ABSTRACT

Title of dissertation:	CAD-BASED MODELING OF ADVANCED ROTARY WING STRUCTURES FOR INTEGRATED 3-D AEROMECHANICS ANALYSIS
	William Staruk, Doctor of Philosophy, 2017
Dissertation directed by:	Professor Inderjit Chopra Professor Anubhav Datta Department of Aerospace Engineering

This dissertation describes the first comprehensive use of integrated 3-D aeromechanics modeling, defined as the coupling of 3-D solid finite element method (FEM) structural dynamics with 3-D computational fluid dynamics (CFD), for the analysis of a real helicopter rotor. The development of this new methodology (a departure from how rotor aeroelastic analysis has been performed for 40 years), its execution on a real rotor, and the fundamental understanding of aeromechanics gained from it, are the key contributions of this dissertation. This work also presents the first CFD/CSD analysis of a tiltrotor in edgewise flight, revealing many of its unique loading mechanisms. The use of 3-D FEM, integrated with a trim solver and aerodynamics modeling, has the potential to enhance the design of advanced rotors by overcoming fundamental limitations of current generation beam-based analysis tools and offering integrated internal dynamic stress and strain predictions for design. Two primary goals drove this research effort: 1) developing a methodology to create 3-D CAD-based brick finite element models of rotors including multibody joints, controls, and aerodynamic interfaces, and 2) refining X3D, the US Army's next generation rotor structural dynamics solver featuring 3-D FEM within a multibody formulation with integrated aerodynamics, to model a tiltrotor in the edgewise conversion flight regime, which drives critical

proprotor structural loads. Prior tiltrotor analysis has primarily focused on hover aerodynamics with rigid blades or forward flight whirl-flutter stability with simplified aerodynamics. The first goal was met with the development of a detailed methodology for generating multibody 3-D structural models, starting from CAD geometry, continuing to higher-order hexahedral finite element meshing, to final assembly of the multibody model by creating joints, assigning material properties, and defining the aerodynamic interface. Several levels of verification and validation were carried out systematically, covering formulation, model accuracy, and accuracy of the physics of the problem and the many complex coupled aeromechanical phenomena that characterize the behavior of a tiltrotor in the conversion corridor. Compatibility of the new structural analysis models with X3D is demonstrated using analytical test cases, including 90° twisted beams and thick composite plates, and a notional bearingless rotor. Prediction of deformations and stresses in composite beams and plates is validated and verified against experimental measurements, theory, and state-of-the-art beam models. The second goal was met through integrated analysis of the Tilt Rotor Aeroacoustic Model (TRAM) proprotor using X3D coupled to Helios – the US Army's next generation CFD framework featuring a high fidelity Reynolds-average Navier-Stokes (RANS) structured/unstructured overset solver – as well as low order aerodynamic models. Although development of CFD was not part of this work, coupling X3D with Helios was, including establishing consistent interface definitions for blade deformations (for CFD mesh motion), aerodynamic interfaces (for loads transfer), and rotor control angles (for trim). It is expected that this method and solver will henceforth be an integral part of the Helios framework, providing an equal fidelity of representation for fluids and structures in the development of future

advanced rotor systems. Structural dynamics analysis of the TRAM model show accurate prediction of the lower natural frequencies, demonstrating the ability to model advanced rotors from first principles using 3-D structural dynamics, and a study of how joint properties affect these frequencies reveals how X3D can be used as a detailed design tool. The CFD/CSD analysis reveals accurate prediction of rotor performance and airloads in edgewise flight when compared to wind tunnel test data. Structural blade loads trends are well predicted at low thrust, but a 3/rev component of flap and lag bending moment appearing in test data at high thrust remains a mystery. Efficiently simulating a gimbaled rotor is not trivial; a time-domain method with only a single blade model is proposed and tested. The internal stress in the blade, particularly at its root where the gimbal action has major influence, is carefully examined, revealing complex localized loading patterns.

CAD-BASED MODELING OF ADVANCED ROTARY WING STRUCTURES FOR INTEGRATED 3-D AEROMECHANICS ANALYSIS

by

William Staruk

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements of the degree of
Doctor of Philosophy
2017

Advisory Committee:

Professor Inderjit Chopra, Co-Chair
Professor Anubhav Datta, Co-Chair
Professor James Baeder
Professor Olivier Bauchau
Professor Amr Baz

Acknowledgments

There are so many people who have contributed to this research and helped me on the path to producing this dissertation. This work has been a collaborative effort, only successful due to the assistance provided by my colleagues at Maryland, NASA, the US Army, and Sandia National Laboratories. I do not have the space here to list every single person who influenced me over the years, but I would like to acknowledge some of the most important contributors.

First, let me thank my professors at the University of Maryland. Inder Chopra has guided me for the past seven years, starting from my first day with the human powered helicopter project and through the end of my research. He is a brilliant engineer and a supportive leader, who, despite managing an enormous research group, finds the time to listen to and help every single student under his care. I am proud to be the latest in a line of tremendous rotorcraft engineers that have grown under the tutelage of Dr. Chopra.

Anubhav Datta is responsible for conceiving the CAD-based Modeling of Advanced Rotary Wing Structures (CMARS) project, and for choosing to partner with the University of Maryland when it first began and he was still a contractor with AFDD. We have worked closely together for four years on this project, and he has provided me with both enduring trust and unlimited patience. Every single time I met with Anubhav, even when things were going poorly, he managed to raise my spirits and bolster my confidence, and I owe him a tremendous debt of gratitude.

I would also like to thank James Baeder, Olivier Bauchau, and Amr Baz, the remaining three members of my committee. All three have offered me instruction and

advice, in the classroom and at meetings of the committee. It is an honor to have such a distinguished committee sending me off to the next phase of my career. Dr. VT Nagaraj is also owed thanks, for teaching me a tremendous amount about the practical concerns of helicopter design, which proved invaluable in learning how to create detailed model rotor heads. Thanks also to the department chair, Norm Wereley, and the dean of the Clark School, Darryll Pines; they were always encouraging and a pleasure to work with.

Next, I would like to thank our partners at the NASA Ames Research Center Aeromechanics Branch. In the single most important moment for this research project, William Warmbrodt agreed to take me onboard as an intern in the summer of 2014. He treated me with care and respect, always available to talk and lend advice. This feat is truly remarkable because I know he offered every single one of the other 50-plus interns the same degree of personal attention. Without that internship I never would have met Larry Young, who deserves a great deal of thanks for sharing the TRAM drawings with me. Not only did he provide me with the resources that were essential for a tremendous portion of this research, he and Bill also reviewed every one of my papers for publication. I know I didn't make it easy for them, and I am forever indebted to them for their patience. Thanks as well to Dr. Wayne Johnson, my mentor for that internship. In addition to guiding me in my literature survey on tiltrotor design and answering my questions on gimbaled rotor dynamics, he was the one that suggested I model the TRAM rotor for my dissertation. Thanks also to Cecil Wally Acree, who provided airfoil tables for the LCTR proprotor. Finally, thanks to everyone else at NASA who were so welcoming, including my officemate Natasha (Barbely) Schatzman, Meredith Segall, Eduardo Solas, and everyone else in 243R.

This work was sponsored by the US Army AFDD, and without them I would still be searching for a research topic. Thanks to Roger Strawn for his unwavering support and enthusiasm for this project. Without his vision and belief, this project would never have gotten so far. Thanks to Tom Maier, who served as a technical monitor for this work under the Vertical Lift Research Center of Excellence (VLRCOE, grant number W911W61120012). In addition to funding and supervision, several researchers at AFDD offered technical aid and are owed thanks, including Buvana Jayaraman who helped with Helios coupling, and Mark Potsdam who provided the TRAM surface needed for CAD modeling. Thanks also to Robert Ormiston, an ardent supporter of the Gamera project, who offered advice on both human powered helicopters and my Ph.D. research.

Next, thanks to the Cubit development team at Sandia National Laboratories. They are an extraordinary group, not just in terms of their brilliant efforts in scientific computing, but also for their willingness to offer detailed support to absolute strangers. While I corresponded with several who have my gratitude, I would like to especially thank Byron Hanks, who provided in person meshing support. I have rarely seen such generosity as I did in Albuquerque, and will never forget it.

There are a few other members of the larger rotorcraft community I would like to thank. Martin Annett at NASA Langley provided advice on composite materials modeling, and also led an awesome tour of the LandIR facility. François Richez from ONERA saw me present at the European Rotorcraft Forum and pointed out an interesting phenomenon in the CFD wake that I had not noticed. Finally, thanks to Blanche Demaret and Anne Brindejonc who took interest in my work and also helped me with translation in Lille when my French knowledge reached its limits.

I would also like to acknowledge everyone else at the University of Maryland who made my time here so memorable. To the entire Gamera team, so many of whom I worked with beyond the human powered helicopter, including Joe Schmaus, Elizabeth Ward, Graham Bowen-Davies, Ben Berry, Ben Woods, Mor Gilad, Chen Friedman, Brandon Bush, PK Koliais, Zak Kaler, Brandon Gudenius, Lauren Trollinger, and many others. Thanks are due to the members of my AHS Design team, where I learned more about helicopters than anywhere else: Anish Sydney, Joe Schmaus, Nitin Sydney, Conor Stahlhut, Jon Elliott, and Mathieu Amiraux. My gratitude also extends to the research faculty of the Alfred Gessow Rotorcraft Center who were always willing to listen and advise, including Ananth Sridharan, Bharath Govindarajan, and Vikram Hrishikeshavan. And I would also like to thank all of the many other friends I have made here at Maryland who made this endeavor as enjoyable as it was educational; the former and senior students, including Jeremy Knittel, Ericka Hocking, Jillian Alfred, Doug Szczublewski, Kate Thorne, Steve Sherman, and James Lamel Lankford; and the newer students, including Dan Escobar, Tyler Sinotte, Peter Oas, Brandyn Phillips, Wanyi Ng, Fred Tsai, Brent Mills, Chris O’Reilly, and James Sutherland-Foggio, several of whom I mentored.

Finally, I would like to express my enduring gratitude to my family, including my parents Harry and Peggy, who have always supported me, no matter what decision I made, fearing the worst and hoping for the best. I think it came out a little closer to the latter than the former. To my siblings, Kathy, Barbara, Johnny, and Lizzy, all of who have made me incredibly proud. And to my lovely wife Maribeth; our relationship predates this research by two months, and I’ll never understand how you stuck with me through all of this.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

.dat	X3D mesh data file extension
.j.dat	X3D joint data file extension
.unv	I-DEAS Universal mesh data file extension
AEBS	**Ae**rodynamic section, **b**ottom **s**urface nodes
AELE	**Ae**rodynamic section, **l**eading **e**dge nodes
AETE	**Ae**rodynamic section, **t**railing **e**dge nodes
AEUS	**Ae**rodynamic section, **u**pper **s**urface nodes
AFDD	(US Army) Aeroflightdynamics Directorate
ANSI	American National Standards Institute
AGRC	Alfred Gessow Rotorcraft Center
APREPRO	An Algebraic Preprocessor for Parameterizing Finite Element Analyses
BEMT	Blade element momentum theory
B-rep	Boundary representation
CAD	Computer-aided design
CAMRAD	Comprehensive Analytical Model of Rotor Aerodynamics and Dynamics
CATIA	Computer Aided Three-Dimensional Interactive Application
CFD	Computational fluid dynamics
CGM	Convergence Geometric Modeler
CNBS	**C**hord-**n**ormal (deformation) section, **b**ottom **s**urface nodes
CNLE	**C**hord-**n**ormal (deformation) section, **l**eading **e**dge nodes
CNTE	**C**hord-**n**ormal (deformation) section, **t**railing **e**dge nodes
CNUS	**C**hord-**n**ormal (deformation) section, **u**pper **s**urface nodes
CREATE	Computational Research and Engineering Acquisition Tools and Environments
CSD	Computational structural dynamics
CSG	Constructive solid geometry
DNW	*Deutsch-Niederländische Windkanäle* (German-Dutch Wind Tunnels)
DoD	(US) Department of Defense
DoF	Degree(s) of freedom
FEA	Finite element analysis
FEM	Finite element method
FETI-DP	Finite element tearing and interconnecting – dual primal
FM	Figure of merit
FSI	Fluid structure interface
GMRES	**G**eneralized **M**inimum **Res**idual

Helios	**Heli**copter **O**verset **S**imulations
Hex8	8-noded hexahedral solid finite element (first order)
Hex27	27-noded hexahedral solid finite element (second order)
ID	Identifier, numerical tag used by Cubit
IGES	International Graphics Exchange Specification
IPB	Inboard pitch (centering) bearing
ITEM	Immersive Topology Environment for Meshing
ISO	International Organization for Standardization
JVX	**J**oint **V**ertical E**x**periment
LCTR	**L**arge **C**ivil **T**ilt**r**otor
LRA	LCTR Reference Airfoils
MBDyn	MultiBody Dynamics
NASA	National Aeronautics and Space Administration
NFAC	National Full-scale Aerodynamics Complex
ONERA	*Office National d'Etudes et de Recherches Aerospatiale*
OPB	Outboard pitch (centering) bearing
PTC	Parametric Technology Corporation
Quad4	4-noded quadrilateral solid finite element (first order)
Quad9	9-noded quadrilateral solid finite element (second order)
RCAS	Rotorcraft Comprehensive Analysis System
SAM	Structural analysis model
SAR	Structural analysis representation
SCBS	**S**tructural **c**ut section, **b**ottom **s**urface nodes
SCLE	**S**tructural **c**ut section, **l**eading **e**dge nodes
SCTE	**S**tructural **c**ut section, **t**railing **e**dge nodes
SCUS	**S**tructural **c**ut section, **u**pper **s**urface nodes
SI	*Système international d'unités* (international system of units)
STEP	Standard for the Exchange of Product model data
STL	Stereolithography
TRAM	Tilt Rotor Aeroacoustic Model
UMARC	University of Maryland Advanced Rotorcraft Code
VTOL	Vertical take-off and landing
X3D	Experimental 3-D Dynamic solver

Nomenclature

a	speed of sound, m/s
A	rotor area, m^2
c	local chord, m
$c_n M^2$	sectional normal force normalized by $\left({}^{1}/_{2}\,\rho a^2 c\right)$
$C_{l_{max}}$	maximum airfoil lift coefficient
C_P	rotor power coefficient, $P/\rho(\Omega R)^3 A$
C_T	rotor thrust coefficient, $T/\rho(\Omega R)^2 A$ (shaft axes)
C_X	rotor propulsive force coefficient, $X/\rho(\Omega R)^2 A$
EI	bending stiffness, Nm^2
f_C	aerodynamic chordwise force on a finite element surface node, N
f_N	aerodynamic normal force on a finite element surface node, N
F_C	aerodynamic chordwise force on an aerodynamic segment, N
F_N	normal force on an aerodynamic segment, N
F_x	force in the x-direction (extension), blade undeformed frame, N
GJ_x	torsional stiffness, Nm^2
K_{ij}	joint stiffness matrix component, N/m (translation) or N/rad (rotation)
$M_{1/4}$	aerodynamic pitching moment about the ¼ chord axis on an aerodynamic segment, Nm
M_{Tip}	blade tip Mach number, $\Omega R/a$
M_x	moment about the x-axis (torsion), blade undeformed frame, Nm
M_y	moment about the y-axis (flap), blade undeformed frame, Nm
M_z	moment about the z-axis (lag), blade undeformed frame, Nm
N_b	number of blades
p	arbitrary integer
P	rotor power, W
r	rotor radial station, m
R	rotor radius, m
Q	vorticity Q-criterion
T	thrust, N
u	deflection in the global x-direction (extension), m
v	deflection in the global y-direction (lag), m
V	wind tunnel velocity, m/s
w	deflection in the global z-direction (flap), m
X	rotor propulsive force (wind axes, positive forward), N
$\alpha_0, \alpha_1, \beta_0$	polynomial constants determining finite element surface node aerodynamic

	force
α_c	shaft angle of attack, corrected, deg.
α_s	shaft angle of attack, wind tunnel measurement, deg.
β_{1c}	longitudinal blade flap angle, deg.
β_{1s}	lateral blade flap angle, deg.
δ	wind tunnel wall correction constant
$\Delta\alpha$	shaft angle correction, $\alpha_c - \alpha_s$, deg.
η	aerodynamic surface node thickness-wise position
θ	blade pitch angle, deg.
θ_x	rotation angle about the x-axis (twist), blade undeformed frame, deg.
θ_y	rotation angle about the y-axis (flap), blade undeformed frame, deg.
θ_z	rotation angle about the z-axis (lag), blade undeformed frame, deg.
κ_h	induced tip loss factor, hover
μ	rotor tip speed ratio $V/(\Omega R)$
ξ	aerodynamic surface node chordwise position
ρ	air density, kg/m^3
σ	rotor solidity (0.105 for TRAM)
σ_{ij}	Cauchy stress component in the i, j direction, N/m^2
ψ	blade azimuth angle (0° is downstream), deg.
Ω	rotor rotation speed, rad/s

Chapter 1: Introduction

1.1 Motivation

Helicopters, due to their vertical take-off and landing (VTOL), hovering, and low speed controlled all-axes flight capabilities, have become an indispensable aviation asset. They complete many critical missions when fixed wing aircraft cannot, including medical evacuation, disaster relief, search and rescue, firefighting, law enforcement support, troop insertion, and close air support. Despite their utility, helicopter operations are hampered by slow speeds and limited range due to the poor cruise efficiency of a rotor in edgewise flow. In order to break through the limitations inherent to the conventional helicopter, aircraft designers have been devoting increased attention to new configurations, such as compound and tiltrotor rotorcraft, and new technologies, such as slowed rotors, advanced geometry blades, and morphing and mission adaptive blade geometries. Designing such aircraft, however, can be a long and expensive process. High fidelity comprehensive analysis tools are necessary during the design phase in order to keep development on track, reliably assess alternatives, and achieve the desired level of efficient performance for any new aircraft. This is especially true for designs incorporating advanced technology where no previous flight test data exist. With increases in fidelity and computational power, these tools can even be used for certification, virtual flight testing, and assessment of vulnerability to damage.

