t nodes. Each new CellSize and VertexSize are determined by the following relations:

NewCellSize(n) = CellSize(n)/(2**(index(n)-1) NewVertexSize(n) = NewCellSize(n)+1

If the StructuredLevel_t node does not exist, then the fine level is assumed by default.

The implementation of the StructuredLevel_t node is planned for fall 2001 and therefore should be soon available for use with the ADPAC solver.

6.6 Conclusion

Most of the data recorded by ADPAC can be easily stored in the CGNS system. The main advantages of using the CGNS standard would be that the data could be interpreted by other codes and that all the data relative to a case could be recorded under a single file.

Two data structures, BaseIterativeData_t and ZoneIterativeData_t, were recently added to the CGNS system allowing ADPAC to record time dependant and iterative data. The support of multi-grid data is planned to be added to the CGNS system in the fall of 2001. The only remaining element missing to the CGNS system for a complete support of ADPAC is the parallelization of CGNS. This issue is currently discussed by the CGNS Committee but is still at a very early stage.

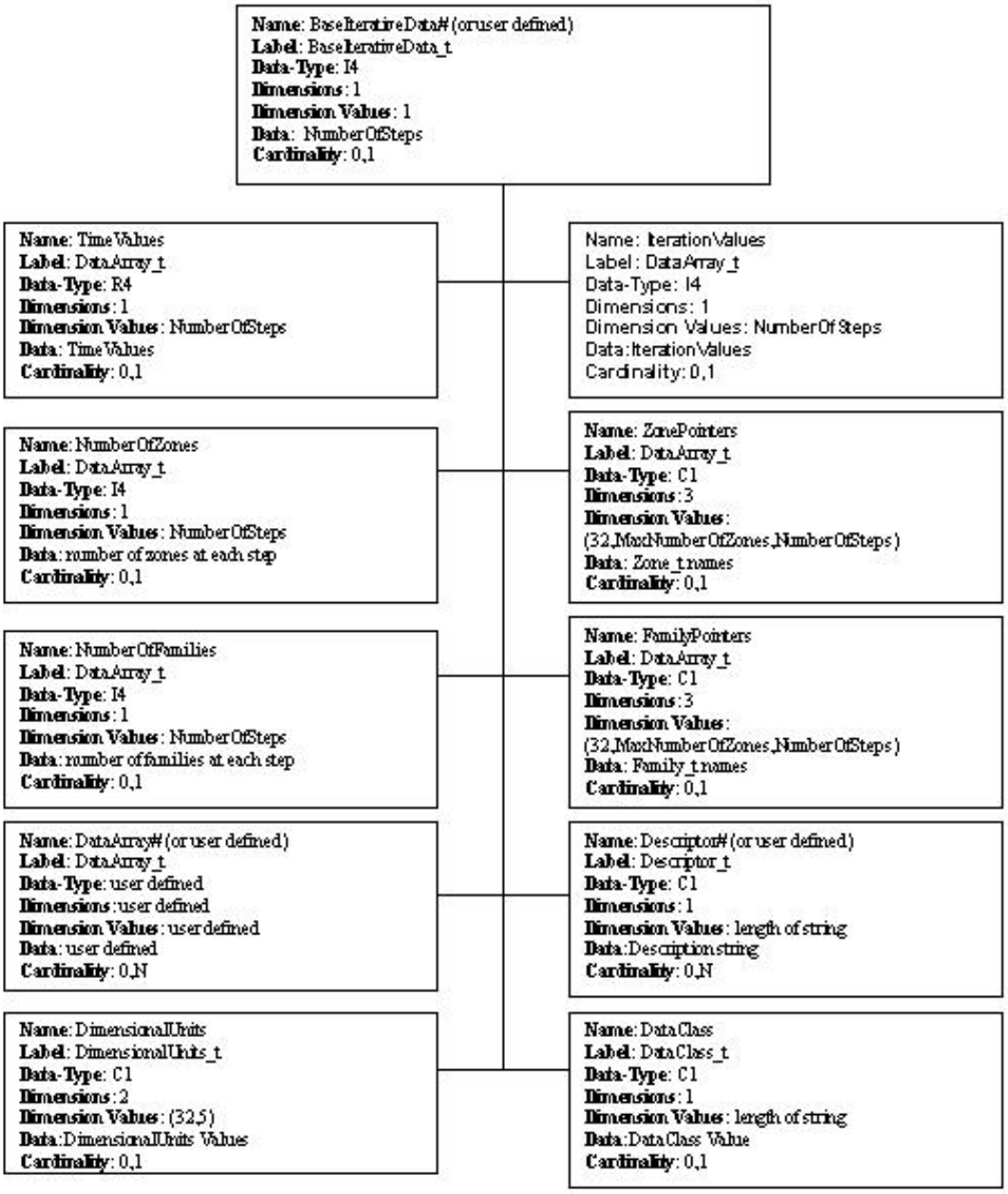

Figure 6.1: ADF file mapping definition of the BaseIterativeData_t data structure.

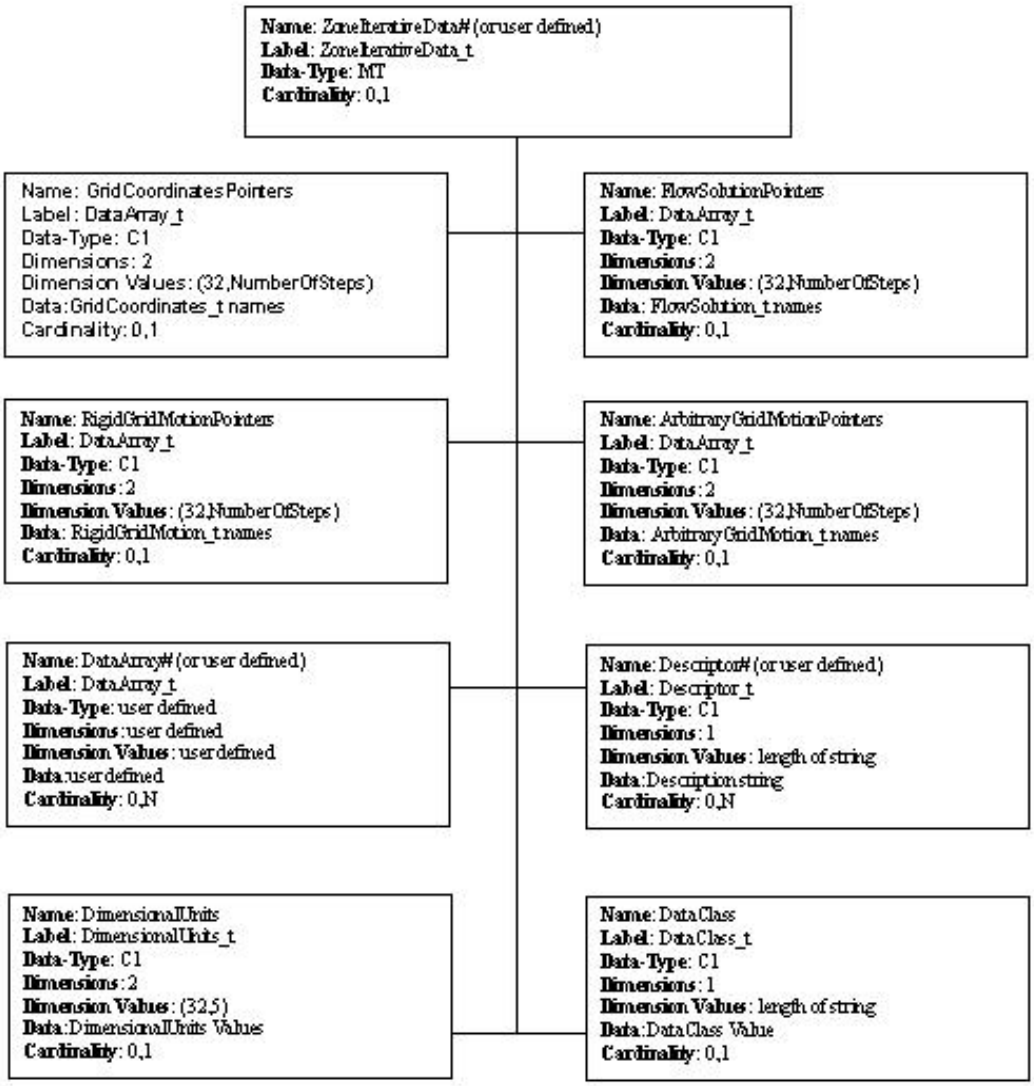

Figure 6.2: ADF file mapping definition of the ZoneIterativeData_t data structure.

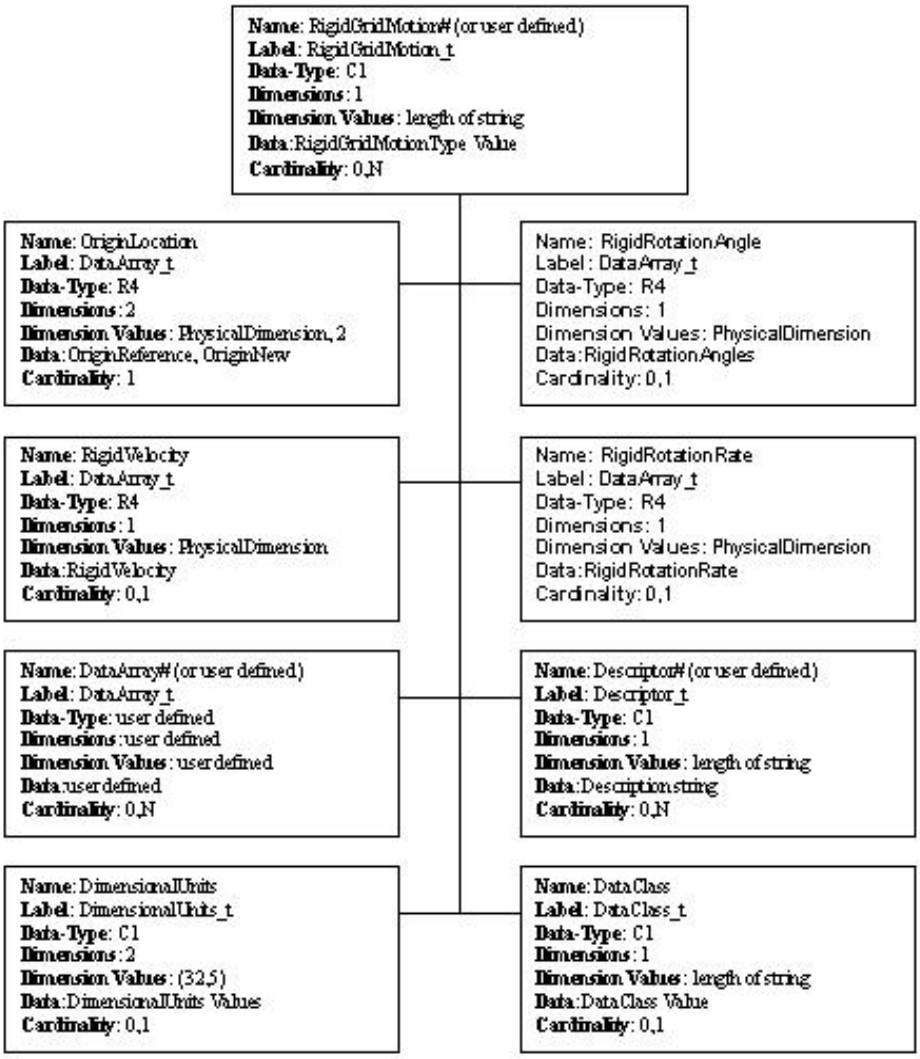

Figure 6.3: ADF file mapping definition of the RigidGridMotion_t data structure.

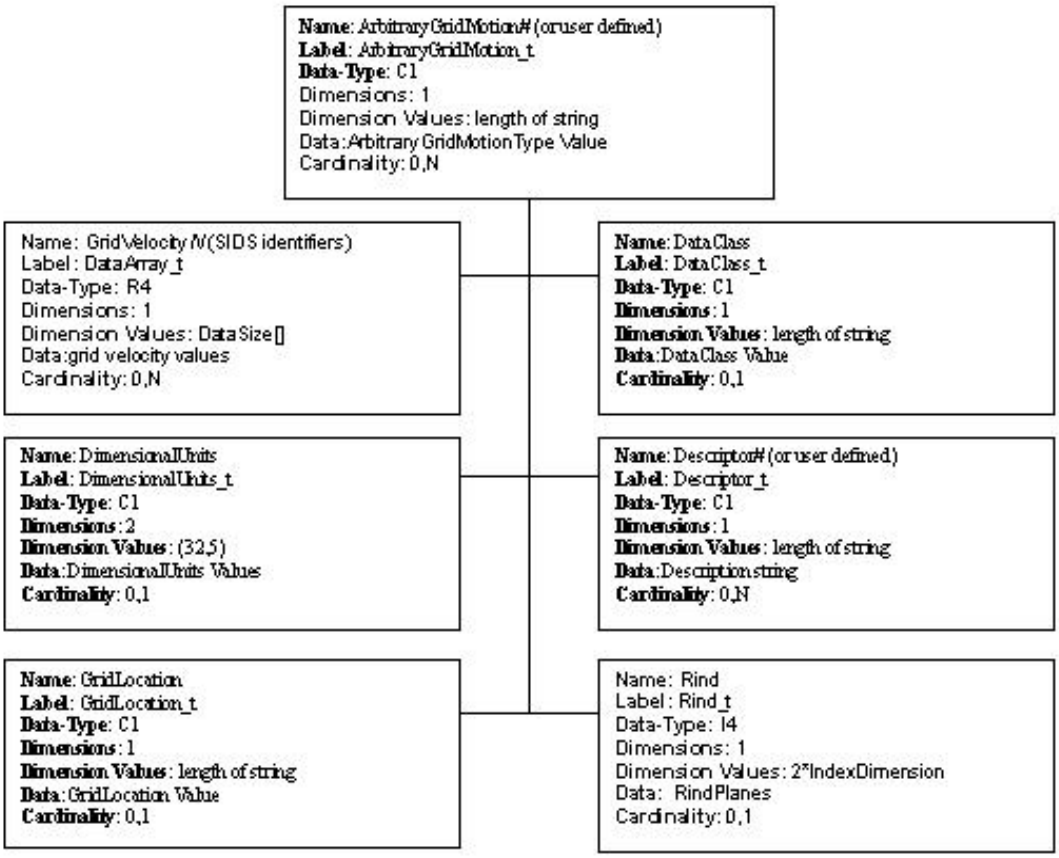

Figure 6.4: ADF file mapping definition of the ArbitraryGridMotion_t data structure.

Chapter 7

CONCLUSIONS

A multidisciplinary analysis software system was enabled for use in the Numerical Propulsion System Simulation (NPSS). The analysis was targeted for the combined aerodynamic and structural analysis of turbomachinery airfoils. Aerodynamic analysis was performed ADPAC CFD analysis code. Mesh generation for the aerodynamic analysis was based on a simplified algebraic H-grid mesh generator.

Structural analysis was based on the ANSYS finite element system. Automated procedures to import geometry, build a solid model, provide meshing, map aerodynamic and thermal data, and perform and output the structural analysis were derived from a script based on the ANSYS macro language.

Coupled aero/structural predictions were used to perform detailed deflection analysis of several test airfoil geometries. The airfoil deflection analysis is based on a two stage operation. Stage I of the analysis provides the necessary computations to determine the airfoil hot-to-cold deflections. This analysis determines the cold manufactured airfoil coordinates necessary to procude the desired (as-designed) hot running airfoil shape at the design condition. Stage II of the deflection analysis computes the deflections necessary to predict the operating shape of the airfoil at any operating condition. Stage II begins with the cold airfoil shape, and then iteratively computed the aerodynamic loads until a converged deflected shape is achieved. The procedure was tested and verified for the Rolls-Royce AE3007 fan, the Advanced Small Turboshaft Compressor 1st rotor, and a rotor from the aft end of an advanced, highly loaded HP core compressor. The multidisciplinary analysis provides a significant improvement in capability, resulting in a reduction in design/analysis time from one week to one day, relative to a 1997 baseline.

A probabilistic analysis was developed to asses the impact of changes in blade tip clearance on aerodynamic performance. A response surface was computed based on a design of experiments approach using the ADPAC aerodynamic analysis code. Predicted performance values along an engine operating line were computed for clearance values ranging from 0.025 to 0.080 inches. The resulting response surface was then used as input for the NESSUS probabilistic analysis code. This effort represents the first such applications of probabilistics to three-dimensional airfoil performance predictions.

The CAPRI library was imported into the ICEM CFD commercial mesh generation package as a means of demonstrating the viability of the CAPRI software in an industrial use tool. The implmentation permits native geometry access via the CAPRI programming interface.

The CFD General Notation System (CGNS) was evaluated to determine the suitability of this data standard for use in the ADPAC analysis code. Detailed studies of all available ADPAC input and output data streams were performed and mapped to the appropriate CGNS function. All ADPAC data elements can be successfully mapped to the CGNS standard except for electronic images. It was subsequently determined that the image files could ultimately be stored as a special data element under CGNS.

The automated deflection analysis capability was found to be very useful and fits in well with ANSYS capabilities. The widespread availability of ADPAC and ANSYS should make this a usable system for all US companies. Our survey of organizations in the aviation and space propulsion industries led to some useful, but limited information regarding input for geometry variability for probabilistic analysis. This would appear to be an area requiring additional work. This program provided a good starting point for overlap between aviation and space propulsion research activities. While aviation companies (GE, PW, RR) have historically been focused on aero performance as a priority, space propulsion designers are focused on forced response within the severe limited life rocket engine environment. Design trends in rocket turbopump blading are trending towards higher aspect ratio, high stage loading blading that has greater similarity to aviation aircraft engine turbomachinery blading. The Reusable Launch Vehicle (RLV) concept also implies the need for greater component life, and therefore accurate assessment of vibratory stress due to forced response has even greater value for both the aviation and space-based turbomachinery design community.

Our experience with the probabilistic analysis tool directly coupled within the aero/structural environment has led to some interesting conclusions. First, the technology to predict aerodynamic variability as a function of probabilistic geometry changes is an emerging technology and requires some added maturity before immediate use in industry. The current wave of probabilistic euphoria spurred by military development programs has greater utility in risk assessment for structural analysis. Our opinion is that the coupling of probabilistic and aerodynamic performance analyses requires further basic research before consideration as a significant portion of the NPSS system. A good deal of this fundamental research is already underway. Dr. David Darmofal (MIT University) recently published an interesting paper dealing with practical applications and implications of this type of analysis [7]. Given a baseline blade design, and a specified variability in the blade shape derived from statistical measurements, the CFD-based probabilistic analysis indicated that there was a 13 predicted "nominal" blade performance and the predicted baseline blade performance. Although, probabilistic tools themselves are changing rapidly, we have recently gained some added confidence that these tools can be effectively applied in a high fidelity multidisciplinary analysis environment.