Consider, for example, the sole production example of a tiltrotor, the V-22 Osprey (Figure 1.1), which took decades to advance from preliminary design to deployment in the field. By using a pair of proprotors pointing upward, the tiltrotor can hover with efficiency approaching that of helicopters. In cruising flight, the proprotors are pointed forward into an axial flight condition to act as propellers, and the wing generates the necessary lift for flight. Although this concept has great demonstrated its effectiveness, in practice tiltrotor aircraft have suffered from limitations. Current generation tiltrotors have large and heavy proprotor blades and hubs, the weight of which are driven by the critical stresses these components experience during transition flight. Transition is the flight regime where the tiltrotor is picking up speed to switch over from edgewise to wing-borne flight, over what is known as the conversion corridor, and is characterized by the proprotors operating in edgewise flight at moderately high shaft angles, as depicted in Figure 1.1. In fact, the tiltrotors often spend a large portion of their mission time in this edgewise configuration, flying and maneuvering at low speed where the proprotors are subjected to high unsteady aerodynamic loads, similar to a helicopter rotor. Without accurate predictions of the stresses and strains experienced by the rotor components in this complex flight condition, proprotor components must be over-designed. In addition to the direct effect of increased rotor weight on vehicle sizing and performance, the heavy rotor results in a thicker wing (nominally 23% thickness to chord) to eliminate the possibility of whirl flutter instability in cruise, a major concern when designing tiltrotors. These thick wings suffer from increased compressibility drag, reducing cruise efficiency and speed which are the key benefits of implementing a tiltrotor configuration in the first place.

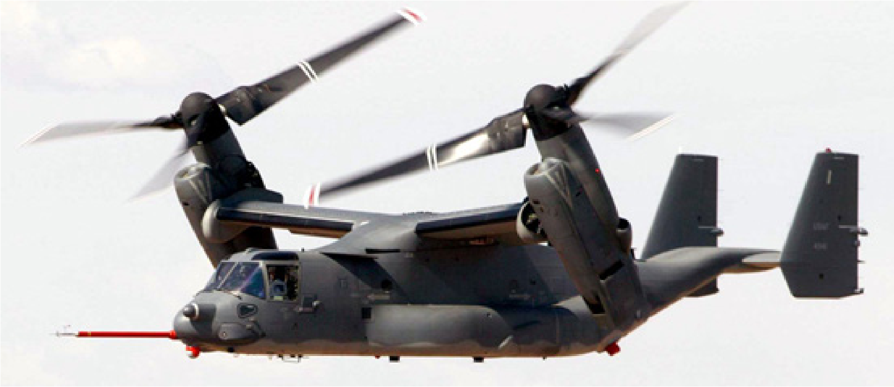


Figure 1.1 The V-22 Osprey in transition from the hover to cruise flight

Current computational aeromechanics utilizes what is known as comprehensive analysis (CA), which couple rotor aerodynamics, rotor structural dynamics, and aircraft flight mechanics models to analyze all aspects of helicopter flight [1, 2]. State-of-the-art analysis couples comprehensive analysis to advanced computational fluid dynamics (CFD) to model the complex rotor aerodynamics [3]. Computational structural dynamics (CSD), on the other hand, receives a simpler treatment. Typically, the rotor blade is modeled as a slender one-dimensional (1-D) beam, often in a multibody formulation, using two-dimensional sectional analysis tools to calculate the spanwise structural properties of the blade [4]. Although accurate and efficient for many current applications, 1-D beam-based analysis faces several limitations: it cannot model from first principles the non-slender 3-D hub structures of advanced rotors that determine the critical couplings important for stability and absorb the critical loads that drive weight; it cannot model airfoil chordwise flexibility important to advanced smart rotors; and it cannot model discontinuities in the blade structure present in advanced blade shapes or from ballistic damage. Three-dimensional analysis is used today for these reasons, but only on isolated pieces, with loads being applied from previous flight test data or low order comprehensive analysis. This process requires iterations (at the cost of time) and significant factors of safety (at the cost of weight). Today, modifications are made to beam-based methods to allow analysis of complicated rotor designs, but they require significant simplifications of the actual structure and almost always rely on a priori experimental measurements to derive equivalent blade root conditions [5]. In 2008, Johnson identified high fidelity structural analysis using three-dimensional (3-D) finite element analysis as a key requirement of next-generation rotor analysis [6].

Utilizing three-dimensional computational structural models has the potential to overcome all of the above limitations of conventional beam-based blade analysis. In the context of the blade, three-dimensional structures refers to the use of solid (brick) finite element analysis (FEA) that solve the 3-D governing equations in the structural domain with no reduced order approximation [7]. The 3-D structure can then be coupled with 3-D aerodynamics through a fluid-structure interface. This level of modeling, with full 3-D CSD coupled to full 3-D CFD, is defined as Integrated 3-D in this work.

The primary benefit of integrated 3-D analysis is not simply increased fidelity, but an increase in capability and the scope of modeling, enabling the prediction of true 3-D stresses and strains in flight; accurate representation of complex hubs, blades and flow fields; and the potential to model advanced geometry blades, morphing rotors, and battle damage. It has the potential to produce designs with higher performance, lighter weight, and lower acquisition and maintenance cost by allowing designers to consider advanced configurations and technologies with little reliance on expensive experimentation and less likelihood of encountering critical problems which require costly corrections late in the design cycle. Although undoubtedly more computationally intensive than conventional analysis, continued advancements in high performance computing resources are expected to enable the adoption of integrated 3-D analysis throughout the rotorcraft design process. Today, 3-D aerodynamics using Reynolds Averaged Navier-Stoke (RANS) CFD is routinely solved using on the order of 1,000 processors; taking advantage of these to include 3-D structural dynamics is expected to contribute little additional computational burden.

1.2 Organization of the Dissertation

The present work is focused on enabling integrated 3-D analysis by developing computer aided design (CAD) based rotor structural analysis models for use with X3D, a next-generation rotor structural dynamics solver [8]; refining the analysis for use in modeling a gimballed tilt rotor; and performing comprehensive RANS/FEM simulation for fundamental understanding of proprotor dynamics, loads, and stresses and strains in the conversion corridor.

Chapter 1 introduces the topic and begins with the history of rotor structural modeling. An introduction to X3D follows, and then a discussion of prior tiltrotor analysis. Finally, a brief review of CAD software is provided.

After the introduction is Chapter 2, which describes the methodology developed for creating rotor models for X3D. This includes a description of the unique requirements for rotor analysis, a detailed breakdown of the workflow (CAD, SAR, meshes, SAM, input decks), and a description of aerodynamic modeling. Examples of the model development technique are provided, including a notional bearingless rotor. Post-analysis methods for extracting equivalent 1-D beam sectional properties and determining integrated blade structural loads from the 3-D strain field are developed.

Chapter 3 presents verification and validation of X3D for structural dynamics analysis. Examples include analysis of isotropic and composite beams, thick composite plates, and the bearingless rotor model introduced in Chapter 2.

Chapter 4 introduces the Tilt Rotor Aeroacoustic Model (TRAM) proprotor and details the modeling efforts performed for this work. The development of a 3-D structural analysis model, using the methodology presented in Chapter 3, is described. Aerodynamics modeling is also discussed, including low-order lifting line theory and RANS. Finally, a method for modeling gimbal dynamics using a single blade is proposed.

Chapter 5 presents key results of analysis of the TRAM rotor. Structural dynamics is examined, with a focus on rotor frequencies as the full 3-D model is built up from a simple representation. Hover and (edgewise) forward flight performance are examined using a variety of aerodynamics models. Forward flight airloads and structural blade loads are also examined, with comparisons between the aerodynamic models being used to draw conclusions about the sources of loading. The full 3-D stress field is also examined, with a focus on the impact of gimbal modeling on the stress at the root of the blade.

Chapter 6 provides, separately, a detailed discussion of some interesting results. Its focus is on drawing out fundamental understanding gained from this work, and examining how integrated 3-D analysis can be used to study rotor design. Finally, Chapter 7 provides some concluding remarks and recommendations for future work.

1.3 Rotor Aerodynamic Modeling and Comprehensive Analysis

Aerodynamics modeling for rotor blades can be thought of as having two distinct parts. The first is the airloads model, which calculates aerodynamic forces on a blade section caused by air moving over the airfoil. The second portion requires modeling the velocity of the air around the blade. This includes direct air velocity due to rotation of the rotor and gusts, but also the motion of the aircraft, requiring a flight dynamics model; inflow due to the influence of the blades on the air, with the wake causing induced velocities at the rotor disk; and relative velocity due to motions of the blade due to rigid body or elastic deflections, which depends on a rotor structural model and trim solution. Today, all of the required disciplines are solved together in what is known as a comprehensive analysis [2], which couples airloads, wake models, blade structural dynamics, and flight dynamics and controls into a single analysis. While this dissertation largely focuses on blade structural modeling, a description of the state-of-the-art in aerodynamics modeling is valuable.

1.3.1 Sectional Airloads

The simplest sectional airloads models use steady 2-D linear airfoil theory, which multiplies the angle of attack of the blade – calculated based on tangential velocity, inflow, and blade pitch – by a known lift curve slope to calculate sectional lift. However, the flow experienced by real rotors is much more complex. Blade pitching motions from controls and elastic deflections are unsteady, providing a rate of change to the angle attack. Flapping of the blades leads to plunging motion. The inflow due to the rotor wake causes gust-like velocities. Capturing these complexities accurately requires an unsteady airloads model.

A simple form of analysis, which can be called a pseudo-steady model, combines these unsteady effects and uses them to calculate an equivalent angle of attack at the sectional ¾-chord point. Airloads are then calculated by correlating the angle of attack and local Mach number to airfoil look-up tables, developed using wind tunnel tests or, when test data is not available, CFD analysis. This model can be refined by including unsteady non-circulatory airloads using quasi-steady thin airfoil theory. Fully unsteady sectional airloads can be calculated by using classical Theodorsen theory in the frequency domain to apply phase and magnitude corrections, thereby accounting for near wake effects. A time domain theory is more useful in the context of helicopter aerodynamics, and so Wagner's indicial response theory can be used instead to account for the shed wake [9].

In recent years, the focus has been on developing more accurate models of unsteady aerodynamics. Semi-empirical models have been developed from wind tunnel testing of oscillating airfoils in an attempt to capture viscous effects, static and dynamic stall, and compressibility effects. Dynamic stall is a phenomenon that occurs on a dynamically pitching airfoil in which flow separation is delayed until a higher angle of attack than expected under static flow conditions. This is followed by high transient loads caused by a shed leading ledge vortex when the section does stall. Both the separation delay and the leading edge vortex effects on airloads must be modeled. An influential semi-empirical indicial unsteady aerodynamics model was developed by Leishman and Beddoes in the 1980s [10, 11] , and extended to account for nonlinear effects due to flow separation [12] and dynamic stall, with and without blade sweep [13, 14]. Other dynamic stall models include those by Johnson [15], Boeing [16], ONERA EDLIN (équations différentielles linéaires) [17], and ONERA BH (bifurcation de Hopf) [18]. All of these are 2-D models built upon layers of corrections to the underlying 2-D airfoil tables. Johnson found that, while all of these models correlate well to one another, there are significant errors when compared to tests of an oscillating 3-D wing [19].

1.3.2 Inflow Modeling

The simplest aerodynamic models, dating back to Galuert in 1926, model the inflow through the rotor disk as uniform. However, even at that early stage, it was known that this was only a crude approximation. In the 1960s, efforts were made to more accurately model the rotor wake geometry. A prescribed-wake model was developed by Piziali and DuWaldt, in which a shape of the wake is assumed to be rigid, with a shape inferred from measurements that does not distort with time, and the vorticity of the wake causes an induced inflow at the blade [20, 21]. Prescribed wake modeling was greatly advanced by Landgrebe using experimental measurements of rotor inflow in hover [22, 23]. The modern free-wake solution, where the shape of the wake is part of the solution and the filaments are allowed to distort with time, was formulated by Scully. The wake was allowed to distort as it convected with the flow using a time-marching algorithm [24]. This time-marching method suffered from convergence issues, however, and Scully developed a more robust relaxation wake model, in which the wake is assumed to be periodic [25].

The development of more accurate wake models (including semi-empirical or empirical models for vortex core size, core growth with time, roll-up mechanism of the tip vortex, and more precise calculation of the wake distortion) has continued over the past five decades. Prescribed wake models have been further refined and extended to forward flight [26, 27, 28]. Modern free-wake models using relaxation methods include the constant velocity contour method developed by Quackenbush and Wachspress [29], the pseudo-implicit predictor-corrector method of Bagai and Lesihman [30] (Marylande free wake), and the general free-wake geometry method from Johnson, including refinements for multiple tip vortices and a consolidation method for treating tip vortex roll-up [31, 32]. Time marching free-wake methods were further developed, with the goal of correcting numerical instabilities by Clark and Leiper [33], Landgrebe [22], and Sadler [34]. Bhagwat and Leishman [35] [36] developed a time accurate free-wake method (Maryland time-accurate free wake), which has seen continued refinement [37]. The relaxation wake (periodic) method is useful for trim solutions, whereas the time accurate (transient) method is needed for maneuvers. These wake methods are essential for vibration predictions. For stability calculations, however, they are not very suitable, as the non-linear nature of the problem (from the Biot-Savart law) makes it difficult to formulate in state-space.

For stability dynamic inflow models are used instead, even though they are lower fidelity models and unsuitable for loads and vibration predictions. The dynamic inflow models originated from the work of Mangler and Square [38] and put in its modern form by Joglekar and Loewy [39]. The formulation was completed and refined by Pitt and Peters [40]. Dynamic inflow presents a faster and more efficient state space alternative to vortex wake methods. These methods represent the inflow using an assumed distribution over the disk as a Fourier series whose coefficients are related to the net rotor thrust and moments. The original Pitt-Peters model was further extended by Peters and He [41], and this model has seen widespread adoption. While they do not model the wake geometry, dynamic inflow models are much less computationally expensive than free-wake models, and have been found to be well suited for aeroelastic stability analysis [42, 43].

1.3.3 Comprehensive Analysis and CFD/CSD Coupling

Comprehensive rotor analysis is defined by Johnson as a solver which is "applicable to a wide range of problems (performance, loads, vibration, response, stability) and operating conditions (hover, cruise, maneuver), and a wide class of configurations (blades, hubs, rotors, aircraft), in all stages (research, conceptual design, detailed design, development) and all aspects (design, testing, evaluation) of the engineering process." While meeting all of those criteria has only occurred in recent years with modern multibody analyses (discussed below in Section 1.4: Rotor Structural Modeling), the term comprehensive analysis has generally been used to described any rotorcraft analysis code which combines aerodynamic modeling, structural modeling, aircraft flight dynamics, and controls (in the form of a trim solution). These codes typically have multiple options for analyzing rotor aerodynamics, including various stall and wake models.

As an alternative to utilizing the methods described above, computational fluid dynamics (CFD) can be used to predict the airloads on a blade. A CFD analysis directly models the flow of air around the blade and the aircraft, as well as the airloads generated over a blade section or fuselage body by solving the Navier-Stokes equations (or approximations based on them). Thus airloads and wake structure are solved at once, eliminating the need for separate airloads and inflow models. While CFD tools have proven to be powerful, capable of accurately resolving airloads for a range of rotorcraft problems [3], they are purely aerodynamic models, and must be coupled to computational structural dynamics (CSD) and trim solvers to calculate the performance, loads, and stability characteristics needed for actual design of the rotor. This task is referred to as CFD/CSD coupling.