An area that has broad applicability within both the aviation and space industries is forced response assessment of turbomachinery blade rows. The forced response analysis

problem requires accurate aerodynamic, structural, and probabilistic assessments. This would appear to be the next logical engineering problem of interest to extend the use of the multidisciplinary analysis.

References

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

[2] Hall, E. J. and Delaney, R. A., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task II - Unsteady Ducted Propfan Analysis - Final Report," NASA CR 187106, NASA Contract NAS3-25270, 1992.

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

[4] Hall, E. J., Topp, D. A., Heidegger, N. J., and Delaney, R. A., "Investigation of Advanced Counterrotation Blade Configuration Concepts for High Speed Turboprop Systems: Task VII - Endwall Treatment Inlet Flow Distortion Analysis Final Report", NASA Contract NAS3-25270, NASA CR-195468, July 1995.

[5] Hall, E. J., Heidegger, N. J., and Delaney, R. A., "Task 15 - ADPAC User's Manual", NASA Contract NAS3-27394, NASA CR 206600, January 1998.

[6] Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, AIAA 30th Aerospace Sciences Meeting & Exhibit, Reno, NV, January 1992.

[7] Garzon, V., and Darmofal, D., "Using Computational Fluid Dynamics in Probabilistic Engineering Design", AIAA Paper 2001-2526, 2001.

[8] Haimes, R., "Computational Analysis PRogramming Interface", Proceddings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations. July, 1998.

[9] CGNS Project Group, "The CFD General Notation System Standard Interface Data Structures", Version 2.0, February 2001; http://www.cgns.org/

[10] CGNS Project Group, "The CFD General Notation System Advanced Data Format (ADF) User's Guide", April 2001; http://www.cgns.org/

[11] CGNS Project Group, "The CFD General Notation System SIDS-to-ADF File Mapping Manual", Version 2.0, February 2001; http://www.cgns.org/

[12] Poirier, D., Allmaras, S., McCarthy, D., Smith, M., and Enomoto, F., "The CGNS System", AIAA Paper 98-3007, June 1998.

[13] CGNS Project Group, The CFD General Notation System Mid-Level Library", July 2001; http://www.cgns.org/

[14] Poirier, D., Bush, R., Cosner, R., Rumsey, C., and McCarthy, D., "Advances in the CGNS Database Standard for Aerodynamics and CFD", AIAA Paper 2000-0681, January 2000.

[15] Rumsey, C., Poirier, D., Bush, R., and Towne, C., "A User's Guide to CGNS", Version 1.0, August 2001; http://www.cgns.org/

Appendix A

FILE SPECIFICATIONS FOR MULTIDISCIPLINARY ANALYSIS

This Appendix describes the blade point data file and the blade data file for the multidisciplinary analysis.

A.1 File Specification for Blade Point File

The first line contains three integers, the number of blades, the number of slices and the number points per slice. The format is (i7,i7,i7).

The file then loops over the number of slices on the outer loop, and loops over the number of points per slice on the inner loop. Blade point coordinates are given in (x, y, z) coordinates. The format is (14.5f, 14.5f, 14.5f).

A.2 File Specification for Blade Data File

The first five lines of the blade point file are header lines for information, such as a title card, file author, application. These lines begin with the '#' symbol.

By convention, the sixth line has 'rpm=' followed by the RPM for blade in this file.

The seventh line has 'tip_gap=' followed by the tip gap for the blade (in inches).

The eigth line has the stack axis, or the amount the blade points should be translated in the axial direction. The seventh line has 'stackaxis=' followed by the stack axis (in inches).

The ninth line has a comment beginning with '#'. This will indicate the beginning of the hub definition. The tenth line is the number of points used to define the hub. The hub points will follow, listing the axial location and radius in inches.

The next line is a comment beginning with '#'. This will indicate the beginning of the

outer casing definition. The next line has the number of points used to define the casing. This will be followed the a listing of the axial location and radius in inches.

This is followed by a comment beginning with '#' to indicate the reference pressure in psi. The next line contains the value of PREF in psi.

This is followed by a comment beginning with '#' to indicate the reference temperature in Rankine. The next line contains the value of TREF in degrees Rankine.

This is followed by a comment beginning with '#' to indicate the ratio of specific heats, gamma. The next line contains the value of gamma.

This is followed by a comment to indicate the inlet profile, followed by the number of radial data points in the inlet profile, NDATA. Next is a comment to clarify what the different columns of data represent. This is followed by NDATA lines with a column for the radial value (in inches), the total pressure (in psi), the total temperature (in Rakine), the radial angle of the flow (in degrees), the tangential angle of the flow (in degrees) and a value of turbulence (non dimensional, used with the Spalart Allmaras model).

The next line is a comment beginning with '#' to indicate the exit static pressure at the hub, followed by the value of the exit static pressure at the hub in psi.

The next line is a comment beginning with '#'. This will indicate the beginning of the airfoil itself. The line has three integers, the number of blades, the number of slices, and the number of points on each slice. Each slice must have the same number of points.

The following lines contain one point on a slice. The format is two integers and three real values. The first two integers are the slice number and the point number. The three real values are the x, y and z coordinate in inches.

The blade point file contains a series of slices up the span of the blade. The first slice is the hub definition without a fillet, and the last slice is the blade profile at the blade tip. There should be no repeated node on a slice. Specifically, the first and last node of a slice should have different coordinate values.

A.3 Example File:

```
# Blade Title
# Created April 2001 for the X engine
# Author: Frank Engineer
# Improved design with lean and sweep
#
rpm=9500.0
tip_gap=0.0350
stackaxis=15.0
# hub definition
3
2.0 7.0
```

```
3.0 7.5
3.0  8.0
# casing definition
3
1.0  13.0
2.0  13.0
4.0 13.0
# PREF (psi)
14.7
# TREF (Rankine)
518.7
# Gamma, ratio of specific heats
1.4
# Inlet profile (in, psi, R, deg, deg)
3
# Rad Ptot Ttot  BetaR BetaT CHI
3.1 14.7 518.7 0.0 0.0 1.0
5.0 14.7 518.7 0.0 0.0 1.0
8.99 14.7 518.7 0.0 0.0 1.0
# Airfoil definition
26  21  256
1  1  -1.000  0.050  7.810
1  2  -1.101  0.051  7.810
etc
```