While the term CFD/CSD coupling is used because of the symmetry of its two terms, it could be more accurately described as CFD/CA (comprehensive analysis) coupling. In this formulation, a comprehensive analysis code is used to predict structural deformation and trim solution, while the CFD code calculates the airloads. In general, the procedure begins with a CA solution using its own low-order (often uniform inflow) aerodynamics. The deformations are then passed to the CFD solver, which deforms its mesh to match the CA solution and calculates high-fidelity airloads. These airloads are then passed back to the CA solution, which takes the difference between the airloads passed by the CFD and its own internal predictions. This difference (or delta) is then applied and a new blade deformation solution is computed. The new deflections are passed to CFD and the process repeats iteratively until convergence. Because the input to CA is a difference in airloads, this method is known as a delta coupling procedure. While there is some debate over terminology [3], generally speaking, delta coupling can be carried out once per revolution – with CA performing a full trim solution, before passing deformations to CFD, and CFD passing airloads for the entire revolution to CA – referred to as loose coupling, or with iterations at every time step, known as tight coupling, the latter of which is necessary for stability analysis and maneuver simulations.

1.4 Rotor Structural Modeling

The earliest rotor dynamics theories, dating back to the first formulation of blade element theories for aerodynamics and the rotor inflow in edgewise flight by Glauert in

1926, treated rotors as rigid blades flapping about hinges at their roots [44]. In their seminal paper in 1958, Houbolt and Brooks [45] first applied beam theory to formulate linearized equations of motions for elastic blades undergoing flap and lag bending and torsion deformations with a single load path to the hub. In the 1960s, interest in developing new hingeless rotor helicopters, including the AH-56 Cheyenne, demanded more detailed theories of blade elasticity that accounted for large deformations for stability analysis [4]. In 1974, Hodges and Dowell established a general non-linear (up to second order non-linearities) theory for coupled flap/lag/torsion dynamics of rotor blades [46]. The following years saw efforts to expand the detail of blade analysis to include exact kinematics, higher order non-linearities, and multiple load paths, including work by Ormiston and Hodges [47], Kvaternik [48], Rosen and Friedmann [49], and Johnson [50]. All of these formulations used Euler-Bernoulli beam theory for isotropic materials. These beam models were well suited for analysis using the finite element method (FEM) [51, 52]. While FEM initially was used to solve for blade structural modes, today’s computers are powerful enough to allow full solutions.

The next key step in rotor structural theory was the development of beam theory for anisotropic, composite materials: Bauchau and Hong (1988) presented a non-linear elastic theory for thin-walled (though the theory was more general) beams undergoing large deflection with small strain which included warping and extension-torsion coupling from anisotropic materials [53]; Hodges (1990) used variational principles to derive equations of motion for anisotropic beams with closed cross-sections and unrestrained warping [54]; and Smith and Chopra (1993) developed an analysis for a blade with a composite laminate box spar [55].

The current state-of-the-art came into being with the development of multibody dynamics analysis tools for rotorcraft comprehensively analysis in the 1990s and 2000s, including DYMORE [56], CAMRAD II [57], MBDyn [58], RCAS [59], and UMARC-2 [60]. Multibody dynamics uses kinematic joints undergoing large rotations connecting structural components; thus the analysis allows different structural pieces to be linked together while allowing large rigid-body relative motions. This enables analysis of complex rotors built up from component pieces. Further, it allows the use of lower deformation beam elements (such as the Hodges and Dowell second order formulation) to be linked together to achieve large deformations. Since their development, use of such multibody comprehensive analysis tools has become commonplace for analysis of complex rotors. Recent work has focused primarily on developing improved means for calculating 2-D sectional properties of composite rotor blades using cross-sectional finite element analysis (to calculate warping terms), and continually refining and improving beam elements to correct for exact geometry, strains, restrained warping, and more, with every new rotor geometry requiring some new correction [61, 62, 63, 64, 65].

Limited work has been performed on analyzing rotor structural dynamics with higher dimensional models than beams. Shell elements have been employed by the wind turbine community for stress analysis, including Bazilevs et al. who presented a CFD/CSD study of a full scale wind tunnel rotor using a shell-based structural model [66] and found good prediction of the blade tip deflection. The use of shell elements in multibody dynamic analysis was discussed by Bauchau and Bottasso [67, 68]. In 2014, Kang et al. developed a shell element for implementation in the RCAS comprehensive analysis tool, assuming large deformations and rotations with small strain [69]; the shell element was designed to be connected to nonlinear beam elements already used in RCAS, allowing reduced computation time by only using shells for a portion of the blade. Comparisons of natural frequency and static deflections predictions were made between combined shell/beam and beam-only models, revealing a good match for slender geometry.

Analysis using 3-D FEM represents a fundamental departure from lower-dimensional structural models (beams and shells) because 3-D element nodes do not carry rotational degrees of freedom to which joints can be easily connected, instead resolving the motion directly through the 3-D displacement field. Yeo and Truong examined the use of 3-D FEA for rotor analysis, using MSC/Marc for rotating beam analysis [70, 71]. Results from this work were limited to a comparison of beam natural frequencies predicted by 3-D analysis to RCAS beam analysis. None of the work involved a real rotor (with neither internal structure nor hinge articulation) and hence did not proceed to include aeromechanics.

1.5 The X3D Structural Dynamics Solver

In 2006, the US Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP) Computational Research and Engineering Acquisition Tools and Environments (CREATE) program was launched with the goal of developing high performance computing (HPC) software tools to aid in the design of next-generation air vehicles and limit the increasing costs of their acquisition [72]. As part of the CREATE-air vehicle (CREATE-AV) initiative, the US Army Aeroflightdynamics Directorate (AFDD) was tasked with developing Helios, a next generation rotorcraft CFD simulation tool designed to take advantage of massive parallelization on future HPC systems. With an interest in exploiting increases in computing power, the desire for a 3-D rotor structural dynamics solver capable of overcoming the limitations of state-of-the-art beam-based analysis was identified [6].

Development of X3D, a next generation 3-D computational structural dynamics solver for rotor aeromechanics, began in 2008. In 2009, Datta and Johnson presented a parallelizable 3-D finite element method (FEM) solver for structural analysis of rotor blades for the first time [73]; this prototype would later form the core of X3D. The FEM analysis was formulated carefully, with an emphasis on non-linearities and inertial terms unique to rotating structures. The solver is built around a single element, the 27-noded hexahedral brick (Hex27) was chosen for its resistance to element locking (a numerical issue). Parallelization was achieved by using an iterative substructuring scheme with a finite element tearing and interconnecting – dual primal (FETI-DP) method used for structural partitioning. The iterative substructuring analysis is performed by a Krylov solver using either a generalized minimum residual (GMRES) update or a simple conjugate gradient (CG) update. This paper demonstrated algorithm scalability of the solution using cantilevered plate natural frequencies and deformation of an idealized UH-60A-like blade mesh in hover and forward flight.

Beam analysis relies on the fundamental assumption that the structure is one-dimensional. Although it can deform in three dimensions, the deformations can only vary along one dimension, and the beam properties only vary along this axis. While applicable for slender beams, where the axial dimension is much larger than the two cross-sectional dimensions, it breaks down for thick components. Additionally, beam theories require that the beam be continuous, though approximate corrections can be introduced for modeling rapid changes of properties. Finally, there is an assumption of small strain, though this is generally a good assumption for helicopter blades since large strain would lead to a mechanical failure. The X3D solver does not suffer from the first two assumptions, making it a good candidate for analyzing non-slender structures at blade roots and parts with discontinuous geometry, which are dominated by 3-D effects. X3D assumes linear material behavior. This is accurate for typical loading cases, however, it does prevent X3D from analyzing cases with significant material non-linearities, including crash impact and crack propagation studies. Beam models also incorporate linear materials assumptions. Dedicated crash models include the non-linear behavior, but they do not contain all of the gyroscopic terms needed to model rotation of the rotor.

In 2010, Datta and Johnson unified a multibody formulation within the 3-D FEM solver [74]. This required the use of special brick elements which could be connected to joints undergoing arbitrary displacements and rotations. The multibody formulation is essential for analysis of rotors, which typically include multiple components and load paths – including bolts, bearings, and dampers that admit large relative motion but no strains – which should be treated as kinematic joints rather than elastically deforming bodies. The multibody solution was verified by considering idealized articulated, hingeless, and bearingless rotor models. Scalability was demonstrated, and showed no detrimental impact from the inclusion of multibody joints.

In 2014, a fully integrated 3-D CSD/CFD aeromechanics solution for an idealized rotor was presented [75]. The term “Integrated 3-D” was coined to indicate this level of analysis, featuring 3-D FEM based multibody structural analysis coupled with 3-D CFD. An idealized, UH-60A-like blade mesh was attached to an articulated root flap/lag/torsion joint and a pitch link mesh for control. A rotor trim solution with low level aerodynamics was attached to the solver to form the X3D aeromechanics tool. X3D was coupled with Helios to perform the CFD/CSD analysis of the UH-60A-like rotor. Airloads were compared to measured data for high speed flight and internal blade axial stress was examined at several azimuth angles. Finally, in 2016, X3D version 1 was released [8]. All of this work, however, used idealized geometries; there was still no formal method or tool suitable for modeling complex, real-world rotor structures.

The research performed for this dissertation was conducted in parallel with the development of X3D, starting in March of 2013. The focus of the current work has been on developing a methodology for creating CAD-based meshes of generic rotor blades, and of using that methodology to develop a structural analysis model of an advanced tilt rotor for integrated 3-D analysis. Success of that endeavor led to an extension of its scope: coupling with Helios CFD for a full-up simulation of a tiltrotor in conversion mode flight and identifying the many unique and complex loading mechanisms inherent to this flight regime. Several papers were published to document the progress of this research effort [76, 77, 78, 79, 80, 81]. Additionally, Ward conducted research using X3D to analyze a composite blade with extension-torsion coupling in an idealized, UH-60A-like slowed rotor [82, 83, 84].

1.6 Tiltrotor Analysis

There is a large body literature on aerodynamic analysis of tiltrotors devoted to performance and flow field predictions in hover – from early work in the 1990s [85, 86, 87, 88] to more recent RANS simulations since 2000 [89, 90, 91, 92, 93, 94]. Blade elasticity has little impact on performance in hover [95], so no structural modeling was required for these studies. There is no gimbal motion in ideal hover and the trim solution is very simple and almost always left out; a measured collective is prescribed instead.

Tiltrotor aeroelasticity research has largely been devoted to whirl flutter stability in cruise – from early pioneering work [96, 97, 98, 99] to more recent studies [100, 101, 102, 103, 104, 105, 5, 106]. The proprotor induced inflow is less important in cruise than helicopter mode operation: the wake is washed away in high speed axial flight. High fidelity aerodynamic modeling does not appear important for these types of studies and a uniform inflow assumption typically suffices, and no wake trajectory is calculated.

This separation breaks down during flight in the conversion corridor where the tiltrotor encounters significant edgewise flow and works more similarly to a helicopter rotor. A proprotor encounters very high oscillatory and vibratory loads in this regime due to its very stiff blades. The fundamental understanding and accurate prediction of these structural loads are essential for the design of light-weight proprotors and low-drag wings for future tiltrotor aircraft.

There have only been limited attempts at predicting loads in the conversion regime. This is partly due to a lack of test data and partly due to the difficulty of modeling the physics, which require a very detailed analysis. A proprotor wake is complex; characterized by stall delay, inboard vortex sheet development, outboard trailer consolidation, and negative lift roll-up near the tip – unique phenomena that require extensive semi-empirical corrections to a conventional helicopter lifting-line free-wake analysis. In addition to aerodynamic complexities, proprotors are challenging to analyze structurally. Blades and hubs encompass spanwise discontinuities and 3-D flexible parts near the root that are connected along multiple load paths through multiple joints. Such a structure is difficult to collapse neatly into equivalent boundary conditions for a conventional beam-type analysis.

Early works by Bilger et al. [107] and Totah and Madden [108] presented simplified calculations that ignored these complexities. The most comprehensive analysis thus far was carried out by Johnson [95, 109, 110, 32] who used systematic validation with TRAM experimental data to establish the important corrections required, for both lifting-line aerodynamics and beam structural dynamics. His study revealed two fundamental barriers: 1) the advancing blade wake induced loading was difficult to predict consistently; corrections that improved airloads prediction deteriorated performance prediction and vice versa, and 2) the root structural conditions were difficult to idealize; deflection and rap tests suggested different corrections and none produced rotating lag frequencies that would be acceptable, so ad hoc corrections were needed. Resolving these barriers in tiltrotor analysis through integrated 3-D CFD/CSD modeling is one the key focus areas of this dissertation. In 2017, Ho and Yeo performed another analysis of the TRAM using RCAS, with a similar fidelity as Johnson, with a focus on blade loads predictions [111], and found similar results.

1.7 Computer Aided Design (CAD)

Computer aided design (CAD) is a term for a range of software tools which are used for generating geometry models of engineering products. The inherent goal of CAD software is to allow a designer to create geometry, store that geometry as a mathematical model in computer memory, and recall the geometry and display it to a user either on a screen or a printout. CAD has become an indispensable part of modern industry in many fields, including engineering, architecture, art, and archaeology [112].

Early CAD tools were limited to 2-D geometry and simple 3-D wireframe geometry, composed of vertices connected by curves, with no capability to define surfaces or solids. While useful for visualization, these methods have obvious limitations. Only simple wireframe representations of the geometry are possible so, for example, edges on the far side of a volume will not be concealed from view by the body of a solid because that solid is not mathematically defined. The first true 3-D solid modeling methodology developed was the boundary representation, or B-rep, technique [113]. With B-reps, the geometry is still defined by surfaces bounded by curves with vertices at their end points, but the surfaces have a defined directionality; each surface has a normal direction defined as pointing inward toward the solid and an opposite normal defined as pointing outward, toward open space. Further, mathematical tools ensure the surfaces of the B-rep are closed, forming a "water tight" volume. With the combination of closed surfaces and directionality, solids can be rigorously defined.

While useful for visualizing geometry, B-rep models are difficult to create. This was remedied by the use of constructive solid geometry, or CSG [114]. In CSG, solids are constructed using collections of 3-D geometric primitives, such as cubes, spheres, and cylinders. Boolean operations are used to join, subtract, and intersect these primitives to form more complex geometries. This simplifies the design process, allowing primitives to be modified using parameters (e.g., height and radius for a cylinder). While still in use today, CSG faces several limitations, including difficulty in modeling complex geometry, and the inability to reflect design intent.

The next major evolution CAD was the development of feature based design [115]. Like CSG, feature based design builds the model through successive operations, but it is not limited to simple geometric primitives. Instead, feature based designs begin with 2-D sketches, which are then used as the basis for forming features, such as extrusions, revolves, lofts, and sweeps. As the model is developed, the CAD program assembles the features into B-rep geometrical representations.

Such models are inherently history based, which is to say the order in which features are defined and reference one another is essential to the model. All feature based models have a design tree which defines how the part was created, with parent features linked to children which reference them. This allows fully parametric design, in which features are defined by parameters including constraints and dimensions. Changes to a parameter within a parent feature (such as a thickness) cascade down to the children automatically, giving the designer greater flexibility. There are, however, downsides to using feature based designs.

Although powerful, feature based design tools have their limitations. They have a steep learning curve, with many different options for creating the same part, making interaction of models generated by multiple users difficult [116]. Additionally, if robust models are not created, changes to early features in the design tree can cause update failures if the references in the feature's children are broken. Despite these issues, the release of Pro/ENGINEER by PTC in the late 1980s, the first feature-based CAD program, revolutionized the industry. Today, nearly all CAD programs employ parametric feature based design.

There have been recent steps made toward a more user-friendly CAD methodology known as direct modeling, beginning with the release of SpaceClaim in 2007. Direct modeling tools allow the user to directly manipulate geometry in the graphical user interface. A feature, such as a rib, can be dragged using the mouse to change its position. A face can be pulled out or pushed in to the change the thickness of a part. Direct modeling is inherently non-historical, as opposed to classic feature based CAD. While feature based modeling remains the standard in most industries, direct modeling features have become popular due to their ease of use in modifying existing geometry, and many feature based tools have incorporated some direct modeling capabilities.

One of the greatest limitation of feature based CAD models is the lack of interoperability [117]. Although many modern CAD tools utilize parametric feature based design, they generate the geometry using different modeling kernels. Something as simple as a cylinder, created by extruding a 2-D sketch of a circle, can be defined differently in various CAD programs. A curve may be represented as a cubic spline, b-spline, Bèzier curve, or non-oriented rational b-spline (NURBS). Different programs may also have different orders of approximation, yielding slightly different geometry from the same input parameters. Some 3-D modeling kernels are commercially available and licensed to software makers, most notably ACIS and Parasolid, but some CAD programs use their own proprietary kernels. As a result, a CAD model generated using one program cannot, as a rule, be opened by another. A list of some of the most important CAD programs in use today can be found in Table 1.1.