Appendix B

NPSS PHASE 1 ANSYS MACRO

This appendix contains a detailed listing of the *npss_Phase1.mac* ANSYS macro. This macro controls the execution of tasks under the NPSS Multidiscplinary Integration and Analysis Stage I (hot-to-cold deflection analysis).

```
!23456789.123456789.123456789.123456789.123456789.123456789.123456789.123456789.
!*********************************************************************************
!*                                                                               *
!*                        ROLLS-ROYCE AEROSPACE ANSYS MACRO                      *
!*                                                                               *
!*********************************************************************************
!* MACRO NAME: npss_phase1                                                       *
!* VERSION:    1A0                                                               *
!* REVISED:    31 AUG. 2001 J. Rasche  - created                                 *
!*                                                                               *
!*********************************************************************************
!* FUNCTION:    This program reads the blade_point.dat file and creates the hot*
!*              running position of the airfoil and meshes it.  The program      *
!*              maps on CFD temperatures and pressures from the ADPAC.ansys      *
!*              file.   The programs function is to calculate the cold shape of*
!*              the hot running airfoils and generate a new blade_point.dat      *
!*              file of the blade in it's cold shape.                            *
!* LIMITATIONS: The current limitation of this program is that it can only       *
!*              create the solid model inside of ANSYS.  The future call for     *
!*              this macro is to drive UG, PRO/E and CADDS5 in an automated      *
!*              fashion, building the 3D model in the CAD system and importing *
!*              the geometry with ANSYS connection products.                     *
!*                                                                               *
!*********************************************************************************
!* OTHER MACROS OR PROGRAMS USED BY THIS MACRO:                                  *
```

```
!*                                                                            *
!* cfd_wrapper.x, interp_deflect.x     No other macros called                 *
!*                                                                            *
!* INPUT AND OUTPUT FILES USED BY THIS MACRO:                                 *
!*                                                                            *
!* blade_point.dat, blade_data.dat, adpac.ansys, adpac.input, mater.mp        *
!* airfoile.surf, deflect_airfoil.surf, junk_data_jr                          *
!*                                                                            *
!****************************************************************************
!* INPUT PARAMETERS: RPM(1), XAXIS                                            *
!*                                                                            *
!* PARAMETERS CREATED:                                                        *
!* ARRAYS:                                                                    *
!*                                                                            *
!* JJ, bladept, afsdata, afedata, nlp, elp, vmask, vmaske, vmaskn, adpac,     *
!* adpace, pt_sli, rpm, invalue, inxyz, p_xyz, s_xyz, le_xyz, te_xyz, p_result *
!* s_results, l_result, t_result, cfd_p_x, cfd_s_x, cfd_p_v, cfd_s_v, surf_v  *
!* int_xyz, int_v, surf_x, surf_v, int_xyz, apply, onode, oelem, nnode, blade *
!* mask                                                                       *
!*                                                                            *
!* SCALAR:                                                                    *
!*                                                                            *
!* AFSIZE, BASIC, BLADE_IN, BLTOL, BN, CFD_TIME, CFL, CFMAX, CNPTS, CTAM, DIAM *
!* ELCT_, F1EQ ,FCART, FCOAG1, FCOAG2,FFULMG,FINVVI,FITCHK,FITFMG,FKINF,      *
!* FMIXLEN,FMULTI,FNCMAX,FREST,FTURBB ,FT_TIME,FWALLF,G ,GAMMA,HALF,HALF_PT   *
!* HBCT,HBMN,HUBCUT,HUBEN,HUBLEN,HUBRATIO,IN_CT,J,JUDGE,K,KN5,KN6,KP1,KP2,KPTG *
!* LEVEL2,LEVEL3,LEVEL4,LE_CT,LE_MIN,LLEN,LPAT,LTEDGE,M,MAX,MAXX,MAXY,MAXZ,   *
!* MINE,MINN,MIN_CT_N,MIN_N,MLN,MNSFE,MNSFN,MXNODE,MXOELEM,MXONODE,MXSFN,NME, *
!* NDCT,NDCT_,ND_CT_S,NLPAT,NNLP,NPTS,NSEC,N_PT,N_SLI,P3DPRT,PI,PREF,PRNO,PRTNO*
!* PS,P_CT,P_MIN,RAD_RPM,RGAS,RMACH,SFNCT,SMSIZE,SS,STARTNUM,ST_TIE,S_CT,S_MIN *
!* T,TARGET,TCF,TE_CT,TE_MIN ,TG1,TG2,TG3,TG4 ,TLINE,TREF,TTPUZ ,TUZ,T_TIME,  *
!* VIS2,VIS4,VPT,XAXIS,FRESID                                                 *
!*                                                                            *
!*                                                                            *
!****************************************************************************
!* COMPONENTS CREATED:                                                        *
!* abase, atop, base, int_n, le, le_a, press, press_a, suck, suck_a, surfnode *
!* takebase, taketop, te, te_a, top                                           *
!*                                                                            *
!*                                                                            *
!****************************************************************************
!* VERIFICATION PROBLEMS:  Type 4, LCC  and S42                               *
!****************************************************************************
fini
*get,st_cfd,active,,time,wall    ! gets start time off wall
```

```
/sys,~kzq0fd/wip/NPSS_RELEASE/SCRIPTS/cfd_wrapper.x blade_data.dat
*get,ft_cfd,active,,time,wall     ! gets finish time off wall
/nopr
/output,junk_data_jr   ! junk_data_jr file is for scrap output from ANSYS
/pmacro                 ! writes all of the commands in this macro to log file
*get,st_time,active,,time,wall    ! gets wall time
!/sys,audioplay -v 60 ~jz9hn2/sounds/npss_start.au
/color,pbak,off                    ! turns the background of ansys to black
fini                               ! makes sure macro goes to begin level
/prep7                             ! Enters /prep7 of ANSYS
*del,jj                            ! deletes jj array if it has been defined
*dim,jj,array,1,2                  ! Defines jj array
*vleng,1,1
*vread,jj(1,1),blade_point,dat     ! reads the number of sections per blade
(7X,F7.0)
*vleng,1,1
*vread,jj(1,2),blade_point,dat     ! reads the number of points per section
(14X,F7.0)
n_pt=jj(1,1)*jj(1,2)               ! calculates the total number of points
Nsec=jj(1,1)                       ! sets Nsec equal to number of sectoins
Npts=jj(1,2)                       ! sets Npts equal to number of points/sec.
Tg1=11                             ! hardware tangency points, can read in
Tg2=70                             ! from file in future release
Tg3=91
Tg4=150
*del,bladept
*dim,bladept,array,n_pt,3       !read in x,y,z points from file,skip 1st row
*vread,bladept(1,1),blade_point,dat,,,,,,1
(2X,F12.5)
*vread,bladept(1,2),blade_point,dat,,,,,,1
(16X,F12.5)
*vread,bladept(1,3),blade_point,dat,,,,,,1
(30X,F12.5)
m=1
b=0
k=1
*do,i,1,Nsec      ! makes kps at each section,offset by 1000
*do,g,1,Npts
k,m,bladept(k,1),bladept(k,2),bladept(k,3)
m=m+1
k=k+1
*enddo
m=1
b=b+1000
m=b+m
```

```
*enddo
/auto
kplo
!
!
/prep7
/go
/output,term
/com,*************************************************************************
/com,                    READING IN mater.mp MATERIAL FILE
/COM,*************************************************************************
/nopr
/output,junk_data_jr
/input,mater,mp   ! material properties file in ANSYS command format.
*SET,AFSIZE  , 0.4000000000000
*SET,huben   , 2.000000000000 ! Number of elements from hub to tangency pt.
*set,level2  , 5.00000000000  ! Number of elements in second level.
*set,level3  , 4.00000        ! Number of elements in 3rd level.
*set,level4  , 3.00000        ! Number of elements in 4th level.
*set,hublen , 8               ! Number of elements from edge to center.
*set,ltedge  , 2.0000         ! Number of elements through the LE and TE
*set,hubratio, 3              ! ratio of first element to last element in
*set,basic   , .3            ! Basic element size, global element size.
*SET,HUBCUT  ,  .25
*SET,TCF     , -459.69           ! Thermal correction factor
*SET,SMSIZE  ,  4.000000000000
*SET,STARTNUM,  60000.00000000
!
!
/uis,msgpop,3             !
/nerr,5,10000000          ! writes out 5 warning messages and 10000 error
                          ! messages to the window if running interactive
                          ! All warnings and errors are written to filename.err
/pnum,kpoi                ! Turns Keypoint number off for graphical display
/pnum,line,               ! Turns line numbers off for graphical display
/view,1,1,.3,.8           ! Sets view vector
/triad,lbot               ! Sets triad to lower left corner of window
/vup,1,z                  ! Sets Z axis up
/go
/output,term
/com,*************************************************************************
/com,                  CREATING SPLINES FOR VOLUME CREATION
/com,*************************************************************************
/nopr
/output,junk_data_jr
```

```
!
!
!
!
/auto
/title,Starting splining operation, %Npts% per section, %Nsec% number of sections
kplo
!
! The following section determines how many splines to make as it goes around
! the stream sections.
offset=0
p=1
cnpts=Npts-6
lpat=cnpts/5
lpat=lpat+1
nlpat=nint(lpat)
judge=lpat-nlpat
*if,judge,lt,0.0,then
   nlpat=nlpat-1
   *elseif,judge,gt,0.0,then
   nlpat=nlpat
  *elseif,judge,eq,0,then
  match=1
  *else
*endif
*do,p,1,Nsec,1
spn1=1+offset
spn2=2+offset
spn3=3+offset
spn4=4+offset
spn5=5+offset
spn6=6+offset
spn7=1+offset
i=1
*do,i,1,nlpat,1
spline,spn1,spn2,spn3,spn4,spn5,spn6
spn1=spn1+5
spn2=spn2+5
spn3=spn3+5
spn4=spn4+5
spn5=spn5+5
spn6=spn6+5
*enddo
*if,match,eq,1,then
   kptg=0
```

```
   *else
kptg=(Npts+offset)-(spn6-5)
 *endif
*if,kptg,eq,4,then
      spline,spn1,spn2,spn3,spn4,spn5,spn7
   *elseif,kptg,eq,3,then
      spline,spn1,spn2,spn3,spn4,spn7
   *elseif,kptg,eq,2,then
     spline,spn1,spn2,spn3,spn7
   *elseif,kptg,eq,1,then
     spline,spn1,spn2,spn7
   *elseif,kptg,eq,0,then
     spline,spn1,spn7
   *elseif,kptg,gt,6,then
*msg,warn
more than 6 keypoints left
*endif
offset=offset+1000
p=p+1
/auto
lplo
*enddo
/auto
/title,End of spline operation, moving on to cutting the hub tangency fillet
lplo
!
! End of the import stage.
!*********************************************************************
!
! Splines up the sides so that the Hub tangency point can be cut.
!
/go
/output,term
/com,*********************************************************************
/com,       SPLINING UP THE SIDES FOR HUB CUT AT HUB FILLET TANG. PT.
/com,*********************************************************************
/nopr
/output,junk_data_jr
/title, Splining up the sides of the airfoil
kn1=1
alls
kplo
*get,mln,line,,num,max
mln=mln+1
*do,i,1,Npts,1
```

```
spline,kn1,kn1+1000,kn1+2000,kn1+3000,kn1+4000,kn1+5000
kn1=kn1+1
*enddo
tline=mln
*do,i,1,Npts,1
*get,llen,line,tline,leng
ldiv,tline,hubcut/llen,,2,0  ! cuts the bottom spline at the hub tangency
! fillet point
tline=tline+5
*enddo
lsel,s,line,,mln,400000,1
ldel,all
alls
lplo
!
!
! Splines the tangency section.
!
offset=0
spn1=Npts+1
spn2=Npts+2
spn3=Npts+3
spn4=Npts+4
spn5=Npts+5
spn6=Npts+6
spn7=spn1
*do,i,1,nlpat,1
spline,spn1,spn2,spn3,spn4,spn5,spn6
        spn1=spn1+5
        spn2=spn2+5
        spn3=spn3+5
        spn4=spn4+5
        spn5=spn5+5
        spn6=spn6+5
*enddo
!*if,match,eq,1,then
!   kptg=0
!   *else
!kptg=(Npts+offset)-(spn6-5)
! *endif
*if,kptg,eq,4,then
      spline,spn1,spn2,spn3,spn4,spn5,spn7
   *elseif,kptg,eq,3,then
      spline,spn1,spn2,spn3,spn4,spn7
   *elseif,kptg,eq,2,then
```

```
     spline,spn1,spn2,spn3,spn7
   *elseif,kptg,eq,1,then
     spline,spn1,spn2,spn7
   *elseif,kptg,eq,0,then
     spline,spn1,spn7
   *elseif,kptg,gt,6,then
*else
*endif
/auto
/title, Combining spline lines
lplo
/go
/output,term
/com,*************************************************************************
/com,              COMBINING SPLINE LINES FOR SOLID MODEL CREATION...
/com,*************************************************************************
/nopr
/output,junk_data_jr
! combining spline lines
/nopr
kn1=Tg1
kn2=Tg2
kn3=Tg3
kn4=Tg4
ksel,s,kp,,1,Tg1
ksel,a,kp,,Tg4,Npts,
lslk,s,1
lcomb,all
ksel,s,kp,,Tg1,Tg2
lslk,s,1
lcomb,all
ksel,s,kp,,Tg2,Tg3
lslk,s,1
lcomb,all
ksel,s,kp,,Tg3,Tg4
lslk,s,1
lcomb,all
ksel,s,kp,,1+Npts,Tg1+Npts
ksel,a,kp,,Tg4+Npts,Npts+Npts,
lslk,s,1
lcomb,all
ksel,s,kp,,Tg1+Npts,Tg2+Npts
lslk,s,1
lcomb,all
ksel,s,kp,,Tg2+Npts,Tg3+Npts
```

```
lslk,s,1
lcomb,all
ksel,s,kp,,Tg3+Npts,Tg4+Npts
lslk,s,1
lcomb,all
kn1=Tg1+1000
kn2=Tg2+1000
kn3=Tg3+1000
kn4=tg4+1000
kn5=1001
kn6=Npts+1000
*do,i,1,Nsec-1,1
ksel,s,kp,,kn5,kn1
ksel,a,kp,,kn4,kn6
lslk,s,1
lcomb,all
ksel,s,kp,,kn1,kn2
lslk,s,1
lcomb,all
ksel,s,kp,,kn2,kn3
lslk,s,1
lcomb,all
ksel,s,kp,,kn3,kn4
lslk,s,1
lcomb,all
kn1=kn1+1000
kn2=kn2+1000
kn3=kn3+1000
kn4=kn4+1000
kn5=kn5+1000
kn6=kn6+1000
*enddo
numcp,kpoi
/pnum,kpoi,1
alls
/auto
lplo
!  spline up the new 4 sides to get shape of airfoil
numcp,all
/auto
lplo
alls
bn=1
*do,i,1,4,1
kn1=bn
```

```
flst,2,Nsec+1,3
*do,j,1,Nsec+1,1
fitem,2,kn1
kn1=kn1+4
*enddo
spline,P51X
bn=bn+1
*enddo
/pnum,line
/pnum,kpoi
/auto
alls
/auto
kn1=1
kn2=2
kn3=3
kn4=4
/title, Creating the volumes used for meshing and meshing with element 95
lplo
/go
/output,term
/com,*************************************************************************
/com,                    CREATING VOLUMES FOR BRICK MESHING......
/com,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,Nsec,1
v,kn1,kn2,kn3,kn4,kn1+4,kn2+4,kn3+4,kn4+4
kn1=kn1+4
kn2=kn2+4
kn3=kn3+4
kn4=kn4+4
*enddo
ksel,s,kp,,1,
ksel,a,kp,,4
lslk,s,1
lesize,all,,,2
ksel,s,kp,,1,2
lslk,s,1
lesize,all,,,15
ksel,s,kp,,2,3
lslk,s,1
lesize,all,,,2
ksel,s,kp,,3,4