Table 1.1 A selected list of modern, feature based CAD programs; note that Dassault and Siemens both have two offerings, noted as high-level and mid-level

Program	Developer	Modeling Kernel	Notes
Inventor	Autodesk	ShapeManager* (formerly ACIS)	
CATIA	Dassault Systèmes	CGM*	High-level
SolidWorks	Dassault Systèmes	Parasolid	Mid-level
Creo (formerly Pro/ENGINEER)	PTC	Granite*	
NX (formerly Unigraphics)	Siemens	Parasolid	High-level
Solid Edge	Siemens	Parasolid (formerly ACIS)	Mid-level

* Proprietary kernel

Several platform-agnostic file formats for 3-D geometry transfer exist, including STL (Stereolithography), IGES (International Graphics Exchange Specification), and STEP (Standard for the Exchange of Product model data) [112], however, they do not allow for true interoperability. As opposed to feature based CAD models, these file standards contain only geometry definitions. STL files consist of a collection of facets, originally intended for additive manufacturing. While useful for visualizing geometry, they do not capture curvature. IGES was a standard created by ANSI (American National Standards Institute) for sharing geometry data between CAD programs. However, the standard was loosely defined, leading to different interpretations by CAD software developers. The STEP standard created by ISO is regarded as a superior method for sharing 3-D model geometry [117].

1.8 Objectives

There are two primary objectives of this research. The first objective is to develop a method for creating detailed models of rotors for integrated 3-D analysis. Conventional rotorcraft analyses use 1-D beam-based structural models, wherein the description of the blade is a simple distribution of properties (mass, stiffness, etc.) along the radius. For 3-D structural analysis, a new type of rotor model is required, one which is based on the actual geometry of the rotor as defined in 3-D CAD models. This research develops a workflow for developing solid finite element meshes from rotor CAD geometry, assembling those meshes with multibody joints, and packaging those components along with their properties into input files capable of being read by X3D. It explores best practices for creating such 3-D structural analysis models, and also defines standards for X3D structural model data files and provides tools for creating these types of files. After the workflow has been developed, simplified notional rotor models are created using the new methodology, and compatibility with X3D for structural analysis is demonstrated.

The second objective of this work is to validate X3D against experimental measurements of a real, advanced rotor. A 3-D structural analysis model is created for the Tilt Rotor Aeroacoustic Model (TRAM), ¼ scale model of the V-22 proprotor tested by NASA in the late 1990s [118]. X3D is used to study the aeromechanics of the proprotor, and predictions of performance, airloads, and structural blade loads are compared to experimental measurements to gain fundamental insight into the physical mechanisms of loading. Results using both low-order aerodynamics models and RANS, achieved by coupling X3D with Helios, are examined.

1.9 Contributions

There are two sets of key contributions from this work related to the state-of-the-art based on the two objectives set forth above: first, those relating to the integrated 3-D model development methodology, and second, those relating to the analysis of the TRAM proprotor in edgewise flight.

The first set of contributions (modeling methodology) are:

1. Developing a methodology for creating 3-D rotor structural analysis models of generic rotor geometry for integrated 3-D aeromechanics analysis
2. Establishing definitions and standards for structural analysis model inputs
3. Developing tools necessary for creating model inputs, including processing part meshes, defining joints, establishing aerodynamic interfaces, and assembling all parts into a multibody structural analysis model
4. Creating new techniques for performing 3-D rotor blade structural analysis, including a method of extracting sectional structural properties for reduced order modeling, and method for calculating equivalent sectional blade loads from 3-D strain predictions for comparison to experimental data.

The second set of contributions (TRAM analysis) are:

5. Developing an accurate 3-D model of the TRAM proprotor, demonstrating the suitability of the modeling methodology for real rotors with complex internal structures and hub structures
6. Performing the first integrated 3-D analysis of a real rotor; also the first coupled 3-D CFD/CSD analysis of a proprotor
7. Demonstrating the ability to simulate rotor kinematics and structural dynamics from first principles, not relying on empirical measurements
8. Comparing airloads predictions for the TRAM rotor using multiple aerodynamics models to experimental data, from simple to progressively more refined models to develop a fundamental understanding of loading mechanisms

9. Comparing blade loads predictions for the TRAM rotor to experimental data, similarly progressing from simple to more refined models
10. Examining the effects of modeling the gimbal at the rotor hub, comparing a simple approximation to a more precise model, revealing that while the precise gimbal model does affect vibratory (3/rev) airloads and blade loads, it has a less significant impact when compared to improvements from precise aerodynamic modeling

Chapter 2: Modeling and Analysis Methodology

X3D, the experimental 3-D rotor dynamic analysis, is an advanced, three-dimensional structural dynamics solver designed for integrated aeromechanics analysis of rotors developed by the US Army AFDD [8]. It utilizes a multibody analysis framework in which flexible components are analyzed using full 3-D FEM, allowing it to overcome the limitations of 1-D beam-based analysis. Kinematic couplings are simulated by multibody analysis and 3-D stresses and strains are recovered from the FEM. This analysis requires an entirely new kind of rotor aeroelastic model compared to current generation beam-based analysis. This chapter describes the requirements of these new 3-D structural analysis models, and describes in detail the workflow and methodology for creating this new kind of model. Finally, it provides some examples of 3-D models generated using this workflow.

2.1 Structural Model Features

With legacy beam-based aeromechanics analysis, rotor structural models are relatively simple. Models consist of arrays of nodal coordinates and sectional properties for each beam element, such as mass and inertia, bending stiffnesses (EI) and torsional stiffnesses (GJ), beam axis geometry (typically the 1/4 chord line), and offsets (center of gravity and elastic axis). In contrast, the nature of a full 3-D analysis requires a new kind of structural analysis model (SAM) description with several unique features.

1. These models require 3-D geometry generated using computer aided design (CAD) tools. This geometry must include all structurally significant features, including internal construction. Developing good CAD models is essential, but discussing the principles of CAD modeling in general is not the purpose of this dissertation. Only the unique feature important to this work are discussed. Many contemporary rotors are already being designed and manufactured using CAD tools, so it is expected that 3-D geometry models will already be available for most future applications.
2. The models must take advantage of both the finite element analysis and multibody dynamics aspects of the solver. This means the model developer needs to plan ahead for which rotor components (e.g., blade or yoke) will be analyzed as flexible parts and which components (e.g., bolts or lag dampers) will be treated as kinematic joints or devices.
3. The flexible parts must be meshed with full second-order 27 node hexahedral elements (Hex27), as illustrated in Figure 2.1. The Hex27 was chosen as the base element for X3D because they prevent locking, a numerical phenomenon present in many FEM applications in which accuracy no longer improves with increasing number of elements if a certain dimension (in this case, thickness) shrinks to a limiting value [119]. With 27-node hexes, thin layers can safely be modeled as solids, allowing a single element type to treat all components of a rotor, leading to greater flexibility for the model designer.

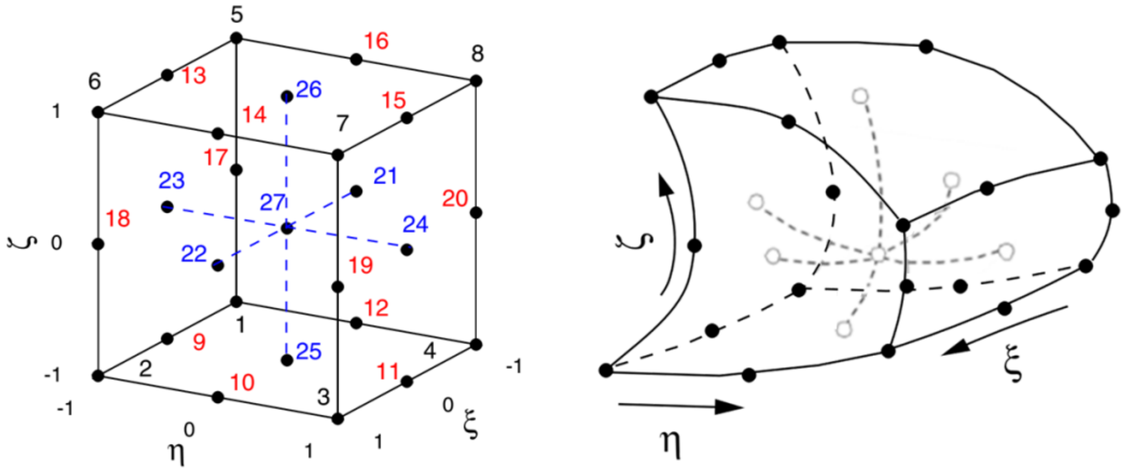


Figure 2.1 (Left) An isoparametric 27 node hexahedral element (Hex27) in natural coordinates; (Right) Hex27 element in physical coordinates

4. Discretization of flexible parts into finite element meshes must be carried out in a manner compatible with the solver. Flexible parts are meshed independently, but must fit into the multibody analysis framework of the rotor. This means provisions must be made during flexible part meshing to properly position nodes for creating future joint attachment points, and these nodes must be identified for input into the solver.

2.2 The Structural Modeling Workflow

This section describes the methodology for developing a 3-D rotor structural analysis model (SAM). Establishing this workflow, and all tools necessary to carry it out for generic rotor geometries, is the first objective of this research. There are five steps in the model development procedure (Figure 2.2):

1. Create or obtain detailed 3-D CAD geometry of the rotor

2. Create a structural analysis representation (SAR) of the rotor
3. Mesh the rotor components that will be treated as flexible parts
4. Process meshes to reduce bandwidth, create joints to connect parts, and define material properties within each flexible part
5. Assemble the final structural analysis model, including meshes, joints, material properties, and aerodynamic interface definitions

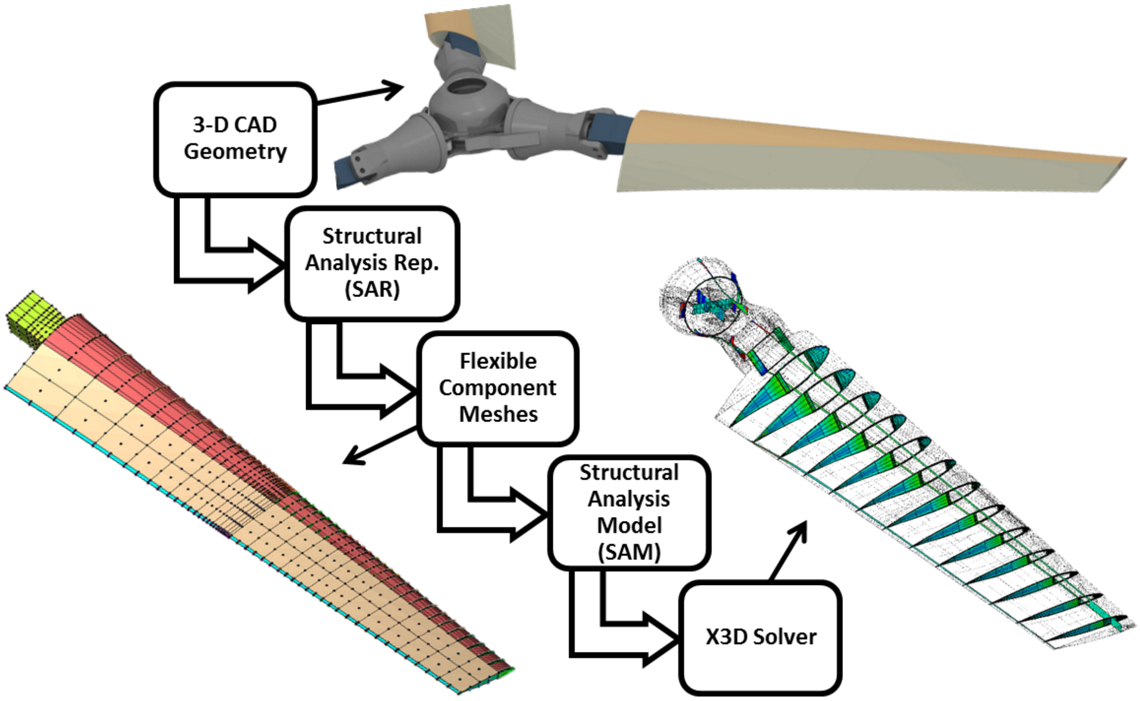

3-D CAD Geometry
Structural Analysis Rep. (SAR)
Flexible Component Meshes
Structural Analysis Model (SAM)
X3D Solver

Figure 2.2 Workflow for generating 3-D structural analysis models for use with X3D

These steps vary in complexity depending on the type of rotor and the information available to the model developer. For example, starting with a 3-D laser scanned CAD model of an existing single-piece (with no internal structure) rotor blade eliminates the first step entirely, whereas creating new CAD models from engineering drawings of a legacy rotor (as was necessary for the TRAM rotor in this work, Chapter 4) will make the first step very critical. For most rotors, developing the CAD model and meshing the flexible components can take a substantial amount of time and effort.

2.3 CAD Geometry

The first step in creating a 3-D structural analysis model (SAM) is to create or obtain detailed CAD geometry of the rotor. This geometry may be created in any CAD software package or through other means, such as a 3-D scanner. For meshing, however, the CAD geometry must be provided in a platform-agnostic geometric file format, such as STEP, IGES, or STL (this is in contrast to feature based file types, such as .CATPart and .SldPrt, which are proprietary file types associated with CATIA and SolidWorks respectively). All of the models presented in this dissertation were created in CATIA V5 and exported in the STEP file format for meshing.

Ideally, CAD geometry used for creating 3-D structural analysis models should be developed with the requirements of meshing in mind. Being familiar with meshing volumes into hexahedral finite elements assists in developing the CAD geometry for the blade and other rotor components. While modifications can be made to the geometry within the meshing software, starting with good CAD geometry greatly simplifies the meshing procedure. When preparing the CAD model, the designer should make educated choices as to what features are structurally significant, which is to say parts that admit large strains. Insignificant features, such as cosmetic fillets, for example, can be removed to ease meshing. Minor modifications can be made to the geometry using the meshing tool, but the model developer and analyst can save time if the CAD geometry does not have those structurally insignificant features in the first place.

2.3.1 Parameterized Models

Most modern CAD programs allow for the creation of parameterized feature based models. Parameterization allows the model designer to control the part's features using variable parameters. For example, the radius of a rotor, the blade chord, and other dimensions can be driven by parameters either entered manually into the program or linked to an external digital spreadsheet (e.g., an .xls file). The various dimensions of the part are then related to the base parameters via formulae. Using parameters allows the part to be quickly modified, allowing a single base CAD model to create several different examples of a part. Figure 2.3 shows an example of the same blade cross-section on a single part with two different sets of parameters driving the geometry.

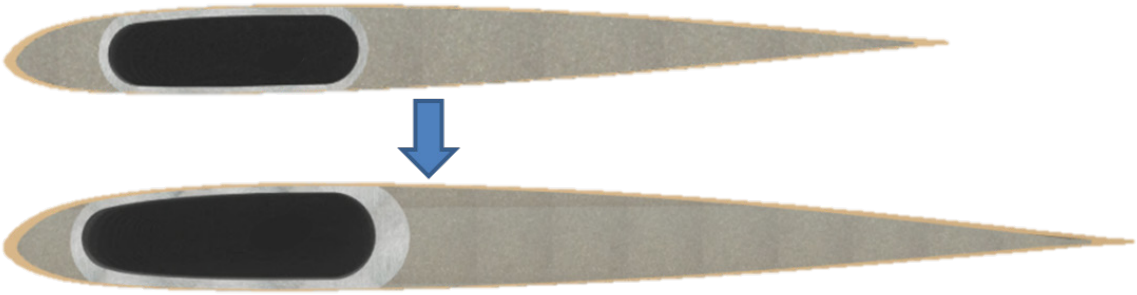


Figure 2.3 Demonstration of CAD parameterization being used to modify blade chord, skin thickness, spar position relative to the leading edge, and spar wall and cap thickness

Parameterization can greatly reduce the time it takes to test families of parts with small variations. Meshing flexible parts can be a time consuming process, but if a model is parameterized, the command scripts which create the mesh only need to be generated once. For the blade seen in Figure 2.3, the same meshing commands can be applied to both versions of the blade; two different meshes are created with minimal additional effort.

2.3.2 Assemblies, Parts, and Bodies

The full rotor CAD model should be an assembly of individual rotor parts. All significant components should be included, even if they are treated as kinematic joints and not meshed for finite element analysis (FEA). This will allow information about the positions and orientations of components to be extracted from the 3-D CAD model later in the process.

Internal to each CAD part, the designer should make use of multiple bodies. The terminology of "bodies" is specific to CATIA; however, many high level CAD applications have similar capabilities. Each body should represent internal components made out of a single material (or composite layup). For example, if a blade has a spar with an aramid honeycomb aft core, a foam leading edge core, and a carbon fiber composite skin, each of these pieces should be a separate body within a single CAD part file. The entire part will be meshed at one time, but the meshing software will maintain the bodies as discrete volumes (discussed further in Section 2.5: Meshing), ensuring that the mesh accurately represents the internal geometry of the blade.