lslk,s,1
```

```
lesize,all,,,15
alls
esize,distkp(4,(Nsec+1)*4)/18
shpp,warn
et,1,95
/go,
/output,term
/com,*************************************************************************
/com,                          MESHING VOLUMES, PLEASE WAIT......
/COM,*************************************************************************
/nopr
/output,junk_data_jr
vmesh,all
/title,Meshing completed
/color,num,oran,1
eplo
/output,term
/com,*************************************************************************
/com,                          MESHING VOLUMES, PLEASE WAIT......
/COM,*************************************************************************
/com,*************************************************************************
/com,                     CREATING COMPONENTS PRESS, SUCK, LE AND TE
/com,*************************************************************************
/nopr
/output,junk_data_jr
!
/com, selecting suction side areas for creation of nodal component
!
kp1=1
kp2=2
ksel,s,kp,,kp1,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,kp2
*enddo
lskp,1
arls,1
cm,suck_a,area
nsla,s,1
nplo
cm,suck,node
!
/com, created suck side area and nodal component
!
```

```
/com, starting on TE side
kp1=2
kp2=3
ksel,s,kp,,kp1,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,kp2
*enddo
lskp,1
arls,1
cm,te_a,area
nsla,s,1
nplo
cm,te,node
!
/com, finished TE side area and nodal component
!
/com, starting on pressure side
kp1=3
kp2=4
ksel,s,kp,,kp1,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,kp2
*enddo
lskp,1
arls,1
cm,press_a,area
nsla,s,1
nplo
cm,press,node
!
/com, finished pressure side area and nodal components
!
/com, starting on LE side
kp1=1
kp2=4
ksel,s,kp,,kp1
ksel,a,kp,,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,
```

```
ksel,a,kp,,kp2
*enddo
lskp,1
arls,1
cm,le_a,area
nsla,s,1
nplo
cm,le,node
/com, finished TE side area and nodal components
cmsel,s,press
cmsel,a,suck
cmsel,a,le
cmsel,a,te
cm,surfnode,node
alls
! deleting parameters
b=
ecount=
emin=
i=
kn1=
kn2=
kn3=
kn4=
ncount=
nmin=
nn1=
nn2=
nn3=
nn4=
np1x=
np1z=
np2x=
np2z=
offset=
p=
spn1=
spn2=
spn3=
spn4=
spn5=
spn6=
spn7=
x=
y=
```

```
/go
/output,term
/com,*************************************************************************
/com,                  SOLID MODEL AND MESH CREATION COMPLETED....
/com,*************************************************************************
/nopr
/output,junk_data_jr
alls
eplo
! add surface mesh for deflect_airfoil.surf file
/go
/output,term
/com,*************************************************************************
/com,                      WRITING OUT THE AIRFOIL.SURF FILE
/COM,*************************************************************************
/nopr
/output,junk_data_jr
fini
/prep7
alls
et,2,22
type,2
esurf
cmsel,s,surfnode
enode,1
cm,mask,elem
esel,s,type,,2
cmsel,u,mask
edel,all
cmsel,s,mask
*get,sfnct,node,,count
*get,ctam,elem,,count
*del,afsdata
*del,afedata
*dim,afsdata,array,sfnct,4,3
*dim,afedata,array,ctam,5
*get,mxsfn,node,,num,max
*get,mnsfn,node,,num,min
*get,mnsfe,elem,,num,min
*do,i,1,sfnct
afsdata(i,1,1)=mnsfn
afsdata(i,2,1)=nx(mnsfn)
afsdata(i,3,1)=ny(mnsfn)
afsdata(i,4,1)=nz(mnsfn)
mnsfn=ndnext(mnsfn)
```

```
*enddo
*do,t,1,ctam
afedata(t,1)=mnsfe
afedata(t,2)=nelem(mnsfe,1)
afedata(t,3)=nelem(mnsfe,2)
afedata(t,4)=nelem(mnsfe,3)
afedata(t,5)=nelem(mnsfe,4)
mnsfe=elnext(mnsfe)
*enddo
*cfopen,airfoil,surf
*vwrite,sfnct
(F7.0,tl1,' ')
*vwrite,afsdata(1,1,1),afsdata(1,2,1),afsdata(1,3,1),afsdata(1,4,1)
(F7.0,tl1,' ',3F12.6)
*vwrite,ctam,
(F7.0,tl1,' ')
*vwrite,afedata(1,1),afedata(1,2),afedata(1,3),afedata(1,4),afedata(1,5),
(F7.0,tl1,' ',F12.0,tl1,' ',F12.0,tl1,' ',F12.0,tl1,' ',F12.0,tl1,' ')
*cfclose,airfoil,surf
esel,s,type,,2
edel,all
alls
etdel,2
/go
/output,term
/com,*********************************************************************
/com,              FINISHED WITH WRITING AIRFOIL.SURF FILE
/com,*********************************************************************
/nopr
/output,junk_data_jr
/go
/output,term
/com,*********************************************************************
/com,             READING IN ADPAC.ANSYS AND ADPAC.INPUT FILES
/COM,*********************************************************************
/nopr
/output,junk_data_jr
fini
/prep7
*del,nlp    ! number of lines of point data, first item in file
*del,elp    ! number of lines of element data, nlp+2 location in file
*del,vmask
*del,vmaske
*del,vmaskn
*del,adpac
```

```
*del,adpace
*del,pt_sli
nlp=
elp=
!
!           read nodal data
!
*dim,nlp,array,1,1            ! sets up initial array to get node count
*dim,pt_sli,array,1,1
*vread,nlp(1,1),adpac,ansys   ! Reads the first line of adpac.ansys file
(F10.0)
*vread,pt_sli(1,1),adpac,ansys  ! Reads number of points per slice
(10X,F10.0)
nnlp=nlp(1,1)+1               ! Adds one to it to include itself
*dim,vmaskn,array,nnlp,1      ! Creates node masking array
*dim,adpac,array,nnlp,6       ! Creates adpac array
*vfill,vmaskn(1,1),ramp,1,1   ! fills array with + 1's
vmaskn(1,1)=-1                ! changes the first row to -1,skips node count #
*vmask,vmaskn(1,1)
*vread,adpac(1,1),adpac,ansys
(F7.0)
*vmask,vmaskn(1,1)
*vread,adpac(1,2),adpac,ansys
(7X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,3),adpac,ansys
(19X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,4),adpac,ansys
(31X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,5),adpac,ansys
(43X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,6),adpac,ansys
(55X,F10.6)
!
*del,rpm
*dim,rpm,,5
/input,adpac,input
*del,invalue
*del,inxyz
*dim,invalue,array,nlp(1,1),2      ! array for in values on moper command
*dim,inxyz,array,nlp(1,1),3       ! array for xyz values for in values
*voper,invalue(1,1),adpac(2,5),add,
```

```
*voper,invalue(1,2),adpac(2,6),add,
*voper,inxyz(1,1),adpac(2,2),add,-xaxis
*voper,inxyz(1,2),adpac(2,3),add
*voper,inxyz(1,3),adpac(2,4),add
!
! add in array operations to move the data to the right array
!
cmsel,s,press
*get,p_ct,node,,count
*get,p_min,node,,num,min
cmsel,s,suck
*get,s_ct,node,,count
*get,s_min,node,,num,min
cmsel,s,le
*get,le_ct,node,,count
*get,le_min,node,,num,min
cmsel,s,te
*get,te_ct,node,,count
*get,te_min,node,,num,min
!
*del,p_xyz
*del,s_xyz
*del,le_xyz
*del,te_xyz
*del,p_result
*del,s_result
*del,l_result
*del,t_result
*del,cfd_p_x
*del,cfd_s_x
*del,cfd_p_v
*del,cfd_s_v
*del,surf_v
*del,surf_x
*del,int_xyz
*del,int_v
*del,surf_x
*del,surf_v
*del,int_xyz
*del,int_v
!
*dim,p_xyz,array,p_ct,3        ! pressure side xyz
*dim,p_result,array,p_ct,3     ! pressure side results
*dim,s_xyz,array,s_ct,3        ! suction side xyz
*dim,s_result,array,s_ct,3     ! suction side results
```

```
*dim,le_xyz,array,le_ct,3
*dim,l_result,array,le_ct,3
*dim,te_xyz,array,te_ct,3
*dim,t_result,array,te_ct,3
!
n_sli=nlp(1,1)/pt_sli(1,1)     ! calculates number of slices in cfd data
half_pt=pt_sli(1,1)/2          ! calculates the number of points per half
                               ! of a slice
half=half_pt*n_sli
!
*dim,cfd_p_x,array,half,3
*dim,cfd_s_x,array,half,3
*dim,cfd_p_v,array,half,2
*dim,cfd_s_v,array,half,2
!
! fill p,s,le and te xyz and result arrays
!
! get the xyz locations and node number of nodes on pressure side
cmsel,s,press
m=p_min
*do,i,1,p_ct
p_xyz(i,1)=nx(m)
p_xyz(i,2)=ny(m)
p_xyz(i,3)=nz(m)
p_result(i,3)=m        ! node number in column 3
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the suction side
cmsel,s,suck
m=s_min
*do,i,1,s_ct
s_xyz(i,1)=nx(m)
s_xyz(i,2)=ny(m)
s_xyz(i,3)=nz(m)
s_result(i,3)=m        ! node number in column 3
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the LE  side!
cmsel,s,le
m=le_min
*do,i,1,le_ct
le_xyz(i,1)=nx(m)
le_xyz(i,2)=ny(m)
```

```
le_xyz(i,3)=nz(m)
l_result(i,3)=m        ! node number in column 3m
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the TE side
cmsel,s,te
m=te_min
*do,i,1,te_ct
te_xyz(i,1)=nx(m)
te_xyz(i,2)=ny(m)
te_xyz(i,3)=nz(m)
t_result(i,3)=m      ! node number in column 3
m=ndnext(m)
*enddo
!
alls
!
k=1   $ps=1  $ss=1
*do,i,1,n_sli
  *do,j,1,half_pt
     cfd_s_x(ss,1)=inxyz(k,1)
     cfd_s_x(ss,2)=inxyz(k,2)
     cfd_s_x(ss,3)=inxyz(k,3)
     cfd_s_v(ss,1)=invalue(k,1)   ! pressure
     cfd_s_v(ss,2)=invalue(k,2)   ! temperature
     k=k+1
     ss=ss+1
  *enddo
  *do,g,1,half_pt
    cfd_p_x(ps,1)=inxyz(k,1)
    cfd_p_x(ps,2)=inxyz(k,2)
    cfd_p_x(ps,3)=inxyz(k,3)
    cfd_p_v(ps,1)=invalue(k,1)
    cfd_p_v(ps,2)=invalue(k,2)
    k=k+1
    ps=ps+1
  *enddo
*enddo
!
! mapping of temperatures and pressures
!

/go
/output,term
```

```
/com,*************************************************************************
/com,     MAPPING TEMPERATURE AND PRESSURE FROM CFD TO STRUCTURAL MODEL
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*moper,p_result(1,2),p_xyz(1,1),map,cfd_p_v(1,2),cfd_p_x(1,1),0   ! Temps
!
*moper,p_result(1,1),p_xyz(1,1),map,cfd_p_v(1,1),cfd_p_x(1,1),0   ! Press
!
*moper,s_result(1,2),s_xyz(1,1),map,cfd_s_v(1,2),cfd_s_x(1,1),0   ! Temps
!
*moper,s_result(1,1),s_xyz(1,1),map,cfd_s_v(1,1),cfd_s_x(1,1),0   ! Press
!
*moper,l_result(1,2),le_xyz(1,1),map,invalue(1,2),inxyz(1,1),0    ! Temps
!
*moper,l_result(1,1),le_xyz(1,1),map,invalue(1,1),inxyz(1,1),0    ! Press
!
*moper,t_result(1,2),te_xyz(1,1),map,invalue(1,2),inxyz(1,1),0    ! Temps
!
*moper,t_result(1,1),te_xyz(1,1),map,invalue(1,1),inxyz(1,1),0    ! Press
!
alls
fini
/solu
!
!
! put on the temperature
!
/go
/output,term
/com,*************************************************************************
/com,                        APPLYING TEMPERATURES
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,p_ct
bf,p_result(i,3),temp,p_result(i,2)
*enddo
*do,i,1,s_ct
bf,s_result(i,3),temp,s_result(i,2)
*enddo
*do,i,1,le_ct
bf,l_result(i,3),temp,l_result(i,2)
*enddo
*do,i,1,te_ct
```

```
bf,t_result(i,3),temp,t_result(i,2)
*enddo
!
!
/go
/output,term
/com,*************************************************************************
/com,                          CALCULATING PRESSURES
/COM,*************************************************************************
/NOPR
/output,junk_data_jr
cmsel,s,surfnode
*get,min_n,node,,num,min
*get,nd_ct_s,node,,count
*dim,surf_x,array,nd_ct_s,3
*dim,surf_v,array,nd_ct_s,1
m=min_n
*do,i,1,nd_ct_s
surf_x(i,1)=nx(m)
surf_x(i,2)=ny(m)
surf_x(i,3)=nz(m)
*get,surf_v(i),node,m,ntem,bfe    ! get nodal temperatures from surface
m=ndnext(m)
*enddo
!
alls
cmsel,u,surfnode
*get,in_ct,node,,count
*get,min_ct_n,node,,num,min
cm,int_n,node
*dim,int_xyz,array,in_ct,3
*dim,int_v,array,in_ct,2
!
m=min_ct_n
*do,i,1,in_ct,
int_xyz(i,1)=nx(m)
int_xyz(i,2)=ny(m)
int_xyz(i,3)=nz(m)
int_v(i,2)=m        ! node number in column 2
m=ndnext(m)
*enddo
!
*moper,int_v(1,1),int_xyz(1,1),map,surf_v(1,1),surf_x(1,1),0    ! Temps
!
/go
```

```
/output,term
/com,*********************************************************************
/com,                       APPLYING INTERNAL TEMPERATURES
/COM,*********************************************************************
/nopr
/output,junk_data_jr
*do,i,1,in_ct
bf,int_v(i,2),temp,int_v(i,1)
*enddo
alls
/auto
/pbf,temp,1
eplo
/nopr
/output,junk_data_jr
fini
/prep7
ksel,s,kp,,1,4
lslk,s,1
asll,s,1
nsla,s,1
cm,base,node   !create a component on nodes at the base / hub.
cm,abase,area  !create a component of areas at the base / hub.
nsla,
cm,takebase,node
ksel,s,kp,,(Nsec*4)+1,(Nsec*4)+4,1
lslk,s,1
asll,s,1
nsla,s,1
cm,top,node   ! create a component of nodes at the top / tip.
cm,atop,area  ! create a component of areas at the top / tip.
nsla,
cm,taketop,node
cmsel,s,base
cmsel,a,top
/auto
nplot
alls
eplo
! put on boundary conditions for the calculation of the thermal pinch.
cmsel,s,base
csys,0
!d,all,uz
ksel,s,kp,,1,3,2,
nslk,s
```

```
d,all,uy
cmsel,s,base
tuz=0
*get,hbct,node,,count
*get,hbmn,node,,num,min
*do,i,1,hbct
ttpuz=NZ(hbmn)
tuz=tuz+ttpuz
hbmn=ndnext(hbmn)
*enddo
tuz=tuz/hbct
target=node(0,0,tuz)
d,target,ux,0
!d,target,uz,0
cmsel,s,base
d,all,uz,0
cmsel,s,base
/pbc,u,1
nplo
/pbc,u,1
nplo
/view,1,1,.3,1
fini
/solu
time,1
NLGEOM,0
NROPT,AUTO, ,OFF
LUMPM,0
EQSLV,sparse,1E-6
SSTIF
PSTRES
tref,70
esel,s,type,,1
nelem
/go
/output,term
/com,*************************************************************************
/com,                  RUNNING A STRUCTURAL ANALYSIS TO CALCULATE
/com,                         CUT BOUNDARY DISPLACEMENTS
/COM,*************************************************************************
/nopr
/output,junk_data_jr
solve
cmsel,s,base
*get,ncount,node,,count
```

```
*get,nmin,node,0,num,min
/title,Putting on calculated displacements from thermal pinch analysis.
nplo
! putting on the calculated displacements
*do,i,1,ncount,1
d,nmin,ux,ux(nmin)
d,nmin,uy,uy(nmin)
d,nmin,uz,uz(nmin)
nmin=ndnext(nmin)
*enddo
alls
/solu
solve
fini
/prep7
/pbc,all
/pbf,temp,1  ! turn on temperatures for plot
eplo
/title,loading model with aero loads, please wait......
nplo
csys,0
!
! move the data from p_results,s_result,l_result and t_result into apply array
!
cmsel,s,surfnode
*get,max,node,,num,max
*del,apply
*dim,apply,array,max,1
*do,i,1,p_ct
apply(%p_result(i,3)%,1)=p_result(i,1)
*enddo
*do,i,1,s_ct
apply(%s_result(i,3)%,1)=s_result(i,1)
*enddo
*do,i,1,le_ct
apply(%l_result(i,3)%,1)=l_result(i,1)
*enddo
*do,i,1,te_ct
apply(%t_result(i,3)%,1)=t_result(i,1)
*enddo
!
eall
cmsel,s,surfnode
sffun,press,apply(1,1)
sf,all,press
```

```
alls
/solu
/go
/output,term
/com,*************************************************************************
/com,           RUNNING ANALYSIS WITH PRESSURE AND TEMPERATURES APPLIED
/COM,*************************************************************************
/NOPR
/output,junk_data_jr
ANTYPE,0
time,1
deltim,1,1,1
NLGEOM,on
NROPT,AUTO, ,off
LUMPM,0
EQSLV,sparse
autots,on
SSTIF,1,
TOFFST,0,
KBC,0
csys
*afun,rad
pi=acos(-1)
rad_rpm=rpm(1,1)
rad_rpm=rad_rpm/60*2*pi
rad_rpm=rad_rpm*(-1)
omega,rad_rpm,
esel,s,type,,1
nelem
solve
FINISH
blade_in=1
*afun,deg
/post1
/efacet,4
esel,s,type,,1
nelem
/view,1,-1,-1.5,1
plns,s,eqv
b=
ecount=
emin=
i=
kn1=
kn2=
```

```
kn3=
kn4=
ncount=
nmin=
nn1=
nn2=
nn3=
nn4=
np1x=
np1z=
np2x=
np2z=
offset=
p=
pl1=
pl2=
pl3=
pl4=
pmaxnode=
rotline=
slopehub=
spn1=
spn2=
spn3=
spn4=
spn5=
spn6=
spn7=
x=
y=
/go
/output,term
/com,*********************************************************************