A CAD model of a notional bearingless rotor is provided in Figure 2.4 as an example. The rotor blade assembly contains several parts, including a blade, a torque tube, a flexure, a damper, and bolts. Taking a cross-section of one of the parts, the blade, reveals that it is constructed using several bodies, including the skin, fore and aft foam cores, and a spar. The spar is actually made up of multiple bodies, although all are rendered the same in Figure 2.4, giving the option of later applying different material properties to each piece of the spar.

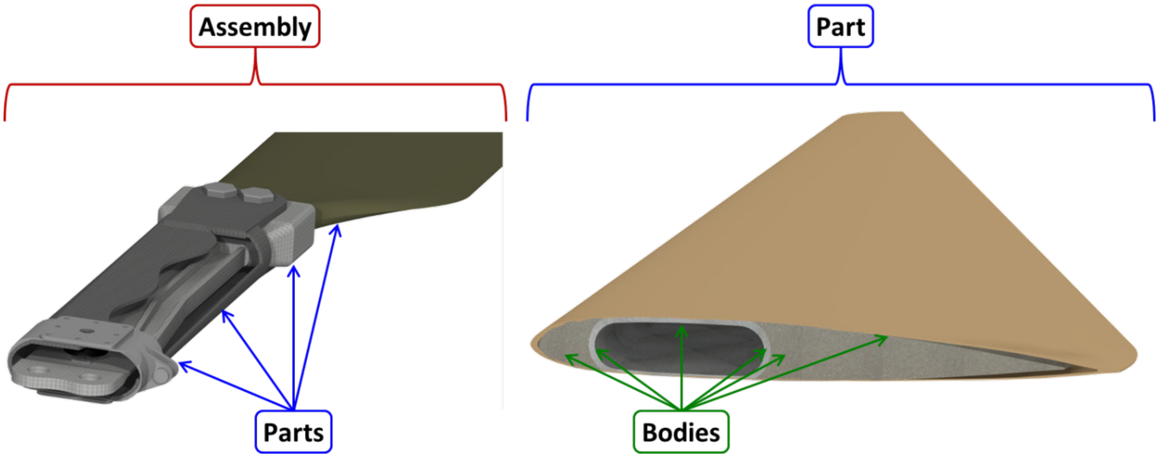


Figure 2.4 Sample CAD models of a bearingless rotor and blade, showing the difference between assemblies, parts, and bodies

2.4 Structural Analysis Representation

Once a CAD assembly of the system has been completed, the next step is to establish a structural analysis representation (SAR) of the model. The SAR is not a discrete digital product, such as a CAD part file or mesh file, but a conceptual plan laying out the topology of the rotor and how to treat each of its components. At this stage, the model developer decides which parts to model as flexible structures using FEA and which ones to treat with simpler representations in the multibody framework. Each part must be designated as belonging to one of three categories: *flex parts*, *joints*, or *devices*. Once the parts of the structure have been identified and classified, their positions and orientations in the rotor reference frame must be determined and the relationships between all of the parts must be defined.

2.4.1 Parts in the Structural Analysis Representation

2.4.1.1 Flex Parts

Flex parts are components which are expected to undergo elastic deformation, and are modeled using three-dimensional finite elements. These parts must be imported as CAD geometry into a mesh preprocessor (such as Cubit, Section 2.5.2) and discretized into Hex27 finite elements for the final structural analysis model.

In general, each flex part has its own coordinate system, defined by the designer during the creation of the CAD geometry. Position and orientation vectors must be defined to transform the individual reference frames of each part into the global reference coordinate frame, ensuring proper arrangement of all parts. Position is defined by a vector [u_1, u_2, u_3], each component of which corresponds to a translation in the three primary axes of the global coordinate frame. Orientation is defined by a three-value vector containing Euler rotation angles. The order of rotations must be specified so the solver applies the rotations correctly. Alternatively, orientation may be described using a three-by-three rotation transformation matrix.

2.4.1.2 Joint Parts

Joint parts are simplified components that serve to connect other parts together in the multibody framework. In addition to attaching parts to one another, joints may also be used to define a boundary, connecting a part to a position in the global reference frame, rather than to another part. The structural analysis representation defines which parts these kinematic joints connect, establishing the topology of the multibody system.

The joint degrees of freedom are established during the definition of the structural analysis model (Section 2.6.2: Joint Definitions). When the joint degrees of freedom are defined, they may be assigned linear stiffness and damping values. Furthermore, joints may be commanded or forced, allowing the solver to actuate or force motion of the system. For example, static deflection of a beam with a tip load can be simulated by attaching a tip joint to the end of the beam mesh; this tip joint can then be assigned a forcing function with a load in the Z-direction. Similarly, rotor trim angles can be prescribed by commanding collective and cyclic degrees of freedom on a pitch bearing and identifying its rotation as a trim variable within the solution input deck.

As with flex parts, joints must have their (undeformed) position and orientation defined. The positions and orientations of the joint can be extracted from the 3-D CAD model assembly. Depending on the type of joint, the position may be more or less important; a weld, for example, will fix components regardless of the position of the joint. Rotation about a hinge, in contrast, can be skewed by orienting the joint at an angle.

2.4.1.3 Device Parts

Device parts are those that do not fit into the other two categories, and can exert special interactions or forcing on other parts. The implementation of devices allows the integration of externally determined dynamic characteristics via a look-up table. A typical example of a device part found in a rotor is a lag damper.

Like other parts, devices must be defined with a position and orientation, and are connected to joints. Specific details of device operation, such as damping properties, are defined in the solver input. The use of devices in X3D is beyond the scope of this research, and limited discussion of their application is found in this dissertation.

2.4.1.4 Part IDs and Ordering

Each part is given two identification numbers (IDs), one global and one specific to the part type. A unique global ID is assigned to each part, starting from 1 and indexing up to the total number of parts. Part specific IDs are assigned based on the type of part and include a letter: F for flex, J for joint, and D for device. In a model with three flex parts, three joints, and one device, the global SAR IDs will be 1, 2, 3, …, 7. The flex parts will have IDs F1 to F3, the joints J1 to J3, and the sole device will be labeled D1.

The ordering of part IDs at this stage can affect the solution time. When run on a single processor, X3D utilizes a skyline solution procedure. The skyline method is most efficient when the height of the skyline, which corresponds to elements lying far from the diagonal of the problem stiffness matrix, is kept small. Parts are stored in the solution matrix based on the order of their global IDs. Practically, this means that the solution will be more efficient if parts are organized in the order in which they are connected. For example, when modeling a rotor one can work from the outside-in, Part 1 being the blade, Part 2 the joint which connects the blade to a grip, Part 3 the grip, and so on until the hub. Further discussion of the effect of part ordering on skyline size, memory usage, and solution time is provided in Chapter 4.

2.4.2 Structural Analysis Representation (SAR) Examples

The information provided by the structural analysis representation includes a list of parts in the rotor, identification of part types, positions and orientations of parts relative to the global rotor coordinate frame, and a list of part connections. This information can be presented in any format, but a tabulated list combined with a load flow diagram is recommended. Below are simple examples of structural analysis representations of X3D models. Chapter 4 discusses the SAR for the TRAM proprotor model.

2.4.2.1 SAR Example: Three Blocks

First, a simple example of a non-rotor structure is introduced to illustrate the basic principles of developing a 3-D structural analysis model. Consider three blocks, arranged in an L shape, as shown in Figure 2.5. The first block (green) is a 1 inch cube placed with its origin at the global origin of the assembly. The second block (blue) is rectangular, 3 inches long, and is joined to the first block at its root face, but allowed to rotate about its longitudinal axis. Although it is not shown in the CAD model, consider the case in which there is a rotary damper acting between blocks one and two. The third block (yellow) is 2 inches long and is attached to the first block rigidly on one face.

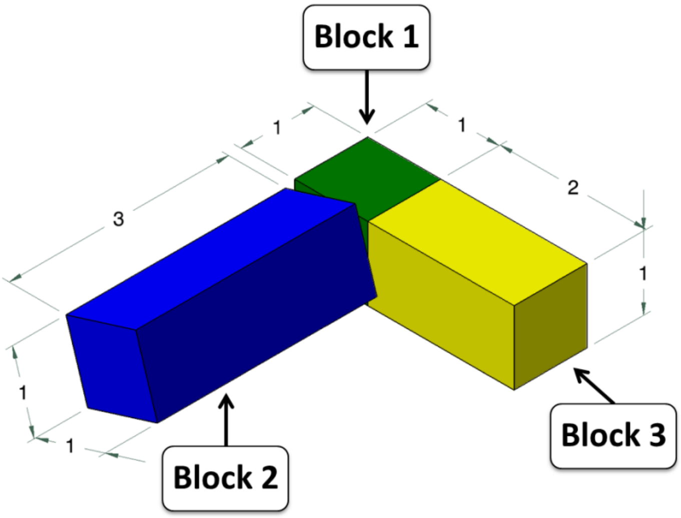


Figure 2.5 CAD model for the simple three block assembly example

In this simple example, there are seven total parts: three flex parts, three joints, and one device, as labeled in Figure 2.6. The three flex parts, F1, F2, and F3, are the three blocks. The first joint, J1, is a boundary connecting F1 to the global reference frame. The second joint, J2, connects F1 to F2 as well as the rotary damper (device D1). This must be a revolute joint to allow block F2 (blue) to rotate about its axis. The last joint J3 fixes flex part F3 to F1 with their adjacent faces.

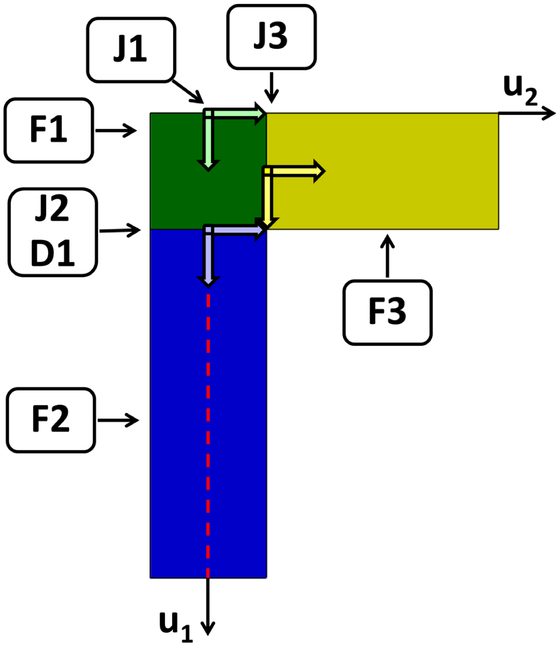


Figure 2.6 Structural analysis representation schematic for a three block example, top view, with flex parts (F), joints (J), and devices (D) labeled; flex part reference frame axes are provided for each block and color coded; global part axes u_1 and u_2 are labeled; part F2 is free to rotate about its longitudinal axis, marked by a red dashed line

Table 2.1 summarizes the structural analysis representation for the three block example. Because ordering the parts to reduce bandwidth is not important for this simple example, the global SAR part IDs are grouped by part type. Figure 2.6 shows that the global reference frame coordinate system of the assembly (u_1, u_2) is aligned to the local coordinate system of the first block, flex part F1, thus its position is given as [`0`, `0`, `0`] in

Table 2.1. The other two blocks (F2 and F3) are each translated by the vector given in the position column of the table. In this case, the coordinate systems for all three blocks are aligned parallel to the reference frame, so all orientation angles are zero. Similarly, each of the joints and the device has its own position and orientation vector. The final column in the table lists all parts which connect to the part described in the row, in terms of the global part ID. For example, the first block (F1) connects to all three joints (J1, J2, J3) which have the part IDs 4, 5, and 6. Note that part 4 is the boundary linking block 1 to the reference coordinate system; this is signified by `-1` in the connections column. In this example the problem is small, so ordering of parts to maintain a small skyline is not important, so parts are grouped by type for ease of reference. If a more efficient solution was desired, the parts might be provided in an alternative order, as shown in Table 2.2, following along the L shape to minimize the distance between connections.

Table 2.1 Structural analysis representation of the three block example, part ordering grouped by type

Part ID	Type ID	Name	Position (in.)	Orientation	Connections
1	F1	Block 1	`[  0,   0,   0]`	`[0, 0, 0]`	`4, 5, 6`
2	F2	Block 2	`[  1,   0,   0]`	`[0, 0, 0]`	`5`
3	F3	Block 3	`[0.5, 0.5,   0]`	`[0, 0, 0]`	`6`
4	J1	Boundary F1	`[  0,   0,   0]`	`[0, 0, 0]`	`-1, 1`
5	J2	Revolute F1/F2/D1	`[  1,   0,   0]`	`[0, 0, 0]`	`1, 2, 7`
6	J3	Fixed F1/F3	`[0.5, 0.5,   0]`	`[0, 0, 0]`	`1, 3`
7	D1	Damper	`[  1,   0,   0]`	`[0, 0, 0]`	`5`

Table 2.2 Structural analysis representation of the three block example, part ordering for lower skyline and greater efficiency

Part ID	Type ID	Name	Position (in.)	Orientation	Connections
1	F3	Block 3	[0.5, 0.5, 0]	[0, 0, 0]	2
2	J3	Fixed F1/F3	[0.5, 0.5, 0]	[0, 0, 0]	1, 3
3	F1	Block 1	[0, 0, 0]	[0, 0, 0]	2, 4, 5
4	J1	Boundary F1	[0, 0, 0]	[0, 0, 0]	-1, 3
5	J2	Revolute F1/F2/D1	[1, 0, 0]	[0, 0, 0]	3, 6, 7
6	F2	Block 2	[1, 0, 0]	[0, 0, 0]	5
7	D1	Damper	[1, 0, 0]	[0, 0, 0]	5

2.4.2.2 SAR Example: Bearingless Rotor

The next example is more realistic; it is a notional bearingless rotor model shown in Figure 2.7. The rotor blade is connected to the hub via a flexure and a torque tube, with a snubber damper connecting the inboard end of the torque tube to the flexure. As shown in the structural analysis representation schematic in Figure 2.8, there are eleven total parts: four flex parts, six joints, and one device. The blade, flexure, torque tube, and pitch link (not shown in the CAD drawings) are treated as flex parts. The blade bolts, the lag damper connections to the flexbeam and torque tube, the hub bolts at the root of the flexbeam, and the top and bottom of the pitch link are all treated as joints. Note that the joint at the bottom of the pitch link can be given a vertical degree of freedom with commanded motion to model the swashplate inputs. Finally, the lag damper is modeled as a device part – a joint with an assigned linear damping.

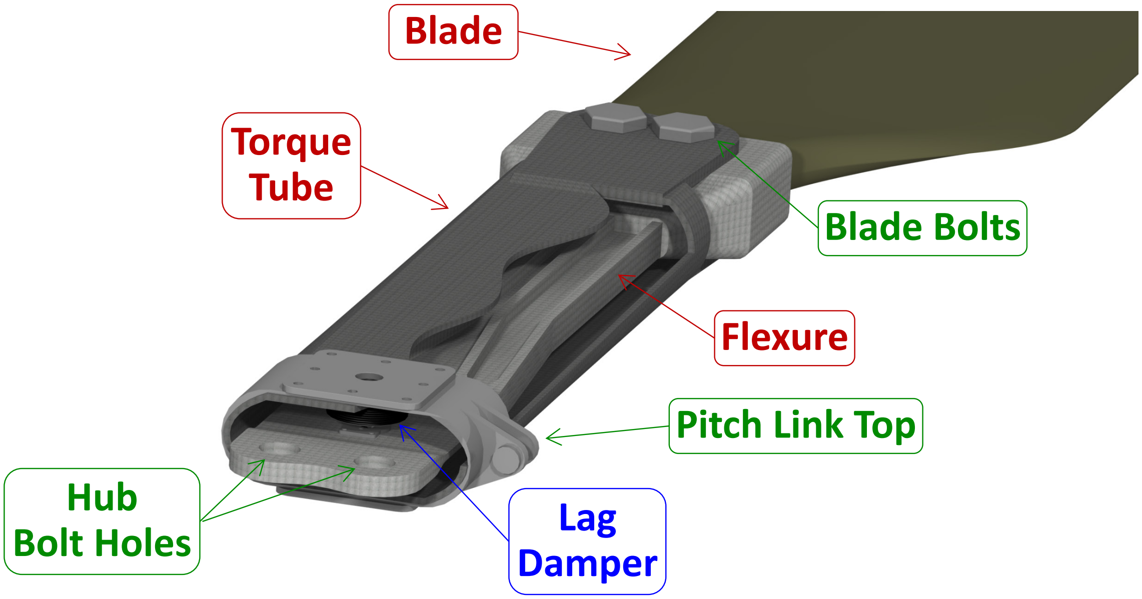


Figure 2.7 CAD model of a notional bearingless rotor blade root (pitch link omitted)

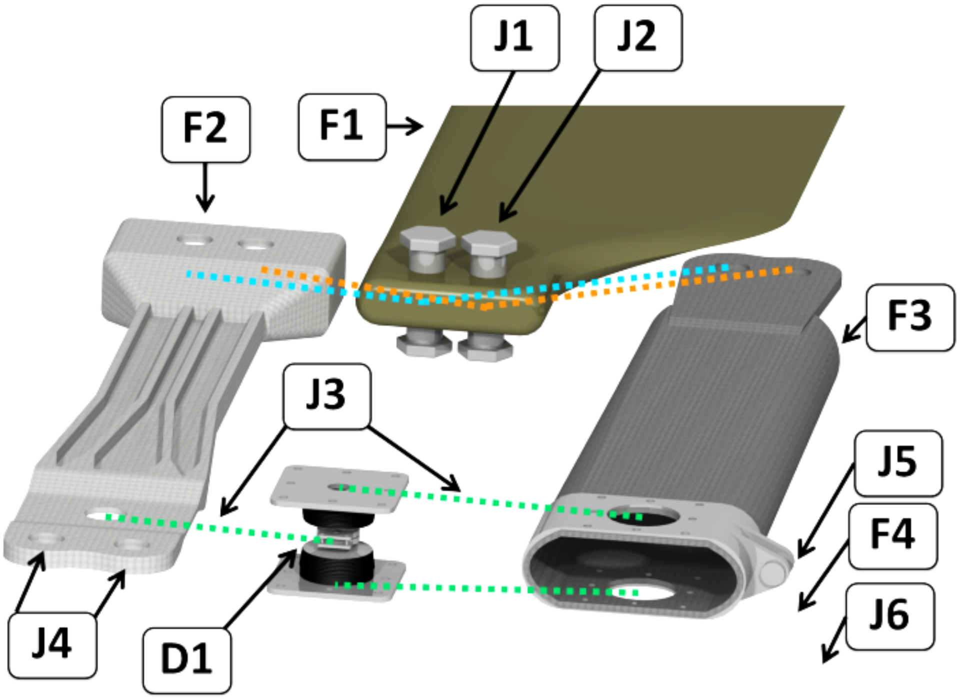


Figure 2.8 Structural analysis representation schematic for the notional bearingless rotor example, with flex parts (F), joints (J), and devices (D) labeled; pitch link (F4) CAD omitted

To maintain skyline solution efficiency, the bearingless rotor has parts ordered outboard to inboard with controls last, as shown in Table 2.3. Each of the flex parts has an orientation rotated 180° about the vertical u_3 axis. This transformation is necessary because of how the CAD was created. The geometry model for each part was drawn assuming the blade was in a zero-azimuth position in a vehicle-fixed coordinate frame, with the u_1 axis pointing toward the front of the rotorcraft, and thus away from the blade tip. The structural analysis model, on the other hand, is designed with the radial u_1 coordinate positive toward the blade tip.

Table 2.3 Structural analysis representation of the notional bearingless rotor example; LE: Leading Edge, TE: Trailing Edge

Part ID	Type ID	Name	Position (in.)	Orientation	Connections
1	F1	Blade	[0, 0, 0]	[0, 0, 180°]	2, 3
2	J1	LE Blade Bolt	[26, 0.625, 0]	[0, 0, 0]	1, 4, 5
3	J2	TE Blade Bolt	[26, -0.625, 0]	[0, 0, 0]	1, 4, 5
4	F2	Flexure	[12, 0, 0]	[0, 0, 180°]	2, 3, 6, 8
5	F3	Torque Tube	[19, 0, 0]	[0, 0, 180°]	2, 3, 6, 9
6	J3	Damper Joint	[11, 0, 0]	[0, 0, 0]	4, 5, 7
7	D1	Lag Damper	[11, 0, 0]	[0, 0, 0]	6
8	J4	Hub Bolts	[13, 0, 0]	[0, 0, 0]	-1, 4
9	J5	Pitch Link Top	[14, 2, 0]	[0, 0, 0]	5, 10
10	F4	Pitch Link	[12, 0, 0]	[0, 0, 180°]	9, 11
11	J6	Pitch Link Bottom	[14, 2, -10]	[0, 0, 0]	-1, 10

2.5 Meshing

Once it has been decided which rotor components will be modeled as flexible components, they must be discretized into finite elements. This section describes the process required for creating meshes of 3-D rotor components from CAD geometry. It discusses basic features of FEM meshes that are necessary for compatibility with X3D. A detailed description of 3-D hexahedral finite element meshing is beyond the scope of this dissertation, but specific steps for meshing helicopter rotor blades are provided: a breakdown of the meshing process is presented from importation of models through export of the final mesh, as well as some specific best practices for dealing with meshing difficulties encountered over the course of this research.

2.5.1 Meshing Software Requirements

This research used the Cubit mesh preprocessor software for meshing, but other preprocessors may be capable of generating compatible meshes. As discussed above (Section 2.5.1: Meshing Software Requirements), the X3D solver currently only supports 3-D solid 27 node hexahedral elements (Hex27, Figure 2.1). Thus, a mesh preprocessor capable of creating Hex27 meshes is necessary.

Integrating solid meshes with X3D requires specific elements, faces, and nodes to be identified in a certain way. This is done in Cubit using features known as blocks, sidesets, and nodesets, discussed in greater detail in Section 2.5.3.5. Similar structures for identifying nodes in mesh files must be output from any mesh preprocessor used.

Finally, the meshes created under this work have all been exported using the I-DEAS Universal .unv file format. This is an ASCII file format, selected because it supports Hex27 elements; identifies blocks, nodesets, and sidesets; and it is defined using a well-documented standard [120]. Tools were developed as part of this research to process I-DEAS Universal .unv files into X3D compatible .dat mesh data files. As such, it is recommended that any meshing software used has the ability to export to the I-DEAS Universal .unv file format, otherwise a new mesh processor will need to be developed.

2.5.2 Cubit

Cubit is a mesh preprocessor developed by Sandia National Laboratories [121]. Cubit was developed with solid hexahedral meshing as a primary goal. While other meshing tools can handle hex meshing, Cubit is adept at it. It can also support meshing with the Hex27 elements required for X3D. Finally, because it was developed by Sandia, licenses are freely available for use by the US government and those performing research under US government contracts. For these reasons it was selected for this research, and Cubit version 14.0 was used to generate all meshes described in this dissertation. Cubit is relatively easy to use, with a robust graphical user interface, thorough documentation, and technical support.

2.5.2.1 Meshing Algorithms in Cubit

The primary algorithms used in CUBIT for finite element discretization are mapping, paving, and sweeping [121], and most components found in a rotor can be meshed using a combination of these tools.

The mapping tool is used to discretize either a surface into quadrilateral elements (quads) or a volume into hexahedral elements by means of establishing a Cartesian grid. For a surface, the mapping algorithm treats the surface as a logical rectangle, identifying four logical corners and four logical sides (Surface A in Figure 2.9, source/target surfaces in left half of Figure 2.10; both figures from [121]). Note that these sides can be made up of multiple curves, as long as the vertices between these curves are not marked as corners. Once a logical rectangle is established, opposite sides have nodes placed at equal intervals and connected in a rectilinear grid. This process extends to volumes, where the entire volume is treated as a logical cuboid, with eight logical corners and six logical faces, as shown in the examples in Figure 2.10. Volumes do not have to be perfectly cuboid in shape. For example, the volume on the left side of Figure 2.11 can be meshed by mapping despite being curved.

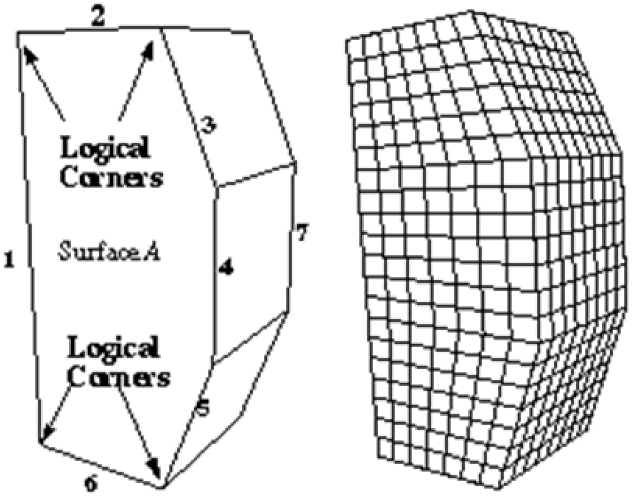


Figure 2.9 Mapping a volume; for Surface A, edges 3, 4, and 5 are treated as one side of a logical rectangle

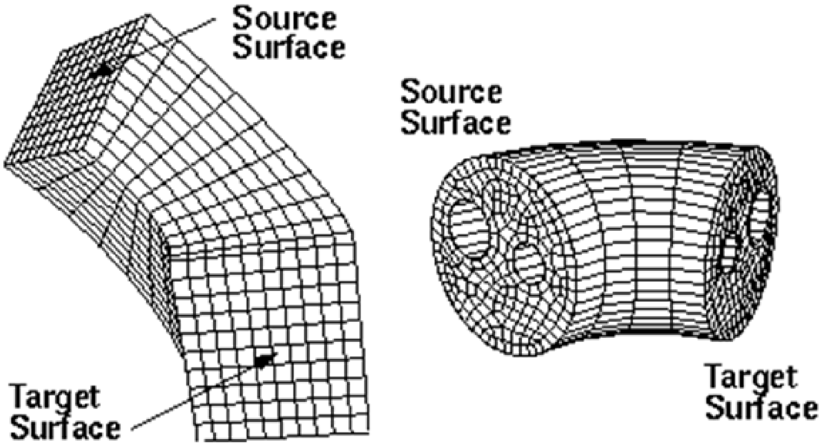


Figure 2.10 Sweeping volumes: (Left) source and target surface meshed using map; (Right) source and target surface meshed using pave

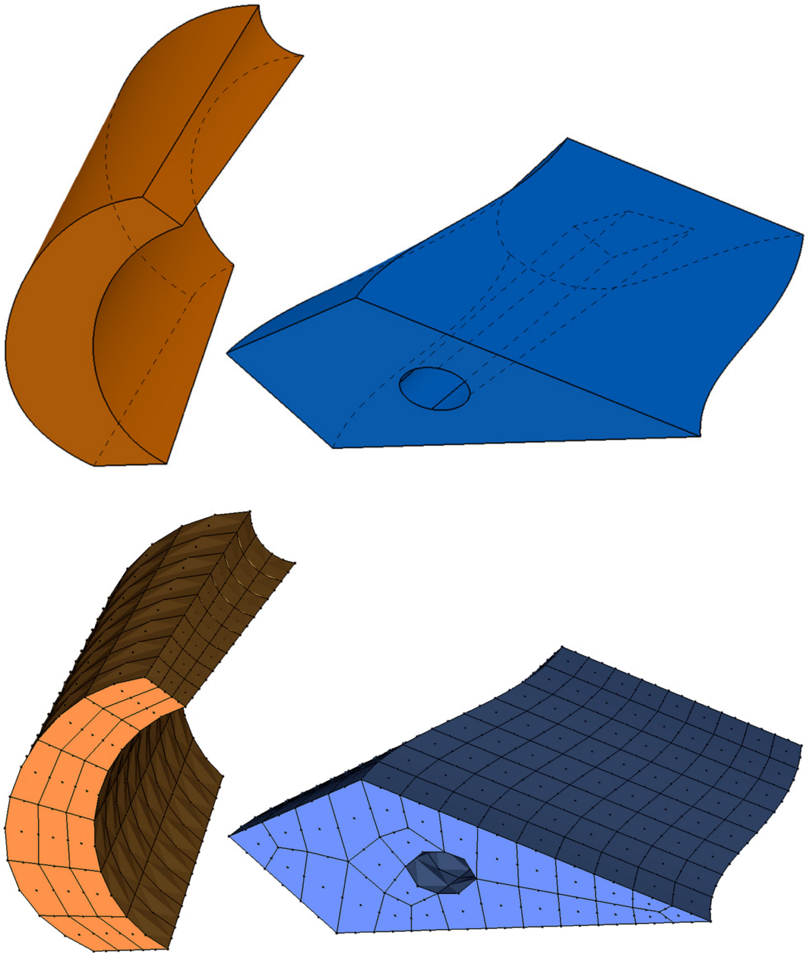


Figure 2.11 Examples of volumes that can be meshed by mapping (left) and sweeping (both); if there were no hole in the center, the volume on the right could also be mapped

Paving is a method by which surfaces of an arbitrary shape are discretized with an unstructured quadrilateral mesh (source surface on right side of Figure 2.10) [122]. It begins by identifying the boundaries of the surface, including any interior holes, and meshing the curve loops that form the boundaries, thus forming a seed mesh from which the surface mesh is grown. The algorithm then chooses a starting point where an initial quadrilateral surface element can be placed (e.g., a corner where three nodes provide an obvious choice for a fourth to form a quad). Once the new quad is placed, the algorithm proceeds around the boundaries, at each step checking new nodes for proximity to existing nodes for potential mergers, ensuring elements do not intersect, and smoothing to ensure high quality quads are created. This algorithm provides a very robust tool for meshing a wide variety of surfaces, including those that are thin or include sharp corners [123]. Other useful surface meshing schemes in Cubit include circle and hole, which mesh circular surfaces and annular surfaces respectively.

Sweeping is one of the most important volume meshing tools available. It works on “2.5-D” bodies, which are volumes with two opposite faces (a source and a target) with similar topology that are connected by linking surfaces, as illustrated in Figure 2.10 and Figure 2.11. The algorithm starts by meshing the source surface (e.g., using the mapping or paving algorithm) and then “sweeps” the meshed source surface to the target surface by mapping the linking surfaces, effectively extruding the initial mesh along an axis [124, 125]. The shape of the cross-section can change as it is swept from the source surface to the target, as long as Cubit can identify similar mesh features on both ends. It should be noted that the source face can actually consist of multiple surfaces, even on different planes, giving the sweep tool additional flexibility. Combined with the versatility of the paving algorithm, most volumes encountered in rotor modeling that cannot be meshed by mapping can be meshed via sweeping. Further, any volume that can be meshed by mapping can also be meshed by sweeping, though mapping is less computationally intensive and the default operation in Cubit when possible.

Another tool recently added to Cubit is known as Sculpt [121]. Sculpt is a tool capable of generating meshes for arbitrary 3-D volumes easily. It works by creating a fine 3-D mesh through the entire part, then clips elements which intersect the surfaces of the volume to try and match the geometry of the volume, using a method similar to overset grids in CFD. Although it is good for meshing difficult shapes with less user input, it requires very fine meshes, which are not suitable for rotor dynamic structural analysis. Since most volumes likely to occur in a rotor can be meshed with mapping and sweeping, the use of Sculpt was not necessary for this work.

Although these and several other automated tools exist in Cubit and other off-the-shelf preprocessors, input from the model developer is still required to yield a functional mesh. The main task in developing a good mesh is decomposing the geometry in such a way that these tools can be applied to different regions of the part. Care must also be taken to ensure the mesh includes nodes at certain prescribed locations, based on the CAD geometry, for connecting to joints. Once the mesh for a part has been completed, it is easy to mesh new variations on the same geometry, as long as the original CAD models are modified using parameters included during model generation (Section 2.3.1: Parameterized Models). More substantial modifications to the original geometry will require more effort to re-mesh.

2.5.2.2 Mesh Quality

Several metrics exist to determine the quality of eight noded hexahedral element (Hex8) meshes. Although these metrics are not tailored to the Hex27, they can provide some insight to find poorly formed elements which can cause errors or disrupt the convergence of iterative solvers. Knupp proposes algebraic quality metrics for hexahedra, which range from 0 for a degenerate element to 1 for an ideal element (a perfect cube) to assess the relative size, shape, and skew of mesh elements [126]. These metrics are based on a three-by-three matrix referred to as the element Jacobian, which contains information regarding the shape and size of the element. Calculation of these metrics are built into Cubit, and guidelines exist to ensure a minimum level of quality is maintained [121] for Hex8 meshes, though it must be noted that even if all quality metrics are met there is no guarantee that the mesh is perfectly suited for dynamic solution. Conversely, not meeting the quality benchmarks determined for Hex8 elements may not necessarily lead to an inaccurate dynamic solution with Hex27s. The use of Hex27 elements can allow accurate solutions with element quality values below typical published guidelines for Hex8 elements. Developing good quality metrics and benchmarks for Hex27 meshes is an area that is in need of further study.

2.5.2.3 Mesh Size

Currently, a means of partitioning structural domains within X3D for a general mesh has yet to be developed, limiting parallelization. Although domain decomposition within X3D has been successfully demonstrated on simpler meshes [75], further development is required before generic meshes of advanced rotors can be partitioned. As such, solution time on a single processor is still a limiting factor and mesh size must be taken into account when modeling the rotor.