/com, COMPLETED INITIAL STRUCTURAL ANALYSIS OF BLADE, MOVING ON TO CALCULATE
/COM, THE COLD SHAPE.  AIRFOIL.SURF AND DEFLECT_AIRFOIL.SURF WILL BE CREATED
/COM,*********************************************************************
/NOPR
/output,junk_data_jr
/output,term
/nopr
!/sys,audioplay -v 60 ~jz9hn2/sounds/npss_conv_start.au &
bltol=0.005
/go,
*msg,info,bltol
THE CONVERGENCE TOLERANE IS SET TO %g
```

```
/nopr
/output,junk_data_jr
fini
/solu
/output,term
/go
/com,*********************************************************************
/com, RUNNING INITIAL ANALYSIS TO GET GUESS POSITION
/com,*********************************************************************
/nopr
/output,junk_data_jr
esel,s,type,,1
nelem
time,1
EQSLV,Sparse
NLGEOM,ON
AUTOTS,ON
deltime,.2,.01,.5,
sstif,on
solve
fini
/prep7
modmsh,detach
fini
/post1
esel,s,type,,1
nelem
!
! get orig. node and element inof
!
fini
/prep7
esel,s,type,,1
nelem
*get,ndct_,node,,count
*get,elct_,elem,,count
*get,mxonode,node,,num,max
*get,mxoelem,elem,,num,max
*get,minn,node,,num,min
*get,mine,elem,,num,min
*del,onode
*del,oelem
*del,nnode
*dim,onode,array,ndct_,4
*dim,oelem,array,elct_,21
```

```
*dim,nnode,array,ndct_4,4
*do,i,1,ndct
onode(i,1)=minn
*get,onode(i,2),node,minn,loc,x
*get,onode(i,3),node,minn,loc,y
*get,onode(i,4),node,minn,loc,z
minn=ndnext(minn)
*enddo
*get,minn,node,,num,min
*do,i,1,elct
oelem(i,1)=mine
*get,oelem(i,2),elem,mine,node,1
*get,oelem(i,3),elem,mine,node,2
*get,oelem(i,4),elem,mine,node,3
*get,oelem(i,5),elem,mine,node,4
*get,oelem(i,6),elem,mine,node,5
*get,oelem(i,7),elem,mine,node,6
*get,oelem(i,8),elem,mine,node,7
*get,oelem(i,9),elem,mine,node,8
*get,oelem(i,10),elem,mine,node,9
*get,oelem(i,11),elem,mine,node,10
*get,oelem(i,12),elem,mine,node,11
*get,oelem(i,13),elem,mine,node,12
*get,oelem(i,14),elem,mine,node,13
*get,oelem(i,15),elem,mine,node,14
*get,oelem(i,16),elem,mine,node,15
*get,oelem(i,17),elem,mine,node,16
*get,oelem(i,18),elem,mine,node,17
*get,oelem(i,19),elem,mine,node,18
*get,oelem(i,20),elem,mine,node,19
*get,oelem(i,21),elem,mine,node,20
mine=elnext(mine)
*enddo
*get,mine,node,,num,min
fini
/post1
cmsel,u,base
*del,blade
*del,mask
*get,mxnode,node,,num,max
*get,ndct,node,,count
*get,name,active,,jobname
*dim,blade,array,mxnode,60,3
*dim,mask,array,mxnode,1,3
*vget,mask(1,1,1),node,,nsel
```

```
*vget,mask(1,1,2),node,,nsel
*vget,mask(1,1,3),node,,nsel
*vget,blade(1,1,1),node,1,loc,x
*vget,blade(1,1,2),node,1,loc,y
*vget,blade(1,1,3),node,1,loc,z
*vget,blade(1,2,1),node,1,u,x
*vget,blade(1,2,2),node,1,u,y
*vget,blade(1,2,3),node,1,u,z
*vmask,mask(1,1,1)
*voper,blade(1,3,1),blade(1,1,1),sub,blade(1,2,1)
*vmask,mask(1,1,2)
*voper,blade(1,3,2),blade(1,1,2),sub,blade(1,2,2)
*vmask,mask(1,1,3)
*voper,blade(1,3,3),blade(1,1,3),sub,blade(1,2,3)
!
fini
/prep7
/output,term
/go,
/com,*********************************************************************
/com,              MOVING THE NODES TO THE FIRST GUESS POSITION
/COM,*********************************************************************
/nopr
/output,junk_data_jr
k=1
/nopr
/output,junk_data_jr
*do,i,1,mxnode
*if,mask(i,1,1),ge,1,then
n,i,blade(i,3,1),blade(i,3,2),blade(i,3,3)
*else
*endif
k=k+1
*enddo
fini
/solu
/output,term
/go,
/com,*********************************************************************
/com,              RUNNING SECOND ANALYSIS WITH GUESSED POSITIONS
/COM,*********************************************************************
/nopr
/output,junk_data_jr
esel,s,type,,1
nelem
```

```
ANTYPE,0
time,1
deltim,.2,.01,.5
NLGEOM,on
LUMPM,0
EQSLV,sparse
autots,on
sstif,on
tref,70
TOFFST,0,
KBC,0
csys
esel,s,type,,1
nelem
solve
fini
/post1
*vget,blade(1,4,1),node,1,u,x
*vget,blade(1,4,2),node,1,u,y
*vget,blade(1,4,3),node,1,u,z
*vmask,mask(1,1,1)
*voper,blade(1,5,1),blade(1,3,1),add,blade(1,4,1)
*vmask,mask(1,1,2)
*voper,blade(1,5,2),blade(1,3,2),add,blade(1,4,2)
*vmask,mask(1,1,3)
*voper,blade(1,5,3),blade(1,3,3),add,blade(1,4,3)
*vmask,mask(1,1,1)
*voper,blade(1,5,1),blade(1,1,1),sub,blade(1,5,1)
*vmask,mask(1,1,2)
*voper,blade(1,5,2),blade(1,1,2),sub,blade(1,5,2)
*vmask,mask(1,1,3)
*voper,blade(1,5,3),blade(1,1,3),sub,blade(1,5,3)
*vmask,mask(1,1,1)
*voper,blade(1,6,1),blade(1,3,1),add,blade(1,5,1)
*vmask,mask(1,1,2)
*voper,blade(1,6,2),blade(1,3,2),add,blade(1,5,2)
*vmask,mask(1,1,3)
*voper,blade(1,6,3),blade(1,3,3),add,blade(1,5,3)
!
fini
/prep7
/output,term
/go,
/com,*********************************************************************
/com,              MOVING THE NODES TO CALCULATED SECOND POSITION
```

```
/com,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,mxnode
*if,mask(i,1,1),ge,1,then
n,i,blade(i,6,1),blade(i,6,2),blade(i,6,3)
*else
*endif
*enddo
fini
/solu
/output,term
/go,
/com,*************************************************************************
/com,              RUNNING THIRD ANALYSIS WITH CALCULATED POSITIONS
/com,*************************************************************************
/nopr
/output,junk_data_jr
esel,s,type,,1
nelem
ANTYPE,0
time,1
deltim,.2,.01,.5
NLGEOM,on
!NROPT,AUTO, ,off
LUMPM,0
EQSLV,sparse
autots,on
SSTIF,1,
TOFFST,0,
tref,70
KBC,0
csys
esel,s,type,,1
nelem
solve
fini
/post1
*vget,blade(1,7,1),node,1,u,x
*vget,blade(1,7,2),node,1,u,y
*vget,blade(1,7,3),node,1,u,z
*vmask,mask(1,1,1)
*voper,blade(1,8,1),blade(1,6,1),add,blade(1,7,1)
*vmask,mask(1,1,2)
*voper,blade(1,8,2),blade(1,6,2),add,blade(1,7,2)
```

```
*vmask,mask(1,1,3)
*voper,blade(1,8,3),blade(1,6,3),add,blade(1,7,3)
*vmask,mask(1,1,1)
*voper,blade(1,8,1),blade(1,1,1),sub,blade(1,8,1)
*vmask,mask(1,1,2)
*voper,blade(1,8,2),blade(1,1,2),sub,blade(1,8,2)
*vmask,mask(1,1,3)
*voper,blade(1,8,3),blade(1,1,3),sub,blade(1,8,3)
!
/output,term
/go
/com,*************************************************************************
/com,              CHECKING CONVERGENCE OF DEFLECTED TO ORIGINAL
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*vabs,1,1,1,1
*vscfun,maxx,max,blade(1,8,1)
*vabs,1,1,1,1
*vscfun,maxy,max,blade(1,8,2)
*vabs,1,1,1,1
*vscfun,maxz,max,blade(1,8,3)
/output,term
/go,
*msg,info,maxx,maxy,maxz
The maximum distance between deflected and original &
is X= %g   Y= %g    Z= %g   units = inches
/nopr
/output,junk_data_jr
*if,maxx,ge,maxy,then
target=maxx
*else
target=maxy
*endif
*if,target,le,maxz,then
target=maxz
*else
*endif
*vmask,mask(1,1,1)
*voper,blade(1,9,1),blade(1,6,1),add,blade(1,8,1)
*vmask,mask(1,1,2)
*voper,blade(1,9,2),blade(1,6,2),add,blade(1,8,2)
*vmask,mask(1,1,3)
*voper,blade(1,9,3),blade(1,6,3),add,blade(1,8,3)
vpt=9
```

```
*if,target,ge,bltol,then
/output,term
/go,
/com,*************************************************************************
/com,             CONVERGENCE WAS NOT MET ON INITIAL SETUP RUNS
/COM,                DO LOOP PROCESS STARTING TO GET CONG. SOL.
/com,*************************************************************************
/nopr
/output,junk_data_jr
! start of do loop to get convergence on deflection between cold and hot
*do,f,1,20
fini
/prep7
esel,s,type,,1
nelem
/output,term
/go,
/com,*************************************************************************
*msg,info,f,
CONVERGENCE ITERATION LOOP  %g out of 20
*msg,info,maxx,bltol
MAX. DISTANCE X = %g   BLTOL =  %g
*msg,info,maxy,bltol
MAX. DISTANCE Y = %g   BLTOL =  %g
*msg,info,maxz,bltol
MAX. DISTANCE Z = %g   BLTOL =  %g
/com,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,mxnode
*if,mask(i,1,1),ge,1,then
n,i,blade(i,vpt,1),blade(i,vpt,2),blade(i,vpt,3)
*else
*endif
*enddo
fini
/solu
esel,s,type,,1
nelem
ANTYPE,0
time,1
deltim,.2,.01,.5
NLGEOM,on
!NROPT,AUTO, ,off
LUMPM,0
```

```
EQSLV,sparse
autots,on
SSTIF,1,
TOFFST,0,
tref,70
KBC,0
csys
esel,s,type,,1
nelem
solve
fini
/post1
*vget,blade(1,vpt+1,1),node,1,u,x
*vget,blade(1,vpt+1,2),node,1,u,y
*vget,blade(1,vpt+1,3),node,1,u,z
*vmask,mask(1,1,1)
*voper,blade(1,vpt+2,1),blade(1,vpt,1),add,blade(1,vpt+1,1)
*vmask,mask(1,1,2)
*voper,blade(1,vpt+2,2),blade(1,vpt,2),add,blade(1,vpt+1,2)
*vmask,mask(1,1,3)
*voper,blade(1,vpt+2,3),blade(1,vpt,3),add,blade(1,vpt+1,3)
*vmask,mask(1,1,1)
*voper,blade(1,vpt+2,1),blade(1,1,1),sub,blade(1,vpt+2,1)
*vmask,mask(1,1,2)
*voper,blade(1,vpt+2,2),blade(1,1,2),sub,blade(1,vpt+2,2)
*vmask,mask(1,1,3)
*voper,blade(1,vpt+2,3),blade(1,1,3),sub,blade(1,vpt+2,3)
!
*vabs,1,1,1,1
*vscfun,maxx,max,blade(1,vpt+2,1)
*vabs,1,1,1,1
*vscfun,maxy,max,blade(1,vpt+2,2)
*vabs,1,1,1,1
*vscfun,maxz,max,blade(1,vpt+2,3)
*if,maxx,ge,maxy,then
target=maxx
*else
target=maxy
*endif
*if,target,le,maxz,then
target=maxz
*else
*endif
*if,target,le,bltol,exit
*else
```

```
vpt=vpt+3
*vmask,mask(1,1,1)
*voper,blade(1,vpt,1),blade(1,vpt-3,1),add,blade(1,vpt-1,1)
*vmask,mask(1,1,2)
*voper,blade(1,vpt,2),blade(1,vpt-3,2),add,blade(1,vpt-1,2)
*vmask,mask(1,1,3)
*voper,blade(1,vpt,3),blade(1,vpt-3,3),add,blade(1,vpt-1,3)
*endif
*enddo
/output,term
/go
/com,*************************************************************
*msg,info,vpt,
Converged geometry is in column %g of 60
*msg,info,maxx
The max tolerance X = %g
*msg,info,maxy,
The max tolerance Y = %g
*msg,info,maxz
The max tolerance Z = %g
/com,*************************************************************
!
/nopr
/output,junk_data_jr
*else
!
/output,term
/go
/com,*************************************************************
/com,Converged on initial set up solution runs
*msg,info,maxx,
The max tolerance X = %g
*msg,info,maxy,
The max tolerance Y = %g
*msg,info,maxz
The max tolerance Z = %g
/com,*************************************************************
/com,*************************************************************
/com, WRITING DEFLECT_AIRFOIL.SURF FILE, COLD LOOP ITER. COMPLETE.
/COM,*************************************************************L
/nopr
/output,junk_data_jr
*endif
/nopr
/output,junk_data_jr
```

```
/auto
eplo
/noerase
/user
lplo
/erase
!/sys,audioplay -v 60 ~jz9hn2/sounds/npss_deflect_airfoil.au &
cmsel,s,surfnode
*get,mnsfn,node,,num,min
*do,i,1,sfnct
afsdata(i,1,2)=mnsfn
afsdata(i,2,2)=nx(mnsfn)
afsdata(i,3,2)=ny(mnsfn)
afsdata(i,4,2)=nz(mnsfn)
mnsfn=ndnext(mnsfn)
*enddo
*voper,afsdata(1,2,3),afsdata(1,2,1),sub,afsdata(1,2,2)
*voper,afsdata(1,3,3),afsdata(1,3,1),sub,afsdata(1,3,2)
*voper,afsdata(1,4,3),afsdata(1,4,1),sub,afsdata(1,4,2)
*voper,afsdata(1,1,3),afsdata(1,1,1),add,0.0
*voper,afsdata(1,2,3),afsdata(1,2,3),mult,-1
*voper,afsdata(1,3,3),afsdata(1,3,3),mult,-1
*voper,afsdata(1,4,3),afsdata(1,4,3),mult,-1
*cfopen,deflect_airfoil,surf
*vwrite,sfnct
(F12.0,tl1,' ')
*vwrite,afsdata(1,1,3),afsdata(1,2,3),afsdata(1,3,3),afsdata(1,4,3)
(F7.0,tl1,' ',1X,F12.8,1X,F12.8,1X,F12.8)
*cfclose,defect_airfoil,surf!
*get,ft_time,active,,time,wall
t_time=(ft_time-st_time)*60
cfd_time=(ft_cfd-st_cfd)*60
*msg,ui,cfd_time,t_time
CFD TIME = %G   COLD MORPH = %G
alls
eplo
/go,
/output,term
/nopr
$*DEL,AFSIZE
$*DEL,BASIC    $*DEL,BLADE_IN,   $*DEL,BLTOL
$*DEL,BN          $*DEL,CFD_TIME    $*DEL,CFL
$*DEL,CFMAX       $*DEL,CNPTS       $*DEL,CTAM
$*DEL,DIAM        $*DEL,ELCT_       $*DEL,F1EQ
$*DEL,FCART       $*DEL,FCOAG1      $*DEL,FCOAG2
```

```
$*DEL,FFULMG      $*DEL,FINVVI      $*DEL,FITCHK
$*DEL,FITFMG      $*DEL,FKINF       $*DEL,FMIXLEN
$*DEL,FMULTI      $*DEL,FNCMAX      $*DEL,FREST
$*DEL,FTURBB      $*DEL,FT_TIME     $*DEL,FWALLF
$*DEL,G           $*DEL,GAMMA       $*DEL,HALF
$*DEL,HALF_PT     $*DEL,HBCT        $*DEL,HBMN
$*DEL,HUBCUT      $*DEL,HUBEN       $*DEL,HUBLEN
$*DEL,HUBRATIO    $*DEL,I           $*DEL,IN_CT
$*DEL,J           $*DEL,JUDGE       $*DEL,K
$*DEL,KN5         $*DEL,KN6         $*DEL,KP1
$*DEL,KP2         $*DEL,KPTG        $*DEL,LEVEL2
$*DEL,LEVEL3      $*DEL,LEVEL4      $*DEL,LE_CT
$*DEL,LE_MIN      $*DEL,LLEN        $*DEL,LPAT
$*DEL,LTEDGE      $*DEL,M           $*DEL,MAX
$*DEL,MAXX        $*DEL,MAXY        $*DEL,MAXZ
$*DEL,MINE        $*DEL,MINN        $*DEL,MIN_CT_N
$*DEL,MIN_N       $*DEL,MLN         $*DEL,MNSFE
$*DEL,MNSFN       $*DEL,MXNODE      $*DEL,MXOELEM
$*DEL,MXONODE     $*DEL,MXSFN       $*DEL,NAME
$*DEL,NDCT        $*DEL,NDCT_       $*DEL,ND_CT_S
$*DEL,NLPAT       $*DEL,NNLP        $*DEL,NPTS
$*DEL,NSEC        $*DEL,N_PT        $*DEL,N_SLI
$*DEL,P3DPRT      $*DEL,PI          $*DEL,PREF
$*DEL,PRNO        $*DEL,PRTNO       $*DEL,PS
$*DEL,P_CT        $*DEL,P_MIN       $*DEL,RAD_RPM
$*DEL,RGAS        $*DEL,RMACH       $*DEL,SFNCT
$*DEL,SMSIZE      $*DEL,SS          $*DEL,STARTNUM,
$*DEL,ST_TIME     $*DEL,S_CT        $*DEL,S_MIN
$*DEL,T           $*DEL,TARGET      $*DEL,TCF
$*DEL,TE_CT       $*DEL,TE_MIN      $*DEL,TG1
$*DEL,TG2         $*DEL,TG3         $*DEL,TG4
$*DEL,TLINE       $*DEL,TREF        $*DEL,TTPUZ
$*DEL,TUZ         $*DEL,T_TIME      $*DEL,VIS2
$*DEL,VIS4        $*DEL,VPT
$*DEL,_BUTTON     $*DEL,FRESID
/GO
/sys,rm junk_data_jr
/go
!
/nopr
/output,junk_data_jr
/sys,cp blade_point.dat orig_blade_point.dat
/sys,cp blade_data.dat orig_blade_data.dat
/sys,rm SYSMSG
*cfopen,temp,input,
```

```
*vwrite,
('blade_data.dat')
*vwrite
('airfoil.surf')
*vwrite,
('deflect_airfoil.surf')
*vwrite,
('cold_blade_data.dat')
*vwrite,
('cold_blade_point.dat')
*cfclose,temp,input
/output,term
/go
/sys,~kzq0fd/wip/NPSS_RELEASE/INTERP_DEFLECT/interp_deflect.x <temp.input
*msg,ui
THE NEW_BLADE_POINT.DAT FILE HAS BEEN CREATED, PROGRAM ENDING.
!
/output,junk_data_jr
/nopr
!
/sys,rm  blade.tmp
/sys,rm  fem.tmp
/sys,rm  new_blade.tmp
/sys,rm  q.tmp
/sys,rm  temp.input
/sys,rm  blade.converge
/sys,rm  blade.forces
/sys,rm  blade.p3dabs
/sys,rm  blade.p3drel
/sys,rm  blade.restart.new
/sys,rm  blade.boundata
/sys,rm  blade.input
/sys,rm  blade.mesh
/sys,rm  menust.tmp
/sys,rm  xansys.log
/sys,rm  xansys_bg.csh
/sys,rm  xansys.tmp
!
! check the cold_blade_point.dat file points overlay the cold mesh
!
fini
/prep7
*del,jj
*dim,jj,array,1,2
*vleng,1,1
```

```
*vread,jj(1,1),cold_blade_point,dat
(7X,F7.0)
*vleng,1,1
*vread,jj(1,2),cold_blade_point,dat
(14X,F7.0)
n_pt=jj(1,1)*jj(1,2)
Nsec=jj(1,1)
Npts=jj(1,2)
Tg1=11
Tg2=70
Tg3=91
Tg4=150
*del,bladept
*dim,bladept,array,n_pt,3
*vread,bladept(1,1),cold_blade_point,dat,,,,,,1
(2X,F12.5)
*vread,bladept(1,2),cold_blade_point,dat,,,,,,1
(16X,F12.5)
*vread,bladept(1,3),cold_blade_point,dat,,,,,,1
(30X,F12.5)
k=1
*do,i,1,Nsec
*do,g,1,Npts
k,,bladept(k,1),bladept(k,2),bladept(k,3)
k=k+1
*enddo
*enddo
/erase
alls
/auto
eplo
/noerase
/user
lplo
kplo
/output,term
/go
/sys,rm junk_data_jr
save
```