Although general guide lines for mesh size have not been developed, it is recommended that overall mesh size for the entire system should be roughly below 20,000 nodes. At this size, it has been found that wind tunnel trim solutions with low order aerodynamics take on the order of two hours using a single Intel Core i5-2400 desktop processor.

Mesh size is driven by the depth of material modeling desired. Composite materials, made up of many plies laminated together, will greatly increase the number of degrees of freedom in the model if each ply is meshed individually. In order to maintain reasonably sized meshes, a material homogenization method has been developed. Homogenization allows a single element to be used across several plies of material. The properties of each ply within a given element are homogenized to create a single effective set of material properties, which can be applied to all elements within the block. This technique is discussed further in Section 2.6.4: Material Definitions.

When meshing surfaces or volumes, mesh size can be set by commanding Cubit to create elements with a target size in units of length, or be setting intervals along curves. The latter method is recommended as it enables fine control of the exact number of nodes around the perimeter of a surface, which can be important when paving and sweeping certain complex geometries.

2.5.2.4 Units in Cubit

Cubit is a unitless program; when CAD models are imported into Cubit it presents the data in whatever units were used when the original geometry file was created. CATIA defaults to exporting the STEP geometry files used in this research in units of millimeters. As such, all of the models discussed in this dissertation were meshed in millimeters, even when the native units in the original parametric CAD file were imperial. Although Cubit does not support tracking of units, it is essential that consistent units are employed, because X3D operates using SI units, including the meter as the standard unit for length. Because the .dat mesh data files used by X3D are also unitless, a scaling factor to meters must be specified for each mesh in the `SAM.input` X3D input file (Section 2.6.5).

2.5.2.5 Journaling in Cubit

Cubit has both a graphical user interface (GUI) and a command line interface. All GUI commands are automatically entered by Cubit into the command line, allowing them to be documented for later use. Journal files are scripts that contain Cubit commands as they would be entered into the command line interface. It is recommended that all meshing be done using journal files (.jou file extension). This provides repeatability, which is critical to creating accurate meshes.

APREPRO (An Algebraic Preprocessor for Parameterizing Finite Element Analyses) is a system built into Cubit for executing basic mathematical and logical commands. This includes setting variables, evaluating mathematical statements, and executing loops. These capabilities can be very powerful in helping the user create a generalized meshing experience. This dissertation provides examples of the utility of APREPRO functionality in journal files. APREPRO statements are indicated in Cubit by the use of braces `{}`. A sample journal file for creating a mesh of a composite box beam intended for static tip loading is provided in Appendix I for reference.

2.5.3 The Meshing Process

Meshing flexible rotor components in Cubit generally follows a standard process. First the geometry is imported. Then, it is cleaned up and decomposed into meshable volumes. Next, these volumes are imprinted upon one another and merged, allowing a continuous mesh. Once merged, the volumes can be meshed. Following meshing, blocks, sidesets, and nodesets are assigned, identifying key features within the mesh. Finally, the mesh is evaluated, cleaned-up, and exported for analysis.

In some cases, steps may be taken out of order or repeated (e.g., requiring more clean-up after imprinting or realizing further decomposition is necessary late in the meshing process) but the general outline provided herein should hold in most cases. Note that although these steps are presented for Cubit, the process should be similar when using other hexahedral meshing tools.

2.5.3.1 Import and Examine the Geometry

The first step in the meshing process is to import the 3-D geometry. This can be done with multiple file formats including STEP, IGES, and STL. Importing a STEP file in Cubit is performed by entering the command:

```
import step “filedir/filename.stp” noheal
```

The `noheal` statement at the end is optional, and tells Cubit not to automatically heal the geometry. Healing may be helpful, but also may create new problems with the geometry under certain circumstances where there are multiple volumes or small-scale geometries. See Section 2.5.4.2: Tolerance Issues, for more details.

When Cubit imports 3-D CAD geometry it will assign numerical identifier labels (IDs) to features (volumes, surfaces, curves, and vertices) automatically. However, Cubit is not always consistent with how these IDs are ordered. If one imports a complex blade into Cubit multiple times, the ID of a specific curve (e.g., that defining the trailing edge) may change from one import to the next. This is problematic because all meshing using journaled commands reference specific feature IDs. For example, the command “`mesh surface 10`” will not work if the surface that needs to be meshed is not ID 10 every time the journal file is run.

To avoid problems with IDs shifting on import, it is recommended that a geometry file be imported one time only at the very beginning of the meshing process. Then, the mesh file should be saved as a Cubit .cub file. The .cub file contains the geometry and all of its parameters, including feature IDs, and can be loaded identically every time the journal file is used. Therefore, the first few commands of a journal file may read:

```
#import step "file_dir\file_name.stp" noheal
#save as “file_dir\file_name_import.cub”
open "file_dir\file_name_import.cub"
```

The above code has the `import` and `save` functions commented out, so it begins with the open command. This ensures the exact same geometry is used every time.

Once the geometry has been imported, it must be inspected to ensure quality. The geometry must be checked to see if it matches the CAD geometry as expected. Measurements should be taken in Cubit to check its orientation in the global reference frame. If the model is imported correctly, the designer can begin identifying which volumes are meshable using map and sweep and which ones are not. Cubit offers tools to help identify meshable geometry, but their use is beyond the scope of this dissertation.

2.5.3.2 Clean-up and Decompose Geometry

The next step in the meshing process is to clean up the geometry and decompose it into smaller pieces of meshable geometry. The goal of the clean-up step is to identify and eliminate spurious vertices, curves, and surfaces which will influence the mesh in an undesired manner. The goal of decomposition is to break the imported part into volumes which can be meshed by the various algorithms in Cubit. Decomposition can also help provide greater control over the final mesh, allowing the user to position nodes where needed for later joint connections in the structural analysis model.

Clean-up may be needed immediately after import, and will often be needed after decomposition and again later after imprinting/merging geometry. Decomposition is best done only once. However, there may be times where the designer creating the mesh does not notice an additional step of decomposition that will make the mesh better or easier until late in the process, for example after imprinting/merging or even after meshing has begun. Because these later tasks will reference specific geometry IDs (e.g., "`mesh surface 42`"), going back to insert additional decomposition before the completed tasks can create unnecessary hardship by changing later geometry IDs. In this case, additional decomposition can be added in the later stages of the meshing process, and may need to be followed by additional clean-up and imprinting/merging.

2.5.3.2.1 Cubit Geometry Clean-up

When importing geometry from CAD files, Cubit may create curves and vertices where none existed (or where the designer did not realize any existed) in CAD. This can lead to small curves and surfaces as well as out of place vertices which affect meshing. Surfaces that are too small to render at the default zoom level can make geometry, which seems otherwise meshable, invalid. Vertices that break up what should be continuous curves can force Cubit to place nodes where the user does not desire them, leading to poor quality meshes.

Any small errors in the geometry need to be cleaned-up in Cubit before meshing can proceed. Even if none are present upon import, errors can appear upon decomposition and again after imprinting. Therefore, the clean-up step may need to be repeated multiple times during the meshing process. For simple parts, on the other hand, it may not be needed at all.

The first step in cleaning-up the geometry is identifying problems in the geometry. One useful way to search for small geometry is by using the "Detect Small Features" tool found in the Immersive Topology Environment for Meshing (ITEM) Wizard under "Prepare Geometry/Remove small features". This tool will identify curves and surfaces with measurements under a certain threshold set by the user, listed from smallest to largest. This can help locate small geometry, too small to identify on the screen by eye.

Sometimes large geometry can also be problematic, especially in the form of spurious vertices. For example, an otherwise good curve may be broken in two by a vertex placed somewhere in its middle. Such out of place vertices can be identified by switching Cubit to wireframe view and using the command "`vertex visibility on`", thereby highlighting all vertices. This allows the user to visually scan for vertices that are breaking up curves that are too big to be caught by the "Detect Small Features" tool. Increasing the size of the displayed vertices using the command "`graphics point size 2`" can make them even easier to identify.

As bad geometry is identified, it must be cleaned-up. The primary tools for fixing bad geometry are the "`composite`" commands. Small surfaces can be composited to larger ones to form a single virtual surface, which Cubit can then mesh as a single continuous entity. If the leading edge of a rotor blade is imported into a cubit as a large number of narrow surfaces, compositing can create one continuous virtual surface for meshing. Similarly, curves can be composited together into a single piece of virtual geometry. If the trailing edge of a blade is broken up by a spurious vertex somewhere along its length, the composite command will create a single continuous virtual curve, allowing the final mesh to ignore the single spurious vertex.

Depending on how the CAD geometry was created, Cubit may generate spurious small surfaces that are non-physical. These can be dealt with using the "`remove surface`" command. Other options for cleaning-up geometry can be found within the ITEM Wizard under "Prepare Geometry/Remove small features".

During the initial clean-up/decomposition stage, other modifications to the geometry may be necessary to make the volumes meshable. A common example is splitting surfaces along fillets, then compositing each half of the now-split filleted surface to its intuitive parent. Such concepts are discussed in the Cubit documentation [121] and left out of this dissertation.

2.5.3.2.2 Decomposition

As described above (Section 2.5.2.1: Meshing Algorithms in Cubit), the two main solid volume meshing tools in Cubit are mapping and sweeping, each of which is best suited for different types of geometry (Figure 2.9, Figure 2.10, and Figure 2.11). The mapping algorithm works on cuboid geometry that can support a Cartesian grid, which is to say a shape with eight corners connected by curves. The sweeping algorithm can mesh geometry which has a "2.5-D" shape (as defined in Section 2.5.2.1), wherein one face is swept along an axis to an opposite face with similar topology. Any volume which can be mapped can be swept.

The goal of the decomposition is to break apart the geometry into pieces which are capable of being meshed. Beyond making un-meshable geometry meshable, partitioning the geometry through decomposition can help the user control the mesh. A simple example is the box beam shown in Figure 2.12. Because it is an extruded shape, the original volume can be meshed by meshing the root and then sweeping. However, paving such a thin walled root surface may create a low quality mesh. Instead, it one can decompose the geometry into walls and corners as shown in Figure 2.13. This allows each piece to be meshed independently, giving the user more control over the final product.

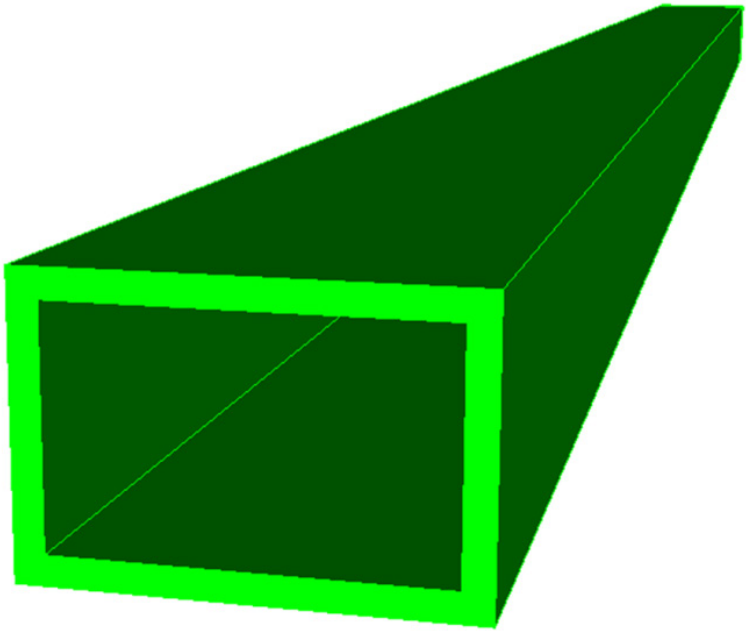


Figure 2.12 A single volume box beam as imported into Cubit; it can be meshed by starting at the root surface then sweeping, but decomposition would give the user finer control over the final product

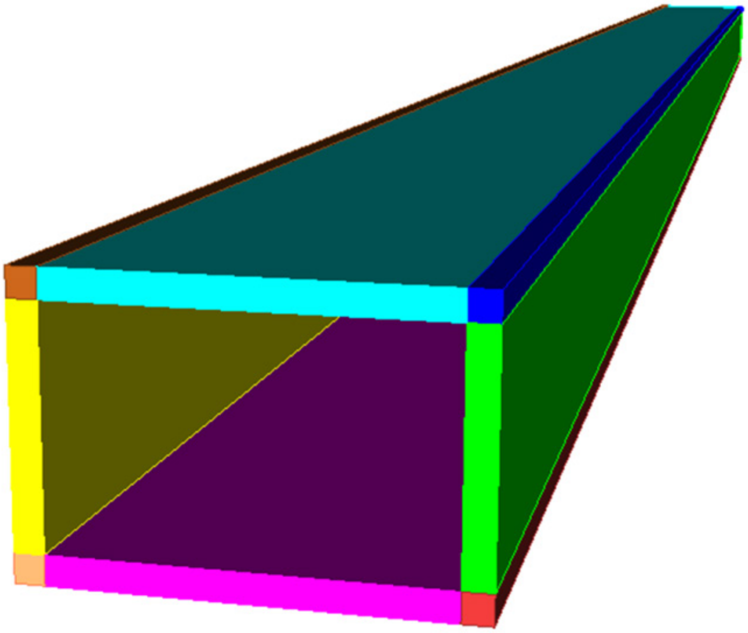


Figure 2.13 A box beam decomposed in Cubit into eight separate volumes, four for the walls and four for the corners

In general, a rotor blade will tend to be amenable to sweeping over the length of its radius. In fact, for a simple blade, the majority of a blade may require no decomposition at all if the different spar components are defined by discrete bodies in the original CAD part file (discussed in Section 2.3.2: Assemblies, Parts, and Bodies). The root portion of a blade, however, may prove more difficult to mesh. In this region a spar might undergo significant changes, including changing shapes of the leading and trailing edges of the airfoil, that can make meshing difficult. It is often good practice to begin decomposition by separating the root from the main aerodynamic length of the blade. Then, the root can be decomposed further into meshable volumes if required. An example of this is shown for a notional tubular spar blade without skin in Figure 2.14.

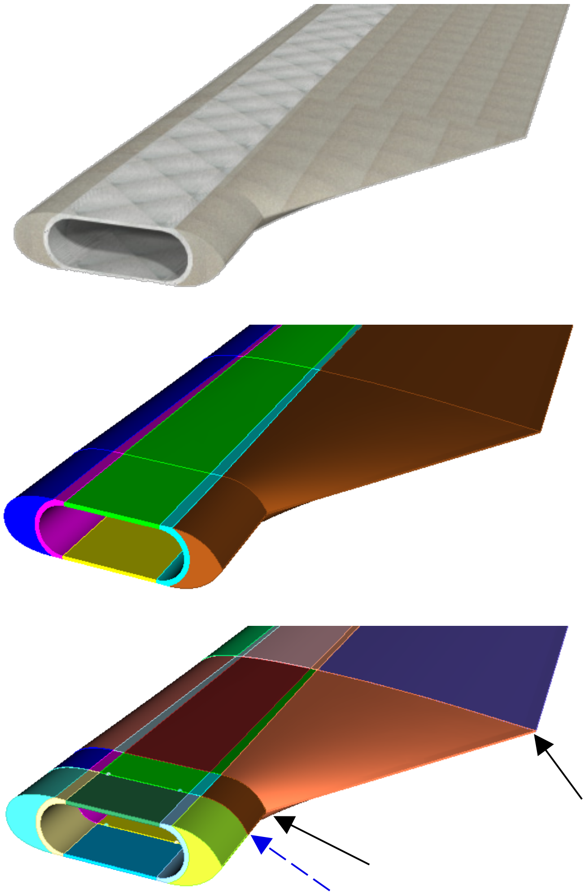


Figure 2.14 A tubular spar blade with no skin being decomposed for meshing: (top) CAD geometry, (center) geometry as imported to Cubit, (bottom) geometry decomposed into meshable regions with cuts at two radial stations marked by solid black arrows; an additional cut is made to place the root bolt nodes, marked by a blue dashed arrow

In more advanced blades, the geometry of the internal structure may change significantly over its radius. Components such as a servo flap or a tip weight can break up the natural sweeping flow of the blade. Large changes in airfoil cross-section or internal spar shape may also cause meshing difficulties. In these cases it may be necessary to decompose the blade by cutting it at several radial stations. Then, the cross-section surfaces at each station may be meshed carefully, allowing the user to create a continuous swept mesh of good quality even for very complicated blades.

In addition to creating meshable geometry, decomposition may be needed to exert control over the mesh to allow node placement for later joint creation. An example of this is shown in Figure 2.14, which has an additional cut placed across the innermost portion of the blade. This forces Cubit to place nodes along the plane of the cut, which will later aid in positioning nodes for joint placement, as discussed further in Section 2.5.3.4.3: Placing Nodes for Joints.