Appendix C

NPSS STAGE 2 ANSYS MACRO

This appendix contains a detailed listing of the *npss_Phase2.mac* ANSYS macro. This macro controls the execution of tasks under the NPSS Multidiscplinary Integration and Analysis Stage II (cold to warm deflection analysis).

```
!23456789.123456789.123456789.123456789.123456789.123456789.123456789.123456789.
!*****************************************************************************
!*                                                                           *
!*                       ROLLS-ROYCE AEROSPACE ANSYS MACRO                   *
!*                                                                           *
!*****************************************************************************
!* MACRO NAME: npss_phase2                                                   *
!* VERSION:    1A0                                                           *
!* REVISED:    Aug 31. 2001 J. Rasche  - created                             *
!*                                                                           *
!*****************************************************************************
!* FUNCTION:    Builds the cold shape solid model using the cold_blade_point.dat
!*              file created in phase 1.  Iterates between ADPAC and ANSYS to  *
!*              get the hot running position for the given off design point.   **
!* LIMITATIONS: The current limitation of this program is that it can only     *
!*              create the solid model inside of ANSYS.  The future call for   *
!*              this macro is to drive UG, PRO/E and CADDS5 in an automated    *
!*              fashion, building the 3D model in the CAD system and importing *
!*              the geometry with ANSYS connection products.                   *
!*                                                                           *
!*****************************************************************************
!* OTHER MACROS OR PROGRAMS USED BY THIS MACRO:                              *
!*                                                                           *
!* cfd_wrapper.x, interp_deflect.x     No other macros called                *
!*                                                                           *
```

```
!* INPUT AND OUTPUT FILES USED BY THIS MACRO:                                *
!*                                                                           *
!* cold_blade_point.dat, cold_blade_data.dat, Offdesign_adpac.input          *
!* Offdesign_blade_data.dat, adpac.ansys, adpac.input, mod_blade_data.dat    *
!* mod_blade_point.dat, new_blade_data.dat, new_blade_point.dat, mater.mp    *
!* airfoile.surf, deflect_airfoil.surf, junk_data_jr, temp.input, cold       *!*
!*                                                                           *
!*****************************************************************************
!* INPUT PARAMETERS: RPM(1), XAXIS                                           *
!*                                                                           *
!* PARAMETERS CREATED:                                                       *
!* ARRAYS:                                                                   *
!*                                                                           *
!* JJ, bladept, afsdata, afedata, nlp, elp, vmask, vmaske, vmaskn, adpac,    *
!* adpace, pt_sli, rpm, invalue, inxyz, p_xyz, s_xyz, le_xyz, te_xyz, p_result *
!* s_results, l_result, t_result, cfd_p_x, cfd_s_x, cfd_p_v, cfd_s_v, surf_v  *
!* int_xyz, int_v, surf_x, surf_v, int_xyz, apply, onode, oelem, nnode, blade *
!* mask                                                                      *
!*                                                                           *
!* SCALAR:                                                                   *
!*                                                                           *
!* AFSIZE, BASIC, BLADE_IN, BLTOL, BN, CFD_TIME, CFL, CFMAX, CNPTS, CTAM, DIAM *
!* ELCT_, F1EQ ,FCART, FCOAG1, FCOAG2,FFULMG,FINVVI,FITCHK,FITFMG,FKINF,     *
!* FMIXLEN,FMULTI,FNCMAX,FREST,FTURBB ,FT_TIME,FWALLF,G ,GAMMA,HALF,HALF_PT  *
!* HBCT,HBMN,HUBCUT,HUBEN,HUBLEN,HUBRATIO,IN_CT,J,JUDGE,K,KN5,KN6,KP1,KP2,KPTG *
!* LEVEL2,LEVEL3,LEVEL4,LE_CT,LE_MIN,LLEN,LPAT,LTEDGE,M,MAX,MAXX,MAXY,MAXZ,  *
!* MINE,MINN,MIN_CT_N,MIN_N,MLN,MNSFE,MNSFN,MXNODE,MXOELEM,MXONODE,MXSFN,NME, *
!* NDCT,NDCT_,ND_CT_S,NLPAT,NNLP,NPTS,NSEC,N_PT,N_SLI,P3DPRT,PI,PREF,PRNO,PRTNO*
!* PS,P_CT,P_MIN,RAD_RPM,RGAS,RMACH,SFNCT,SMSIZE,SS,STARTNUM,ST_TIE,S_CT,S_MIN *
!* T,TARGET,TCF,TE_CT,TE_MIN ,TG1,TG2,TG3,TG4 ,TLINE,TREF,TTPUZ ,TUZ,T_TIME,  *
!* VIS2,VIS4,VPT,XAXIS,FRESID,INT                                            *
!*                                                                           *
!*                                                                           *
!*****************************************************************************
!* COMPONENTS CREATED:                                                       *
!* abase, atop, base, int_n, le, le_a, press, press_a, suck, suck_a, surfnode *
!* takebase, taketop, te, te_a, top                                          *
!*                                                                           *
!*                                                                           *
!*****************************************************************************
!* VERIFICATION PROBLEMS:  Type 4, LCC  and S42                              *
!*****************************************************************************
fini
/clear
/nopr
```

```
/output,junk_data_jr
*get,sttime,active,,time,wall
fini
/clear,
!
/sys,~kzq0fd/wip/NPSS_RELEASE/SCRIPTS/cfd_wrapper.x cold_blade_data.dat > cold
!
*del,nlp    ! number of lines of point data, first item in file
*del,elp    ! number of lines of element data, nlp+2 location in file
*del,vmask
*del,vmaske
*del,vmaskn
*del,adpac
*del,adpace
*del,pt_sli
nlp=
!
blade=1
!           read nodal data
!
*dim,nlp,array,1,1            ! sets up initial array to get node count
*dim,pt_sli,array,1,1
*vread,nlp(1,1),adpac,ansys   ! Reads the first line of adpac.ansys file
(F10.0)
*vread,pt_sli(1,1),adpac,ansys  ! Reads number of points per slice
(10X,F10.0)
nnlp=nlp(1,1)+1               ! Adds one to it to include itself
*dim,vmaskn,array,nnlp,1      ! Creates node masking array
*dim,adpac,array,nnlp,6       ! Creates adpac array
*vfill,vmaskn(1,1),ramp,1,1   ! fills array with + 1's
vmaskn(1,1)=-1                ! changes the first row to -1,skips node count #
*vmask,vmaskn(1,1)
*vread,adpac(1,1),adpac,ansys
(F7.0)
*vmask,vmaskn(1,1)
*vread,adpac(1,2),adpac,ansys
(7X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,3),adpac,ansys
(19X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,4),adpac,ansys
(31X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,5),adpac,ansys
```

```
(43X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,6),adpac,ansys
(55X,F10.6)
!
*del,RPM(1,1)
*dim,RPM,,5
/input,offdesign_adpac,input
*del,invalue
*del,inxyz
*dim,invalue,array,nlp(1,1),2      ! array for in values on moper command
*dim,inxyz,array,nlp(1,1),3       ! array for xyz values for in values
*voper,invalue(1,1),adpac(2,5),add,
*voper,invalue(1,2),adpac(2,6),add,
*voper,inxyz(1,1),adpac(2,2),add,-xaxis
*voper,inxyz(1,2),adpac(2,3),add
*voper,inxyz(1,3),adpac(2,4),add
!
! add in array operations to move the data to the right array
!
!
/color,pbak,off                    ! turns the background of ansys to black
fini                               ! makes sure macro goes to begin level
/prep7                             ! Enters /prep7 of ANSYS
*del,jj                            ! deletes jj array if it has been defined
*dim,jj,array,1,2                  ! Defines jj array
*vleng,1,1
*vread,jj(1,1),cold_blade_point,dat  ! reads the number of sections per blade
(7X,F7.0)
*vleng,1,1
*vread,jj(1,2),cold_blade_point,dat  ! reads the number of points per section
(14X,F7.0)
n_pt=jj(1,1)*jj(1,2)               ! calculates the total number of points
Nsec=jj(1,1)                       ! sets Nsec equal to number of sectoins
Npts=jj(1,2)                       ! sets Npts equal to number of points/sec.
Tg1=11                             ! hardware tangency points, can read in
Tg2=70                             ! from file in future release
Tg3=91
Tg4=150
*del,bladept
*dim,bladept,array,n_pt,3       !read in x,y,z points from file,skip 1st row
*vread,bladept(1,1),cold_blade_point,dat,,,,,,1
(2X,F12.5)
*vread,bladept(1,2),cold_blade_point,dat,,,,,,1
(16X,F12.5)
```

```
*vread,bladept(1,3),cold_blade_point,dat,,,,,,1
(30X,F12.5)
m=1
b=0
k=1
*do,i,1,Nsec      ! makes kps at each section,offset by 1000
*do,g,1,Npts
k,m,bladept(k,1),bladept(k,2),bladept(k,3)
m=m+1
k=k+1
*enddo
m=1
b=b+1000
m=b+m
*enddo
/auto
kplo
!
!
/prep7
/go
/output,term
/com,*************************************************************************
/com,                    READING IN mater.mp MATERIAL FILE
/COM,*************************************************************************
/nopr
/output,junk_data_jr
/input,mater,mp   ! material properties file in ANSYS command format.
*SET,AFSIZE  , 0.4000000000000
*SET,huben   , 2.000000000000 ! Number of elements from hub to tangency pt.
*set,level2  , 5.00000000000  ! Number of elements in second level.
*set,level3  , 4.00000        ! Number of elements in 3rd level.
*set,level4  , 3.00000        ! Number of elements in 4th level.
*set,hublen , 8               ! Number of elements from edge to center.
*set,ltedge  , 2.0000         ! Number of elements through the LE and TE
*set,hubratio, 3              ! ratio of first element to last element in
*set,basic   , .3            ! Basic element size, global element size.
*SET,HUBCUT  ,  .25
*SET,TCF     , -459.69           ! Thermal correction factor
*SET,SMSIZE  ,  4.000000000000
*SET,STARTNUM,  60000.00000000
!
!
/uis,msgpop,3             !
/nerr,5,10000000          ! writes out 5 warning messages and 10000 error
```

```
                          ! messages to the window if running interactive
                          ! All warnings and errors are written to filename.err
/pnum,kpoi                ! Turns Keypoint number off for graphical display
/pnum,line,               ! Turns line numbers off for graphical display
/view,1,1,.3,.8           ! Sets view vector
/triad,lbot               ! Sets triad to lower left corner of window
/vup,1,z                  ! Sets Z axis up
/go
/output,term
/com,*********************************************************************
/com,                   CREATING SPLINES FOR VOLUME CREATION
/com,*********************************************************************
/nopr
/output,junk_data_jr
!
!
!
!
/auto
/title,Starting splining operation, %Npts% per section, %Nsec% number of sections
kplo
!
! The following section determines how many splines to make as it goes around
! the stream sections.
offset=0
p=1
cnpts=Npts-6
lpat=cnpts/5
lpat=lpat+1
nlpat=nint(lpat)
judge=lpat-nlpat
*if,judge,lt,0.0,then
   nlpat=nlpat-1
   *elseif,judge,gt,0.0,then
   nlpat=nlpat
  *elseif,judge,eq,0,then
  match=1
  *else
*endif
*do,p,1,Nsec,1
spn1=1+offset
spn2=2+offset
spn3=3+offset
spn4=4+offset
spn5=5+offset
```

```
spn6=6+offset
spn7=1+offset
i=1
*do,i,1,nlpat,1
spline,spn1,spn2,spn3,spn4,spn5,spn6
spn1=spn1+5
spn2=spn2+5
spn3=spn3+5
spn4=spn4+5
spn5=spn5+5
spn6=spn6+5
*enddo
*if,match,eq,1,then
   kptg=0
   *else
kptg=(Npts+offset)-(spn6-5)
 *endif
*if,kptg,eq,4,then
      spline,spn1,spn2,spn3,spn4,spn5,spn7
   *elseif,kptg,eq,3,then
      spline,spn1,spn2,spn3,spn4,spn7
   *elseif,kptg,eq,2,then
     spline,spn1,spn2,spn3,spn7
   *elseif,kptg,eq,1,then
     spline,spn1,spn2,spn7
   *elseif,kptg,eq,0,then
     spline,spn1,spn7
   *elseif,kptg,gt,6,then
*msg,warn
more than 6 keypoints left
*endif
offset=offset+1000
p=p+1
/auto
lplo
*enddo
/auto
/title,End of spline operation, moving on to cutting the hub tangency fillet
lplo
!
! End of the import stage.
!*********************************************************************************
!
! Splines up the sides so that the Hub tangency point can be cut.
!
```

```
/go
/output,term
/com,*************************************************************************
/com,       SPLINING UP THE SIDES FOR HUB CUT AT HUB FILLET TANG. PT.
/com,*************************************************************************
/nopr
/output,junk_data_jr
/title, Splining up the sides of the airfoil
kn1=1
alls
kplo
*get,mln,line,,num,max
mln=mln+1
*do,i,1,Npts,1
spline,kn1,kn1+1000,kn1+2000,kn1+3000,kn1+4000,kn1+5000
kn1=kn1+1
*enddo
tline=mln
*do,i,1,Npts,1
*get,llen,line,tline,leng
ldiv,tline,hubcut/llen,,2,0  ! cuts the bottom spline at the hub tangency
! fillet point
tline=tline+5
*enddo
lsel,s,line,,mln,400000,1
ldel,all
alls
lplo
!
!
! Splines the tangency section.
!
offset=0
spn1=Npts+1
spn2=Npts+2
spn3=Npts+3
spn4=Npts+4
spn5=Npts+5
spn6=Npts+6
spn7=spn1
*do,i,1,nlpat,1
spline,spn1,spn2,spn3,spn4,spn5,spn6
        spn1=spn1+5
        spn2=spn2+5
        spn3=spn3+5
```

```
        spn4=spn4+5
        spn5=spn5+5
        spn6=spn6+5
*enddo
!*if,match,eq,1,then
!   kptg=0
!   *else
!kptg=(Npts+offset)-(spn6-5)
! *endif
*if,kptg,eq,4,then
      spline,spn1,spn2,spn3,spn4,spn5,spn7
   *elseif,kptg,eq,3,then
      spline,spn1,spn2,spn3,spn4,spn7
   *elseif,kptg,eq,2,then
     spline,spn1,spn2,spn3,spn7
   *elseif,kptg,eq,1,then
     spline,spn1,spn2,spn7
   *elseif,kptg,eq,0,then
     spline,spn1,spn7
   *elseif,kptg,gt,6,then
*else
*endif
/auto
/title, Combining spline lines
lplo
/go
/output,term
/com,*************************************************************************
/com,              COMBINING SPLINE LINES FOR SOLID MODEL CREATION...
/com,*************************************************************************
/nopr
/output,junk_data_jr
! combining spline lines
/nopr
kn1=Tg1
kn2=Tg2
kn3=Tg3
kn4=Tg4
ksel,s,kp,,1,Tg1
ksel,a,kp,,Tg4,Npts,
lslk,s,1
lcomb,all
ksel,s,kp,,Tg1,Tg2
lslk,s,1
lcomb,all
```

```
ksel,s,kp,,Tg2,Tg3
lslk,s,1
lcomb,all
ksel,s,kp,,Tg3,Tg4
lslk,s,1
lcomb,all
ksel,s,kp,,1+Npts,Tg1+Npts
ksel,a,kp,,Tg4+Npts,Npts+Npts,
lslk,s,1
lcomb,all
ksel,s,kp,,Tg1+Npts,Tg2+Npts
lslk,s,1
lcomb,all
ksel,s,kp,,Tg2+Npts,Tg3+Npts
lslk,s,1
lcomb,all
ksel,s,kp,,Tg3+Npts,Tg4+Npts
lslk,s,1
lcomb,all
kn1=Tg1+1000
kn2=Tg2+1000
kn3=Tg3+1000
kn4=tg4+1000
kn5=1001
kn6=Npts+1000
*do,i,1,Nsec-1,1
ksel,s,kp,,kn5,kn1
ksel,a,kp,,kn4,kn6
lslk,s,1
lcomb,all
ksel,s,kp,,kn1,kn2
lslk,s,1
lcomb,all
ksel,s,kp,,kn2,kn3
lslk,s,1
lcomb,all
ksel,s,kp,,kn3,kn4
lslk,s,1
lcomb,all
kn1=kn1+1000
kn2=kn2+1000
kn3=kn3+1000
kn4=kn4+1000
kn5=kn5+1000
kn6=kn6+1000
```

```
*enddo
numcp,kpoi
/pnum,kpoi,1
alls
/auto
lplo
!  spline up the new 4 sides to get shape of airfoil
numcp,all
/auto
lplo
alls
bn=1
*do,i,1,4,1
kn1=bn
flst,2,Nsec+1,3
*do,j,1,Nsec+1,1
fitem,2,kn1
kn1=kn1+4
*enddo
spline,P51X
bn=bn+1
*enddo
/pnum,line
/pnum,kpoi
/auto
alls
/auto
kn1=1
kn2=2
kn3=3
kn4=4
/title, Creating the volumes used for meshing and meshing with element 95
lplo
/go
/output,term
/com,*************************************************************************
/com,                    CREATING VOLUMES FOR BRICK MESHING......
/com,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,Nsec,1
v,kn1,kn2,kn3,kn4,kn1+4,kn2+4,kn3+4,kn4+4
kn1=kn1+4
kn2=kn2+4
kn3=kn3+4
```

```
kn4=kn4+4
*enddo
ksel,s,kp,,1,
ksel,a,kp,,4
lslk,s,1
lesize,all,,,2
ksel,s,kp,,1,2
lslk,s,1
lesize,all,,,15
ksel,s,kp,,2,3
lslk,s,1
lesize,all,,,2
ksel,s,kp,,3,4
lslk,s,1
lesize,all,,,15
alls
esize,distkp(4,(Nsec+1)*4)/18
shpp,warn
et,1,95
/go,
/output,term
/com,*********************************************************************
/com,                        MESHING VOLUMES, PLEASE WAIT......
/COM,*********************************************************************
/nopr
/output,junk_data_jr
vmesh,all
/title,Meshing completed
/color,num,oran,1
eplo
!
!
!
!
!   completed volume creation and meshing
!
!
!
!
!
!
!
/output,term
/com,*********************************************************************
/com,                        MESHING VOLUMES, PLEASE WAIT......
```

```
/COM,*************************************************************************
/com,*************************************************************************
/com,                    CREATING COMPONENTS PRESS, SUCK, LE AND TE
/com,*************************************************************************
/nopr
/output,junk_data_jr
!
/com, selecting suction side areas for creation of nodal component
!
kp1=1
kp2=2
ksel,s,kp,,kp1,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,kp2
*enddo
lskp,1
arls,1
cm,suck_a,area
nsla,s,1
nplo
cm,suck,node
!
/com, created suck side area and nodal component
!
/com, starting on TE side
kp1=2
kp2=3
ksel,s,kp,,kp1,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,kp2
*enddo
lskp,1
arls,1
cm,te_a,area
nsla,s,1
nplo
cm,te,node
!
/com, finished TE side area and nodal component
!
/com, starting on pressure side
```

```
kp1=3
kp2=4
ksel,s,kp,,kp1,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,kp2
*enddo
lskp,1
arls,1
cm,press_a,area
nsla,s,1
nplo
cm,press,node
!
/com, finished pressure side area and nodal components
!
/com, starting on LE side
kp1=1
kp2=4
ksel,s,kp,,kp1
ksel,a,kp,,kp2
*do,i,1,Nsec
kp1=kp1+4
kp2=kp2+4
ksel,a,kp,,kp1,
ksel,a,kp,,kp2
*enddo
lskp,1
arls,1
cm,le_a,area
nsla,s,1
nplo
cm,le,node
/com, finished TE side area and nodal components
cmsel,s,press
cmsel,a,suck
cmsel,a,le
cmsel,a,te
cm,surfnode,node
alls
! deleting parameters
b=
ecount=
emin=
```

```
i=
kn1=
kn2=
kn3=
kn4=
ncount=
nmin=
nn1=
nn2=
nn3=
nn4=
np1x=
np1z=
np2x=
np2z=
offset=
p=
spn1=
spn2=
spn3=
spn4=
spn5=
spn6=
spn7=
x=
y=
/go
/output,term
/com,*************************************************************************
/com,                  SOLID MODEL AND MESH CREATION COMPLETED....
/com,*************************************************************************
/nopr
/output,junk_data_jr
alls
eplo
! add surface mesh for deflect_airfoil.surf file
/go
/output,term
/com,*************************************************************************
/com,                     WRITING OUT THE AIRFOIL.SURF FILE
/COM,*************************************************************************
/nopr
/output,junk_data_jr
fini
/prep7
```

```
alls
et,2,22
type,2
esurf
cmsel,s,surfnode
enode,1
cm,mask,elem
esel,s,type,,2
cmsel,u,mask
edel,all
cmsel,s,mask
*get,sfnct,node,,count
*get,ctam,elem,,count
*del,afsdata
*del,afedata
*dim,afsdata,array,sfnct,4,8
*dim,afedata,array,ctam,5
*get,mxsfn,node,,num,max
*get,mnsfn,node,,num,min
*get,mnsfe,elem,,num,min
*do,i,1,sfnct
afsdata(i,1,1)=mnsfn
afsdata(i,2,1)=nx(mnsfn)
afsdata(i,3,1)=ny(mnsfn)
afsdata(i,4,1)=nz(mnsfn)
mnsfn=ndnext(mnsfn)
*enddo
*do,t,1,ctam
afedata(t,1)=mnsfe
afedata(t,2)=nelem(mnsfe,1)
afedata(t,3)=nelem(mnsfe,2)
afedata(t,4)=nelem(mnsfe,3)
afedata(t,5)=nelem(mnsfe,4)
mnsfe=elnext(mnsfe)
*enddo
!
!
!
!  Writes out airfoil.surf file
!
!
!
!
*cfopen,airfoil,surf
*vwrite,sfnct
```

```
(F7.0,tl1,' ')
*vwrite,afsdata(1,1,1),afsdata(1,2,1),afsdata(1,3,1),afsdata(1,4,1)
(F7.0,tl1,' ',3F12.6)
*vwrite,ctam,
(F7.0,tl1,' ')
*vwrite,afedata(1,1),afedata(1,2),afedata(1,3),afedata(1,4),afedata(1,5),
(F7.0,tl1,' ',F12.0,tl1,' ',F12.0,tl1,' ',F12.0,tl1,' ',F12.0,tl1,' ')
*cfclose,airfoil,surf
esel,s,type,,2
edel,all
alls
etdel,2
/go
/output,term
/com,*************************************************************************
/com,              FINISHED WITH WRITING AIRFOIL.SURF FILE
/com,*************************************************************************
/nopr
/output,junk_data_jr
/go
/output,term
/com,*************************************************************************
/com, READING IN ADPAC.ANSYS AND ADPAC.INPUT FILES FROM ORIGINAL DESIGN POINT
/COM,*************************************************************************
/nopr
/output,junk_data_jr
fini
/prep7
*del,nlp    ! number of lines of point data, first item in file
*del,elp    ! number of lines of element data, nlp+2 location in file
*del,vmask
*del,vmaske
*del,vmaskn
*del,adpac
*del,adpace
*del,pt_sli
nlp=
elp=
!
!           read nodal data
!
*dim,nlp,array,1,1            ! sets up initial array to get node count
*dim,pt_sli,array,1,1
*vread,nlp(1,1),adpac,ansys   ! Reads the first line of adpac.ansys file
(F10.0)
```

```
*vread,pt_sli(1,1),adpac,ansys  ! Reads number of points per slice
(10X,F10.0)
nnlp=nlp(1,1)+1               ! Adds one to it to include itself
*dim,vmaskn,array,nnlp,1      ! Creates node masking array
*dim,adpac,array,nnlp,6       ! Creates adpac array
*vfill,vmaskn(1,1),ramp,1,1   ! fills array with + 1's
!vmaskn(1,1)=-1                ! changes the first row to -1,skips node count #
!*vmask,vmaskn(1,1)
!*vread,adpac(1,1),adpac,ansys
!(F7.0)
!*vmask,vmaskn(1,1)
!*vread,adpac(1,2),adpac,ansys
!(7X,F12.6)
!*vmask,vmaskn(1,1)
!*vread,adpac(1,3),adpac,ansys
!(19X,F12.6)
!*vmask,vmaskn(1,1)
!*vread,adpac(1,4),adpac,ansys
!(31X,F12.6)
!
!
*vmask,vmaskn(1,1)
*vread,adpac(1,5),adpac,ansys
(43X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,6),adpac,ansys
(55X,F10.6)
!
*del,invalue
*dim,invalue,array,nlp(1,1),2      ! array for in values on moper command
!*dim,inxyz,array,nlp(1,1),3       ! array for xyz values for in values
*voper,invalue(1,1),adpac(2,5),add,
*voper,invalue(1,2),adpac(2,6),add,
!*voper,inxyz(1,1),adpac(2,2),add,-xaxis
!*voper,inxyz(1,2),adpac(2,3),add
!*voper,inxyz(1,3),adpac(2,4),add
!
! add in array operations to move the data to the right array
!
cmsel,s,press
*get,p_ct,node,,count
*get,p_min,node,,num,min
cmsel,s,suck
*get,s_ct,node,,count
*get,s_min,node,,num,min
```

```
cmsel,s,le
*get,le_ct,node,,count
*get,le_min,node,,num,min
cmsel,s,te
*get,te_ct,node,,count
*get,te_min,node,,num,min
!
*del,p_xyz
*del,s_xyz
*del,le_xyz
*del,te_xyz
*del,p_result
*del,s_result
*del,l_result
*del,t_result
*del,cfd_p_x
*del,cfd_s_x
*del,cfd_p_v
*del,cfd_s_v
*del,surf_v
*del,surf_x
*del,int_xyz
*del,int_v
*del,surf_x
*del,surf_v
*del,int_xyz
*del,int_v
!
*dim,p_xyz,array,p_ct,3        ! pressure side xyz
*dim,p_result,array,p_ct,3     ! pressure side results
*dim,s_xyz,array,s_ct,3        ! suction side xyz
*dim,s_result,array,s_ct,3     ! suction side results
*dim,le_xyz,array,le_ct,3
*dim,l_result,array,le_ct,3
*dim,te_xyz,array,te_ct,3
*dim,t_result,array,te_ct,3
!
n_sli=nlp(1,1)/pt_sli(1,1)     ! calculates number of slices in cfd data
half_pt=pt_sli(1,1)/2          ! calculates the number of points per half
                               ! of a slice
half=half_pt*n_sli
!
*dim,cfd_p_x,array,half,3
*dim,cfd_s_x,array,half,3
*dim,cfd_p_v,array,half,2
```

```
*dim,cfd_s_v,array,half,2
!
! fill p,s,le and te xyz and result arrays
!
! get the xyz locations and node number of nodes on pressure side
cmsel,s,press
m=p_min
*do,i,1,p_ct
p_xyz(i,1)=nx(m)
p_xyz(i,2)=ny(m)
p_xyz(i,3)=nz(m)
p_result(i,3)=m        ! node number in column 3
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the suction side
cmsel,s,suck
m=s_min
*do,i,1,s_ct
s_xyz(i,1)=nx(m)
s_xyz(i,2)=ny(m)
s_xyz(i,3)=nz(m)
s_result(i,3)=m        ! node number in column 3
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the LE  side!
cmsel,s,le
m=le_min
*do,i,1,le_ct
le_xyz(i,1)=nx(m)
le_xyz(i,2)=ny(m)
le_xyz(i,3)=nz(m)
l_result(i,3)=m        ! node number in column 3m
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the TE side
cmsel,s,te
m=te_min
*do,i,1,te_ct
te_xyz(i,1)=nx(m)
te_xyz(i,2)=ny(m)
te_xyz(i,3)=nz(m)
t_result(i,3)=m      ! node number in column 3
```

```
m=ndnext(m)
*enddo
!
alls
!
k=1   $ps=1  $ss=1
*do,i,1,n_sli
  *do,j,1,half_pt
     cfd_s_x(ss,1)=inxyz(k,1)
     cfd_s_x(ss,2)=inxyz(k,2)
     cfd_s_x(ss,3)=inxyz(k,3)
     cfd_s_v(ss,1)=invalue(k,1)   ! pressure
     cfd_s_v(ss,2)=invalue(k,2)   ! temperature
     k=k+1
     ss=ss+1
  *enddo
  *do,g,1,half_pt
    cfd_p_x(ps,1)=inxyz(k,1)
    cfd_p_x(ps,2)=inxyz(k,2)
    cfd_p_x(ps,3)=inxyz(k,3)
    cfd_p_v(ps,1)=invalue(k,1)
    cfd_p_v(ps,2)=invalue(k,2)
    k=k+1
    ps=ps+1
  *enddo
*enddo
!
! mapping of temperatures and pressures
!

/go
/output,term
/com,*************************************************************************
/com,     MAPPING TEMPERATURE AND PRESSURE FROM CFD TO STRUCTURAL MODEL
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*moper,p_result(1,2),p_xyz(1,1),map,cfd_p_v(1,2),cfd_p_x(1,1),0   ! Temps
!
*moper,p_result(1,1),p_xyz(1,1),map,cfd_p_v(1,1),cfd_p_x(1,1),0   ! Press
!
*moper,s_result(1,2),s_xyz(1,1),map,cfd_s_v(1,2),cfd_s_x(1,1),0   ! Temps
!
*moper,s_result(1,1),s_xyz(1,1),map,cfd_s_v(1,1),cfd_s_x(1,1),0   ! Press
!
```

```
*moper,l_result(1,2),le_xyz(1,1),map,invalue(1,2),inxyz(1,1),0    ! Temps
!
*moper,l_result(1,1),le_xyz(1,1),map,invalue(1,1),inxyz(1,1),0    ! Press
!
*moper,t_result(1,2),te_xyz(1,1),map,invalue(1,2),inxyz(1,1),0    ! Temps
!
*moper,t_result(1,1),te_xyz(1,1),map,invalue(1,1),inxyz(1,1),0    ! Press
!
alls
fini
/solu
!
!
! put on the temperature
!
/go
/output,term
/com,*************************************************************************
/com,                         APPLYING TEMPERATURES
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,p_ct
bf,p_result(i,3),temp,p_result(i,2)
*enddo
*do,i,1,s_ct
bf,s_result(i,3),temp,s_result(i,2)
*enddo
*do,i,1,le_ct
bf,l_result(i,3),temp,l_result(i,2)
*enddo
*do,i,1,te_ct
bf,t_result(i,3),temp,t_result(i,2)
*enddo
!
!
/go
/output,term
/com,*************************************************************************
/com,                          CALCULATING PRESSURES
/COM,*************************************************************************
/NOPR
/output,junk_data_jr
cmsel,s,surfnode
*get,min_n,node,,num,min
```

```
*get,nd_ct_s,node,,count
*dim,surf_x,array,nd_ct_s,3
*dim,surf_v,array,nd_ct_s,1
m=min_n
*do,i,1,nd_ct_s
surf_x(i,1)=nx(m)
surf_x(i,2)=ny(m)
surf_x(i,3)=nz(m)
*get,surf_v(i),node,m,ntem,bfe    ! get nodal temperatures from surface
m=ndnext(m)
*enddo
!
alls
cmsel,u,surfnode
*get,in_ct,node,,count
*get,min_ct_n,node,,num,min
cm,int_n,node
*dim,int_xyz,array,in_ct,3
*dim,int_v,array,in_ct,2
!
m=min_ct_n
*do,i,1,in_ct,
int_xyz(i,1)=nx(m)
int_xyz(i,2)=ny(m)
int_xyz(i,3)=nz(m)
int_v(i,2)=m        ! node number in column 2
m=ndnext(m)
*enddo
!
*moper,int_v(1,1),int_xyz(1,1),map,surf_v(1,1),surf_x(1,1),0    ! Temps
!
/go
/output,term
/com,*************************************************************************
/com,                       APPLYING INTERNAL TEMPERATURES
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,in_ct
bf,int_v(i,2),temp,int_v(i,1)
*enddo
alls
/auto
/pbf,temp,1
eplo
```

```
/show,term
/nopr
/output,junk_data_jr
fini
/prep7
ksel,s,kp,,1,4
lslk,s,1
asll,s,1
nsla,s,1
cm,base,node   !create a component on nodes at the base / hub.
cm,abase,area  !create a component of areas at the base / hub.
nsla,
cm,takebase,node
ksel,s,kp,,(Nsec*4)+1,(Nsec*4)+4,1
lslk,s,1
asll,s,1
nsla,s,1
cm,top,node   ! create a component of nodes at the top / tip.
cm,atop,area  ! create a component of areas at the top / tip.
nsla,
cm,taketop,node
cmsel,s,base
cmsel,a,top
/auto
nplot
alls
eplo
! put on boundary conditions for the calculation of the thermal pinch.
cmsel,s,base
csys,0
!d,all,uz
ksel,s,kp,,1,3,2,
nslk,s
d,all,uy
cmsel,s,base
tuz=0
*get,hbct,node,,count
*get,hbmn,node,,num,min
*do,i,1,hbct
ttpuz=NZ(hbmn)
tuz=tuz+ttpuz
hbmn=ndnext(hbmn)
*enddo
tuz=tuz/hbct
target=node(0,0,tuz)
```

```
d,target,ux,0
!d,target,uz,0
cmsel,s,base
d,all,uz,0
cmsel,s,base
/pbc,u,1
nplo
/pbc,u,1
nplo
/view,1,1,.3,1
fini
/solu
time,1
NLGEOM,0
NROPT,AUTO, ,OFF
LUMPM,0
EQSLV,sparse,1E-6
SSTIF
PSTRES
tref,70
esel,s,type,,1
nelem
/go
/output,term
/com,*************************************************************************
/com,                  RUNNING A STRUCTURAL ANALYSIS TO CALCULATE
/com,                         CUT BOUNDARY DISPLACEMENTS
/COM,*************************************************************************
/nopr
/output,junk_data_jr
solve
cmsel,s,base
*get,ncount,node,,count
*get,nmin,node,0,num,min
/title,Putting on calculated displacements from thermal pinch analysis.
nplo
! putting on the calculated displacements
*do,i,1,ncount,1
d,nmin,ux,ux(nmin)
d,nmin,uy,uy(nmin)
d,nmin,uz,uz(nmin)
nmin=ndnext(nmin)
*enddo
alls
/solu
```

```
solve
fini
/prep7
/pbc,all
/pbf,temp,1  ! turn on temperatures for plot
eplo
/title,loading model with aero loads, please wait......
nplo
csys,0
!
! move the data from p_results,s_result,l_result and t_result into apply array
!
cmsel,s,surfnode
*get,max,node,,num,max
*del,apply
*dim,apply,array,max,1
*do,i,1,p_ct
apply(%p_result(i,3)%,1)=p_result(i,1)
*enddo
*do,i,1,s_ct
apply(%s_result(i,3)%,1)=s_result(i,1)
*enddo
*do,i,1,le_ct
apply(%l_result(i,3)%,1)=l_result(i,1)
*enddo
*do,i,1,te_ct
apply(%t_result(i,3)%,1)=t_result(i,1)
*enddo
!
eall
cmsel,s,surfnode
sffun,press,apply(1,1)
sf,all,press
alls
eplo
/solu
/go
/output,term
/com,*************************************************************************
/com,           RUNNING ANALYSIS WITH PRESSURE AND TEMPERATURES APPLIED
/COM,*************************************************************************
/NOPR
/output,junk_data_jr
ANTYPE,0
time,1
```

```
deltim,1,1,1
NLGEOM,on
NROPT,AUTO, ,off
LUMPM,0
EQSLV,sparse
autots,on
SSTIF,1,
TOFFST,0,
KBC,0
csys
*afun,rad
pi=acos(-1)
rad_rpm=(rpm(1,1)*2*pi)/60
omega,rad_rpm,
esel,s,type,,1
nelem
solve
FINISH
blade_in=1
*afun,deg
/post1
/efacet,4
esel,s,type,,1
nelem
/view,1,-1,-1.5,1
plns,s,eqv
b=
ecount=
emin=
i=
kn1=
kn2=
kn3=
kn4=
ncount=
nmin=
nn1=
nn2=
nn3=
nn4=
np1x=
np1z=
np2x=
np2z=
offset=
```

```
p=
pl1=
pl2=
pl3=
pl4=
pmaxnode=
rotline=
slopehub=
spn1=
spn2=
spn3=
spn4=
spn5=
spn6=
spn7=
x=
y=
!/sys,audioplay -v 60 ~jz9hn2/sounds/npss_conv_start.au &
/com,*************************************************************
/com, WRITING DEFLECT_AIRFOIL.SURF FILE, COLD LOOP ITER. COMPLETE.
/COM,*************************************************************
/nopr
/output,junk_data_jr
/auto
fini
/post1
eplo
/noerase
/user
lplo
/erase
!/sys,audioplay -v 60 ~jz9hn2/sounds/npss_deflect_airfoil.au &
cmsel,s,surfnode
*get,sfnct,node,,count
*get,mnsfn,node,,num,min
*do,i,1,sfnct
afsdata(i,1,2)=mnsfn
afsdata(i,2,2)=ux(mnsfn)
afsdata(i,3,2)=uy(mnsfn)
afsdata(i,4,2)=uz(mnsfn)
mnsfn=ndnext(mnsfn)
*enddo
*cfopen,deflect_airfoil,surf
*vwrite,sfnct
(F12.0,tl1,' ')
```

```
*vwrite,afsdata(1,1,2),afsdata(1,2,2),afsdata(1,3,2),afsdata(1,4,2)
(F7.0,tl1,' ',1X,F12.8,1X,F12.8,1X,F12.8)
*cfclose,defect_airfoil,surf!
alls
eplo
/go,
/output,term
/nopr
$*DEL,AFSIZE
$*DEL,BASIC    $*DEL,BLADE_IN,   $*DEL,BLTOL
$*DEL,BN          $*DEL,CFD_TIME    $*DEL,CFL
$*DEL,CFMAX       $*DEL,CNPTS       $*DEL,CTAM
$*DEL,DIAM        $*DEL,ELCT_       $*DEL,F1EQ
$*DEL,FCART       $*DEL,FCOAG1      $*DEL,FCOAG2
$*DEL,FFULMG      $*DEL,FINVVI      $*DEL,FITCHK
$*DEL,FITFMG      $*DEL,FKINF       $*DEL,FMIXLEN
$*DEL,FMULTI      $*DEL,FNCMAX      $*DEL,FREST
$*DEL,FTURBB      $*DEL,FT_TIME     $*DEL,FWALLF
$*DEL,G           $*DEL,GAMMA       $*DEL,HALF
$*DEL,HALF_PT     $*DEL,HBCT        $*DEL,HBMN
$*DEL,HUBCUT      $*DEL,HUBEN       $*DEL,HUBLEN
$*DEL,HUBRATIO    $*DEL,I           $*DEL,IN_CT
$*DEL,J           $*DEL,JUDGE       $*DEL,K
$*DEL,KN5         $*DEL,KN6         $*DEL,KP1
$*DEL,KP2         $*DEL,KPTG        $*DEL,LEVEL2
$*DEL,LEVEL3      $*DEL,LEVEL4      $*DEL,LE_CT
$*DEL,LE_MIN      $*DEL,LLEN        $*DEL,LPAT
$*DEL,LTEDGE      $*DEL,M           $*DEL,MAX
$*DEL,MAXX        $*DEL,MAXY        $*DEL,MAXZ
$*DEL,MINE        $*DEL,MINN        $*DEL,MIN_CT_N
$*DEL,MIN_N       $*DEL,MLN         $*DEL,MNSFE
$*DEL,MNSFN       $*DEL,MXNODE      $*DEL,MXOELEM
$*DEL,MXONODE     $*DEL,MXSFN       $*DEL,NAME
$*DEL,NDCT        $*DEL,NDCT_       $*DEL,ND_CT_S
$*DEL,NLPAT       $*DEL,NNLP        $*DEL,NPTS
$*DEL,NSEC        $*DEL,N_PT        $*DEL,N_SLI
$*DEL,P3DPRT      $*DEL,PI          $*DEL,PREF
$*DEL,PRNO        $*DEL,PRTNO       $*DEL,PS
$*DEL,P_CT        $*DEL,P_MIN      ! $*DEL,RAD_RPM
$*DEL,RGAS        $*DEL,RMACH       $*DEL,SFNCT
$*DEL,SMSIZE      $*DEL,SS          $*DEL,STARTNUM,
$*DEL,ST_TIME     $*DEL,S_CT        $*DEL,S_MIN
$*DEL,T           $*DEL,TARGET      $*DEL,TCF
$*DEL,TE_CT       $*DEL,TE_MIN      $*DEL,TG1
$*DEL,TG2         $*DEL,TG3         $*DEL,TG4
```

```
$*DEL,TLINE       $*DEL,TREF        $*DEL,TTPUZ
$*DEL,TUZ         $*DEL,T_TIME      $*DEL,VIS2
$*DEL,VIS4        $*DEL,VPT
$*DEL,_BUTTON     $*DEL,FRESID
/GO
!
/nopr
/output,junk_data_jr
/sys,rm SYSMSG
*cfopen,temp,input,
*vwrite,
('cold_blade_data.dat')
*vwrite
('airfoil.surf')
*vwrite,
('deflect_airfoil.surf')
*vwrite,
('mod_blade_data.dat')
*vwrite,
('mod_blade_point.dat')
*cfclose,temp,input
/output,term
/go
/sys,~kzq0fd/wip/NPSS_RELEASE/INTERP_DEFLECT/interp_deflect.x <temp.input
*msg,info
THE MOD_BLADE_POINT.DAT FILE HAS BEEN CREATED
!
/sys,~kzq0fd/wip/NPSS_RELEASE/SCRIPTS/cfd_wrapper.x mod_blade_data.dat
!
fini
/solu
save
airtol=.000001
*voper,afsdata(1,1,3),afsdata(1,1,1),mult,1
*voper,afsdata(1,2,3),afsdata(1,2,2),mult,1
*voper,afsdata(1,3,3),afsdata(1,3,2),mult,1
*voper,afsdata(1,4,3),afsdata(1,4,2),mult,1
vv=1
!
ITN=20
!
!
!
/output,term
/go
```

```
/COM,
/com,*************************************************************************
*msg,info,ITN
   STARTING *DO LOOP TO CONVERGE ON HOT RUNNING AIRFOIL SHAPE, LOOP =%G
*msg,info,airtol
                  CONVERGANCE IS SET AT %G
/COM,*************************************************************************
/COM,
/nopr
/output,junk_data_jr
!
!
!
!
*DO,r,1,itn
fini
/solu
alls
sfdel,all,all
bfdel,all,all
/go
/output,term
/com,*************************************************************************
*msg,info,r,itn
PROCESSING LOOP %G OUT OF %G TOTAL
/COM,*************************************************************************
/COM,
/com,*************************************************************************
/com,  READING IN ADPAC.ANSYS AND ADPAC.INPUT FILES FROM MODIFIED AIRFOIL
/COM,*************************************************************************
/nopr
/output,junk_data_jr
fini
/prep7
*del,nlp    ! number of lines of point data, first item in file
*del,elp    ! number of lines of element data, nlp+2 location in file
*del,vmask
*del,vmaske
*del,vmaskn
*del,adpac
*del,adpace
*del,pt_sli
nlp=
elp=
!
```

```
!           read nodal data
!
*dim,nlp,array,1,1            ! sets up initial array to get node count
*dim,pt_sli,array,1,1
*vread,nlp(1,1),adpac,ansys   ! Reads the first line of adpac.ansys file
(F10.0)
*vread,pt_sli(1,1),adpac,ansys  ! Reads number of points per slice
(10X,F10.0)
nnlp=nlp(1,1)+1               ! Adds one to it to include itself
*dim,vmaskn,array,nnlp,1      ! Creates node masking array
*dim,adpac,array,nnlp,6       ! Creates adpac array
*vfill,vmaskn(1,1),ramp,1,1   ! fills array with + 1's
vmaskn(1,1)=-1                ! changes the first row to -1,skips node count #
!*vmask,vmaskn(1,1)
!*vread,adpac(1,1),adpac,ansys
!(F7.0)
!*vmask,vmaskn(1,1)
!*vread,adpac(1,2),adpac,ansys
!(7X,F12.6)
!*vmask,vmaskn(1,1)
!*vread,adpac(1,3),adpac,ansys
!(19X,F12.6)
!*vmask,vmaskn(1,1)
!*vread,adpac(1,4),adpac,ansys
!(31X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,5),adpac,ansys
(43X,F12.6)
*vmask,vmaskn(1,1)
*vread,adpac(1,6),adpac,ansys
(55X,F10.6)
nomove=0
*del,invalue
*dim,invalue,array,nlp(1,1),2      ! array for in values on moper command
!*dim,inxyz,array,nlp(1,1),3       ! array for xyz values for in values
*voper,invalue(1,1),adpac(2,5),add,
*voper,invalue(1,2),adpac(2,6),add,
!*voper,inxyz(1,1),adpac(2,2),add,-xaxis
!*voper,inxyz(1,2),adpac(2,3),add
!*voper,inxyz(1,3),adpac(2,4),add
!
! add in array operations to move the data to the right array
!
cmsel,s,press
*get,p_ct,node,,count
```

```
*get,p_min,node,,num,min
cmsel,s,suck
*get,s_ct,node,,count
*get,s_min,node,,num,min
cmsel,s,le
*get,le_ct,node,,count
*get,le_min,node,,num,min
cmsel,s,te
*get,te_ct,node,,count
*get,te_min,node,,num,min
!
*del,p_xyz
*del,s_xyz
*del,le_xyz
*del,te_xyz
*del,p_result
*del,s_result
*del,l_result
*del,t_result
*del,cfd_p_x
*del,cfd_s_x
*del,cfd_p_v
*del,cfd_s_v
*del,surf_v
*del,surf_x
*del,int_xyz
*del,int_v
*del,surf_x
*del,surf_v
*del,int_xyz
*del,int_v
!
*dim,p_xyz,array,p_ct,3        ! pressure side xyz
*dim,p_result,array,p_ct,3     ! pressure side results
*dim,s_xyz,array,s_ct,3        ! suction side xyz
*dim,s_result,array,s_ct,3     ! suction side results
*dim,le_xyz,array,le_ct,3
*dim,l_result,array,le_ct,3
*dim,te_xyz,array,te_ct,3
*dim,t_result,array,te_ct,3
!
n_sli=nlp(1,1)/pt_sli(1,1)     ! calculates number of slices in cfd data
half_pt=pt_sli(1,1)/2          ! calculates the number of points per half
                               ! of a slice
half=half_pt*n_sli
```

```
!
*dim,cfd_p_x,array,half,3
*dim,cfd_s_x,array,half,3
*dim,cfd_p_v,array,half,2
*dim,cfd_s_v,array,half,2
!
! fill p,s,le and te xyz and result arrays
!
! get the xyz locations and node number of nodes on pressure side
cmsel,s,press
m=p_min
*do,w,1,p_ct
p_xyz(w,1)=nx(m)
p_xyz(w,2)=ny(m)
p_xyz(w,3)=nz(m)
p_result(w,3)=m        ! node number in column 3
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the suction side
cmsel,s,suck
m=s_min
*do,i,1,s_ct
s_xyz(i,1)=nx(m)
s_xyz(i,2)=ny(m)
s_xyz(i,3)=nz(m)
s_result(i,3)=m        ! node number in column 3
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the LE  side!
cmsel,s,le
m=le_min
*do,i,1,le_ct
le_xyz(i,1)=nx(m)
le_xyz(i,2)=ny(m)
le_xyz(i,3)=nz(m)
l_result(i,3)=m        ! node number in column 3m
m=ndnext(m)
*enddo
!
! get the xyz locations and node number of nodes on the TE side
cmsel,s,te
m=te_min
*do,i,1,te_ct
```

```
te_xyz(i,1)=nx(m)
te_xyz(i,2)=ny(m)
te_xyz(i,3)=nz(m)
t_result(i,3)=m      ! node number in column 3
m=ndnext(m)
*enddo
!
alls
!
k=1   $ps=1  $ss=1
*do,i,1,n_sli
  *do,j,1,half_pt
     cfd_s_x(ss,1)=inxyz(k,1)
     cfd_s_x(ss,2)=inxyz(k,2)
     cfd_s_x(ss,3)=inxyz(k,3)
     cfd_s_v(ss,1)=invalue(k,1)   ! pressure
     cfd_s_v(ss,2)=invalue(k,2)   ! temperature
     k=k+1
     ss=ss+1
  *enddo
  *do,g,1,half_pt
    cfd_p_x(ps,1)=inxyz(k,1)
    cfd_p_x(ps,2)=inxyz(k,2)
    cfd_p_x(ps,3)=inxyz(k,3)
    cfd_p_v(ps,1)=invalue(k,1)
    cfd_p_v(ps,2)=invalue(k,2)
    k=k+1
    ps=ps+1
  *enddo
*enddo
!
! mapping of temperatures and pressures
!

/go
/output,term
/com,*************************************************************************
/com,     MAPPING TEMPERATURE AND PRESSURE FROM CFD TO STRUCTURAL MODEL
/COM,*************************************************************************
/noprp
/output,junk_data_jr
*moper,p_result(1,2),p_xyz(1,1),map,cfd_p_v(1,2),cfd_p_x(1,1),0   ! Temps
!
*moper,p_result(1,1),p_xyz(1,1),map,cfd_p_v(1,1),cfd_p_x(1,1),0   ! Press
!
```

```
*moper,s_result(1,2),s_xyz(1,1),map,cfd_s_v(1,2),cfd_s_x(1,1),0   ! Temps
!
*moper,s_result(1,1),s_xyz(1,1),map,cfd_s_v(1,1),cfd_s_x(1,1),0   ! Press
!
*moper,l_result(1,2),le_xyz(1,1),map,invalue(1,2),inxyz(1,1),0    ! Temps
!
*moper,l_result(1,1),le_xyz(1,1),map,invalue(1,1),inxyz(1,1),0    ! Press
!
*moper,t_result(1,2),te_xyz(1,1),map,invalue(1,2),inxyz(1,1),0    ! Temps
!
*moper,t_result(1,1),te_xyz(1,1),map,invalue(1,1),inxyz(1,1),0    ! Press
!
alls
fini
/solu
!
!
! put on the temperature
!
/go
/output,term
/com,*************************************************************************
/com,                        APPLYING TEMPERATURES
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,p_ct
bf,p_result(i,3),temp,p_result(i,2)
*enddo
*do,i,1,s_ct
bf,s_result(i,3),temp,s_result(i,2)
*enddo
*do,i,1,le_ct
bf,l_result(i,3),temp,l_result(i,2)
*enddo
*do,i,1,te_ct
bf,t_result(i,3),temp,t_result(i,2)
*enddo
!
!
/go
/output,term
/com,*************************************************************************
/com,                         CALCULATING PRESSURES
/COM,*************************************************************************
```

```
/NOPR
/output,junk_data_jr
cmsel,s,surfnode
*get,min_n,node,,num,min
*get,nd_ct_s,node,,count
*dim,surf_x,array,nd_ct_s,3
*dim,surf_v,array,nd_ct_s,1
m=min_n
*do,i,1,nd_ct_s
surf_x(i,1)=nx(m)
surf_x(i,2)=ny(m)
surf_x(i,3)=nz(m)
*get,surf_v(i),node,m,ntem,bfe    ! get nodal temperatures from surface
m=ndnext(m)
*enddo
!
alls
cmsel,u,surfnode
*get,in_ct,node,,count
*get,min_ct_n,node,,num,min
cm,int_n,node
*dim,int_xyz,array,in_ct,3
*dim,int_v,array,in_ct,2
!
m=min_ct_n
*do,i,1,in_ct,
int_xyz(i,1)=nx(m)
int_xyz(i,2)=ny(m)
int_xyz(i,3)=nz(m)
int_v(i,2)=m        ! node number in column 2
m=ndnext(m)
*enddo
!
*moper,int_v(1,1),int_xyz(1,1),map,surf_v(1,1),surf_x(1,1),0    ! Temps
!
/go
/output,term
/com,*************************************************************************
/com,                       APPLYING INTERNAL TEMPERATURES
/COM,*************************************************************************
/nopr
/output,junk_data_jr
*do,i,1,in_ct
bf,int_v(i,2),temp,int_v(i,1)
*enddo
```

```
alls
/auto
/pbf,temp,1
eplo
/show,term
/nopr
/output,junk_data_jr
fini
/prep7
/pbc,all
/pbf,temp,1  ! turn on temperatures for plot
eplo
/title,loading model with aero loads, please wait......
nplo
csys,0
!
! move the data from p_results,s_result,l_result and t_result into apply array
!
cmsel,s,surfnode
*get,mmax,node,,num,max
*del,apply
*dim,apply,array,mmax,1
*do,i,1,p_ct
apply(%p_result(i,3)%,1)=p_result(i,1)
*enddo
*do,i,1,s_ct
apply(%s_result(i,3)%,1)=s_result(i,1)
*enddo
*do,i,1,le_ct
apply(%l_result(i,3)%,1)=l_result(i,1)
*enddo
*do,i,1,te_ct
apply(%t_result(i,3)%,1)=t_result(i,1)
*enddo
!
eall
cmsel,s,surfnode
sffun,press,apply(1,1)
sf,all,press
alls
/solu
/go
/output,term
/com,*************************************************************************
/com,           RUNNING ANALYSIS WITH PRESSURE AND TEMPERATURES APPLIED
```

```
/COM,*************************************************************************
/NOPR
/output,junk_data_jr
ANTYPE,0
time,1
NLGEOM,on
NROPT,AUTO, ,off
LUMPM,0
EQSLV,sparse
autots,on
SSTIF,1,
solcontrol,on
TOFFST,0,
KBC,0
csys
*afun,rad
omega,rad_rpm,
esel,s,type,,1
nelem
solve
FINISH
/post1
set,last
alls
/view,1,-.2,-1,.2
/psf,press,2
eplo
plns,u,sum
cmsel,s,surfnode
*get,sfnct,node,,count
*get,mnsfn,node,,num,min
*do,i,1,sfnct
afsdata(i,1,4)=mnsfn
afsdata(i,2,4)=ux(mnsfn)
afsdata(i,3,4)=uy(mnsfn)
afsdata(i,4,4)=uz(mnsfn)
mnsfn=ndnext(mnsfn)
*enddo
!*cfopen,defl,%r%
!*vwrite,afsdata(1,1,4),afsdata(1,2,4),afsdata(1,3,4),afsdata(1,4,4),
!(4F12.6)
!*cfopen,defl,%r%
*vwrite,afsdata(1,1,4),afsdata(1,2,4),afsdata(1,3,4),afsdata(1,4,4),
(4F12.6)
*cfclose,deflection
```

```
*voper,afsdata(1,2,5),afsdata(1,2,3),sub,afsdata(1,2,4)
*voper,afsdata(1,3,5),afsdata(1,3,3),sub,afsdata(1,3,4)
*voper,afsdata(1,4,5),afsdata(1,4,3),sub,afsdata(1,4,4)
!*cfopen,array5
!*vwrite,afsdata(1,1,5),afsdata(1,2,5),afsdata(1,3,5),afsdata(1,4,5),
!(4F12.6)
!*cfclose,array5
*voper,afsdata(1,2,3),afsdata(1,2,3),mult,0
*voper,afsdata(1,3,3),afsdata(1,3,3),mult,0
*voper,afsdata(1,4,3),afsdata(1,4,3),mult,0
!*cfopen,array3a
!*vwrite,afsdata(1,1,3),afsdata(1,2,3),afsdata(1,3,3),afsdata(1,4,3),
!(4F12.6)
!*cfclose,array3a
*voper,afsdata(1,2,3),afsdata(1,2,3),add,afsdata(1,2,4)
*voper,afsdata(1,3,3),afsdata(1,3,3),add,afsdata(1,3,4)
*voper,afsdata(1,4,3),afsdata(1,4,3),add,afsdata(1,4,4)
!*cfopen,array3b
!*vwrite,afsdata(1,1,5),afsdata(1,2,5),afsdata(1,3,5),afsdata(1,4,5),
!(4F12.6)
!*cfclose,array3b
!
/nopr
/output,junk_data_jr
*vabs,1,1,1,1
*vscfun,mx,max,afsdata(1,2,5)
*vabs,1,1,1,1
*vscfun,my,max,afsdata(1,3,5)
*vabs,1,1,1,1
*vscfun,mz,max,afsdata(1,4,5)
!
!save,iter%r%,db
!
*if,mx,ge,my,then
target=mx
*else
target=my
*endif
*if,target,le,mz,then
target=mz
*else
*endif
/output,term
/go,
w=%r%-1
```

```
*msg,info,r,w,mx,my,mz
THE MAXIMUM DELTA BETWEEN LOOP  %G AND  %G NODAL DEFLECTIONS ARE AS FOLLOWS: &
X= %g   Y= %g    Z= %g
*msg,info,target,airtol,
THE MAX. DELTA IS %G ,AIRTOL IS SET TO  %G
/nopr
/output,junk_data_jr
*if,target,le,airtol,exit
*else
*endif
!
!
!
!
!
/output,term
/go
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,         RUNNING INTERP_DEFLECT.X PROGRAM WITH TEMP.INPUT FILE
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/com,*************************************************************************
/NOPR
/output,junk_data_jr
!
*cfopen,deflect_airfoil,surf
*vwrite,sfnct
(F12.0,tl1,' ')
*vwrite,afsdata(1,1,4),afsdata(1,2,4),afsdata(1,3,4),afsdata(1,4,4)
(F7.0,tl1,' ',1X,F12.8,1X,F12.8,1X,F12.8)
*cfclose,defect_airfoil,surf!
!
/nopr
/output,junk_data_jr
/sys,rm SYSMSG
*cfopen,temp,input,
*vwrite,
```

```
('offdesign_blade_data.dat')
*vwrite
('airfoil.surf')
*vwrite,
('deflect_airfoil.surf')
*vwrite,
('mod_blade_data.dat')
*vwrite,
('mod_blade_point.dat')
*cfclose,temp,input
/show,term
/go
/sys,~kzq0fd/wip/NPSS_RELEASE/INTERP_DEFLECT/interp_deflect.x <temp.input
!
!/sys,cp mod_adpac.input adpac.input
!
/go
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,                        RUNNING CFD_WRAPPER.X PROGRAM
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/com,*********************************************************************
/NOPR
!
/sys,~kzq0fd/wip/NPSS_RELEASE/SCRIPTS/cfd_wrapper.x mod_blade_data.dat
/nopr
/output,junk_data_jr
!
!
*ENDDO
/GO
/output,TERM
/COM,*********************************************************************
*MSG,INFO,AIRTOL,r
CONVERGANCE OF %G WAS MET ON LOOP %G
/COM, THE CONVERGED "BLADE_POINT.DAT" FILE IS WRITTEN TO MOD_BLADE_POINT.DAT
/COM, THE CONVERGED "BLADE_DATA.DAT" FILE IS WRITTEN TO MOD_BLADE_DAT.DAT
```

```
/COM, ANSYS CONVERANGANCE PROCESS STOPPING !!!!!!!!!!!
/COM,*************************************************************************

/NOPR
/output,junk_data_jr
!
FINISH
/post1
set,last
alls
plns,u,sum
cmsel,s,surfnode
*get,sfnct,node,,count
*get,mnsfn,node,,num,min
*do,i,1,sfnct
afsdata(i,1,4)=mnsfn
afsdata(i,2,4)=ux(mnsfn)
afsdata(i,3,4)=uy(mnsfn)
afsdata(i,4,4)=uz(mnsfn)
mnsfn=ndnext(mnsfn)
*enddo
!
!
!
*cfopen,deflect_airfoil,surf
*vwrite,sfnct
(F12.0,tl1,' ')
*vwrite,afsdata(1,1,4),afsdata(1,2,4),afsdata(1,3,4),afsdata(1,4,4)
(F7.0,tl1,' ',1X,F12.8,1X,F12.8,1X,F12.8)
*cfclose,defect_airfoil,surf!
!
*cfopen,temp,input,
*vwrite,
('cold_blade_data.dat')
*vwrite
('airfoil.surf')
*vwrite,
('deflect_airfoil.surf')
*vwrite,
('new_blade_data.dat')
*vwrite,
('new_blade_point.dat')
*cfclose,temp,input
/sys,~kzq0fd/wip/NPSS_RELEASE/INTERP_DEFLECT/interp_deflect.x <temp.input
!
```

```
!
!
/sys,rm  blade.tmp
/sys,rm  fem.tmp
/sys,rm  new_blade.tmp
/sys,rm  q.tmp
/sys,rm  temp.input
/sys,rm  blade.converge
/sys,rm  blade.forces
/sys,rm  blade.p3dabs
/sys,rm  blade.p3drel
/sys,rm  blade.restart.new
/sys,rm  blade.boundata
/sys,rm  blade.input
/sys,rm  blade.mesh
/sys,rm  menust.tmp
/sys,rm  xansys.log
/sys,rm  xansys_bg.csh
/sys,rm  xansys.tmp
!
/output,term
/go
!
!  makes keypoints that show the location of the off design point airfoil
!
/nopr
/output,junk_data_jr
fini
!
/color,pbak,off                    ! turns the background of ansys to black
fini                               ! makes sure macro goes to begin level
/prep7                             ! Enters /prep7 of ANSYS
*del,jj                            ! deletes jj array if it has been defined
*dim,jj,array,1,2                  ! Defines jj array
*vleng,1,1
*vread,jj(1,1),new_blade_point,dat  ! reads the number of sections per blade
(7X,F7.0)
*vleng,1,1
*vread,jj(1,2),new_blade_point,dat  ! reads the number of points per section
(14X,F7.0)
n_pt=jj(1,1)*jj(1,2)               ! calculates the total number of points
Nsec=jj(1,1)                       ! sets Nsec equal to number of sectoins
Npts=jj(1,2)                       ! sets Npts equal to number of points/sec.
Tg1=11                             ! hardware tangency points, can read in
Tg2=70                             ! from file in future release
```

```
Tg3=91
Tg4=150
*del,bladept
*dim,bladept,array,n_pt,3       !read in x,y,z points from file,skip 1st row
*vread,bladept(1,1),new_blade_point,dat,,,,,,1
(2X,F12.5)
*vread,bladept(1,2),new_blade_point,dat,,,,,,1
(16X,F12.5)
*vread,bladept(1,3),new_blade_point,dat,,,,,,1
(30X,F12.5)
k=1
*do,i,1,Nsec      ! makes kps at each section,offset by 1000
*do,g,1,Npts
k,,bladept(k,1),bladept(k,2),bladept(k,3)
k=k+1
*enddo
*enddo
kplo
!
!  makes key points that show the original position.
!
/nopr
fini
!
/color,pbak,off                    ! turns the background of ansys to black
fini                               ! makes sure macro goes to begin level
/prep7                             ! Enters /prep7 of ANSYS
*del,jj                            ! deletes jj array if it has been defined
*dim,jj,array,1,2                  ! Defines jj array
*vleng,1,1
*vread,jj(1,1),orig_blade_point,dat  ! reads the number of sections per blade
(7X,F7.0)
*vleng,1,1
*vread,jj(1,2),orig_blade_point,dat  ! reads the number of points per section
(14X,F7.0)
n_pt=jj(1,1)*jj(1,2)               ! calculates the total number of points
Nsec=jj(1,1)                       ! sets Nsec equal to number of sectoins
Npts=jj(1,2)                       ! sets Npts equal to number of points/sec.
Tg1=11                             ! hardware tangency points, can read in
Tg2=70                             ! from file in future release
Tg3=91
Tg4=150
*del,bladept
*dim,bladept,array,n_pt,3       !read in x,y,z points from file,skip 1st row
*vread,bladept(1,1),orig_blade_point,dat,,,,,,1
```

```
(2X,F12.5)
*vread,bladept(1,2),orig_blade_point,dat,,,,,,1
(16X,F12.5)
*vread,bladept(1,3),orig_blade_point,dat,,,,,,1
(30X,F12.5)
k=1
*do,i,1,Nsec      ! makes kps at each section,offset by 1000
*do,g,1,Npts
k,,bladept(k,1),bladept(k,2),bladept(k,3)
k=k+1
*enddo
*enddo
kplo
/sys,rm junk_data_jr
/output,term
/go
*get,finish,active,,time,wall
total=finish-sttime
total=total*60
*cfclose,finish,time
*vwrite,total
(' The total time to solve the off design point is ',f10.2,'minutes')
*cfclose,finish,time
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2006	Final Contractor Report—March 2000–January 2002

4. TITLE AND SUBTITLE

NPSS Multidisciplinary Integration and Analysis

5. FUNDING NUMBERS

WBS–22–302–15–20
NAS3–98003, Task 5

6. AUTHOR(S)

Edward J. Hall, Joseph Rasche, Todd A. Simons, and Daniel Hoyniak

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

Rolls-Royce Corporation
2355 S. Tibbs Avenue
Indianapolis, Indiana 46241

8. PERFORMING ORGANIZATION REPORT NUMBER

E–15261

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

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2006-213890

11. SUPPLEMENTARY NOTES

Project Manager, Donald Van Drei. Responsible person, Joseph P. Veres, Propulsion Systems Division, NASA Glenn Research Center, organization code RTT, 216–433–2436.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 02 and 07

Available electronically at http://gltrs.grc.nasa.gov

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

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The objective of this task was to enhance the capability of the Numerical Propulsion System Simulation (NPSS) by expanding its reach into the high-fidelity multidisciplinary analysis area. This task investigated numerical techniques to convert between cold static to hot running geometry of compressor blades. Numerical calculations of blade deformations were iteratively done with high fidelity flow simulations together with high fidelity structural analysis of the compressor blade. The flow simulations were performed with the Advanced Ducted Propfan Analysis (ADPAC) code, while structural analyses were performed with the ANSYS code. High fidelity analyses were used to evaluate the effects on performance of: variations in tip clearance, uncertainty in manufacturing tolerance, variable inlet guide vane scheduling, and the effects of rotational speed on the hot running geometry of the compressor blades.

14. SUBJECT TERMS

Turbomachinery; Computational fluid dynamics; Computer simulation; Concurrent engineering; Structural analysis

15. NUMBER OF PAGES

179

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	Unclassified	