2.5.3.2.3 Webcutting in Cubit

Decomposition in Cubit is accomplished by using the "`webcut`" operation. There are several types of webcut which may be used in Cubit. The most basic involve separating volumes using reference planes. One useful tool allows webcutting based on a sheet extended from a surface, in which Cubit automatically extrapolates a complex 3-D surface into a cutting sheet. This can be useful when trying to use complex internal geometry to cut through the skin of a blade, for example. Figure 2.15 shows an example of a twisted box beam, the walls of which are webcut using sheets extended from the four internal surfaces of the box, allowing cuts at the corners to be formed using the curved surfaces of the geometry.

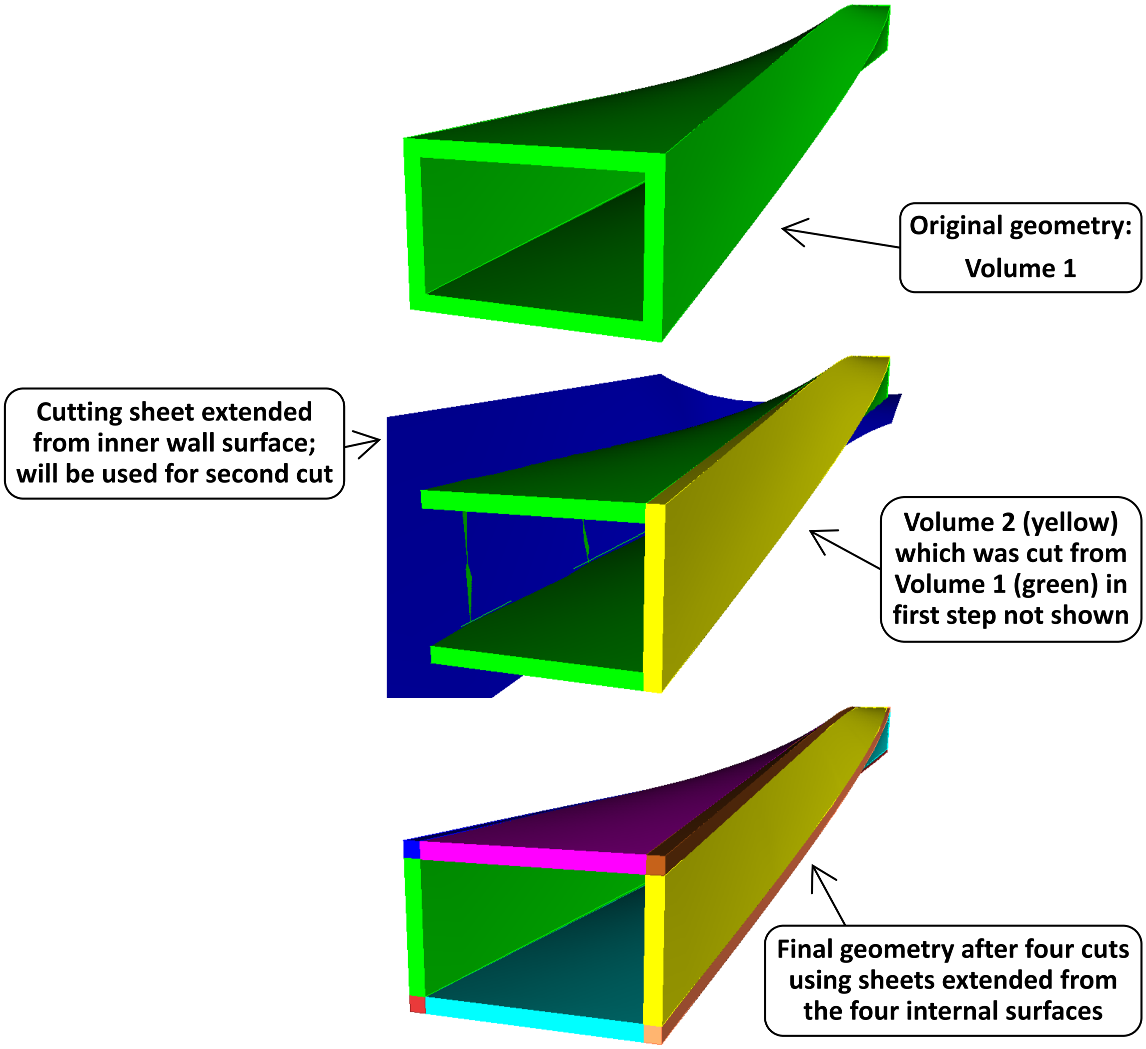


Figure 2.15 A twisted box beam being decomposed using "`webcut with sheet extended from surface`": (top) original geometry as imported into cubit; (center) the box beam after the first webcut (right hand wall separated from the rest of the box) with a preview of the second webcut (a sheet extended from the inner surface of the left hand wall), (bottom) geometry after fourth and final webcut

At times it may be necessary to decompose a body based on two curves which define similar features on opposite surfaces. For example, consider a geometry in which the quarter chord line is defined on the upper and lower surfaces of a blade using corresponding curves. If the user desires the mesh has nodes placed at the quarter chord through the thickness of the blade, webcutting the blade on a plane connecting these curves will create an interior surface, forcing Cubit to place nodes on it. This can be done by creating a new surface using the “`create surface skin curve`” command. Then, this new surface can be used to decompose the geometry using either the “`webcut with sheet`” or “`webcut with sheet extended from surface`” commands.

It should be noted that decomposition can sometimes lead to small geometry. When webcutting with an extended surface, for example, it is possible that the cutting sheet does not cleanly cut the intended volumes. This can lead to sliver surfaces with very small surface area, or even tiny volumes which should not exist. After decomposing the geometry it is important to check over the geometry again for any additional clean-up required.

2.5.3.2.4 Symmetry

Decomposition can also be used to take advantage of symmetry in a part. While rotor blades are typically not symmetric, other inboard components may be. Parts can be decomposed along one or more planes of symmetry; volumes on one side of the plane are deleted, and the remaining volumes are meshed. This can often reduce the difficulty of a meshing task and guarantees a symmetric mesh. If the user intends to exploit symmetry, an additional step between Mesh the Geometry (Section 2.5.3.4) and Define Blocks, Sidesets, and Nodesets (2.5.3.5) must be taken to reconstruct the full part. First, the volumes must be copied, including their meshes, using the command “`volume all copy reflect ...`” and then entering an option for the plane of symmetry, which can be a geometric reference plane (x, y, or z) or defined using a curve normal to the plane. After that, a final "`merge all`" command will be needed to merge the nodes which lie on the plane of symmetry.

2.5.3.3 Imprint and Merge

Once the geometry has been decomposed into meshable volumes, adjacent volumes must be joined together. This is accomplished within Cubit by imprinting and merging. The imprint command creates corresponding vertices, curves, and surfaces on adjacent volumes. The merge command then joins adjacent and identical geometry, so two surfaces (or curves, or vertices) become one. This step is essential to creating a unified mesh. If skipped, volumes created in the decomposition stage will be meshed independently, leading to mismatched meshes. By imprinting and merging, a continuous mesh is created across volumes.

The importance of imprinting and merging is shown by the example of a cube adjoining a cylinder in Figure 2.16. In the top portion of the figure, the two volumes are meshed independently. The cube is mapped and the cylinder is swept, with its paved surface adjacent to the cube as its source, and the surface facing the reader as its target. The two meshes do not match at their interface; the mesh is not continuous and cannot be used for analysis. In the bottom half of the figure, the cylinder and cube have been imprinted on one another and merged. This makes the mesh at the interface match, affecting the mesh of both volumes. The effect on the cube is clear, with a new curved influence from the mesh where the cylinder is imprinted; now the cube must be swept using two source surfaces, the one shared with the cylinder (hidden) and the one visible to the reader. Similarly, the cylinder must be swept from newly the merged surface and the remaining un-merged portion of the (hidden) circular surface, towards the single target surface facing the reader. Note that the effect of the straight edge of the cube can be seen on the target surface of the cylinder, where it has nearly straight mesh edges called-out by red arrows.

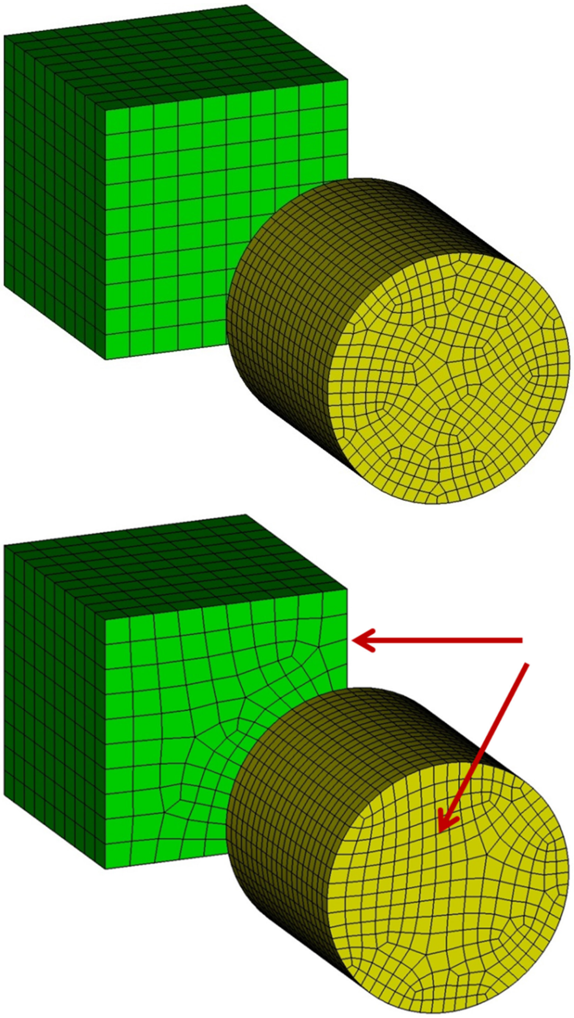


Figure 2.16 Example of a block and a cylinder (top) meshed before imprinting and merging, leading to two independent meshes; and (bottom) after imprinting and merging, leading to a single continuous mesh – red arrows call-out the rectilinear mesh pattern on the surface of the cylinder caused by imprinting the cube on their shared surface

For simple geometry, imprinting and merging can often be completed automatically by Cubit. The command “`imprint all`” will detect adjacent geometry and perform the imprint operation. This is followed by “`merge all`” which automatically finds pairs of identical overlapping geometric entities and merges them. With more complex geometry this process can become more difficult. To diagnose imprint and merging problems as they occur, the commands “`draw surfaces is_merged`” and “`draw surfaces not is_merged`” can be used to identify surfaces that did or did not merge properly. Problems may arise due to tolerance issues (Section 2.5.4.2) or bad geometry. Some potential solutions are provided in Section 2.5.4.3: Troubleshooting Imprint/Merge Problems.

Imprinting can also be used to control the meshing process. Often, it is necessary to place nodes at specific locations for joint attachments. Reference curves and vertices can be imprinted onto surfaces to assist in this effort. Imprinted curves can split a surface, forcing Cubit to place nodes along the curve. Similarly, imprinting a vertex will force Cubit to place a node at its location. This reference geometry can be imported from the CAD model or created within the journal file. An example of this process is given in Section 2.5.3.4.3: Placing Nodes for Joints.

2.5.3.4 Mesh the Geometry

Once the part has been decomposed, imprinted, merged, and cleaned-up, meshing can be performed. While a detailed tutorial on meshing a general 3-D geometry is beyond the scope of this work, this section offers specific methods for meshing rotor blades, then discusses special node placement for defining joint connections between meshed parts. The general process for meshing a rotor blade consists of meshing the surfaces of a cross-section then sweeping the surface mesh outboard or inboard (or both) to complete the mesh.

2.5.3.4.1 Mesh Cross-section Surfaces

The first step in the meshing procedure is to select an appropriate cross-section for surface meshing. For a simple blade, this section may be the exposed surface at the tip or root of a blade. In general, an appropriate section should have been created during geometry decomposition using the webcut command, such as when the root was separated in the example shown in Figure 2.14. For more complex blades with internal geometry that varies greatly over the radius, it may be necessary to surface mesh multiple cross-sections over the length of the blade. Continuing the simple example of the tubular spar blade introduced in Figure 2.14, Figure 2.17 shows a mesh for the cross-section between the root and main lifting portion of the blade.

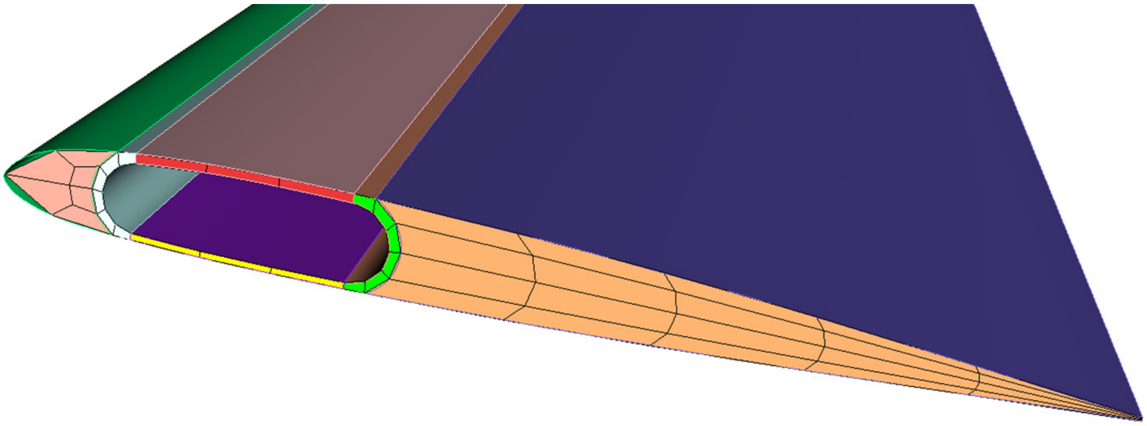


Figure 2.17 Cross-section surface mesh for a notional tubular spar blade without skin

The first step in meshing the cross-section surfaces is determining the size of the mesh. This can be done by setting either the approximate mesh size or the number of elements around the perimeter of the surfaces. Approximate mesh size can be specified using the command "`surface ## size {M_Size}`", in which `M_Size` is an APREPRO variable (defined in Section 2.5.2.5) containing the size of the mesh in the units of the geometry imported into Cubit (recall from Section 2.5.2.4 that Cubit is unitless, so this size may be millimeters or another unit). While easy to implement, specifying approximate element size gives imprecise control over the final mesh.

The other means of controlling the mesh size is to specify the number of elements around the perimeters of the surfaces which define the cross-section. This is performed curve by curve, with commands in the form of "`curve ## ## ## interval {M_Int1}`", which sets several curves to be meshed with an interval count stored in `M_Int1`, resulting in that number of elements along that perimeter. This gives much more precise control over the mesh but it may require many more commands than specifying an approximate mesh size, and may be more difficult to set up using APREPRO variables. Take the example in Figure 2.17: the curves on top and bottom of the leading edge each have an interval of 2; the four curved boundaries of the fore and aft spar webs have an interval of 5; the four horizontal curves at the top and bottom of the upper and lower spar caps each have an interval of 2; the curves defining the boundaries between the spar walls and caps each have an interval of 1; and the top and bottom surfaces of the trailing edge core have an interval of 5. The problem is that some of these values depend on one another; for example, paving the leading edge root surface requires an even number of intervals around the perimeter (a constraint of the pave meshing algorithm), therefore the number of intervals on the top and bottom must be linked algebraically to the number of intervals on the vertical curve at the front of the shear web. This makes creating robust meshes linked to variables more difficult. However, this precise control may be necessary when meshing complex blades with changing internal structures.

Once the size of the mesh is determined, either by approximate element size, intervals, or a combination of the two, the surfaces can be meshed. Cubit can automatically select a meshing algorithm, or the user can specify one. Surfaces may be best meshed using `map`, `pave`, `circle`, `hole`, or another scheme.

2.5.3.4.2 Mesh Volumes

Once the cross-section surfaces have been meshed, the radial size of the mesh is set. Once again, this can be specified by setting an approximate element size or by setting an interval count. If a good cross-section mesh has been achieved, then the blade should be able to be swept regardless of the radial mesh size. This means that both intervals and approximate element size are well suited to APREPRO variables, allowing radial mesh size to be easily adjusted.

Next, the blade volumes can be meshed, as shown in Figure 2.18. With simple rotors, Cubit will often be able to automatically detect the best meshing scheme (map or sweep) and apply it automatically using a simple “`mesh volume ## ## ##`” command. If automatic meshing fails, the user can specify one or multiple source surfaces and a single target surface for the sweep command. If this fails, reasons could include incorrect interval specification, bad/un-meshable geometry, or poor imprinting/merging. Section 2.5.4: Advanced Concepts in Meshing discusses methods for dealing with some of these problems.

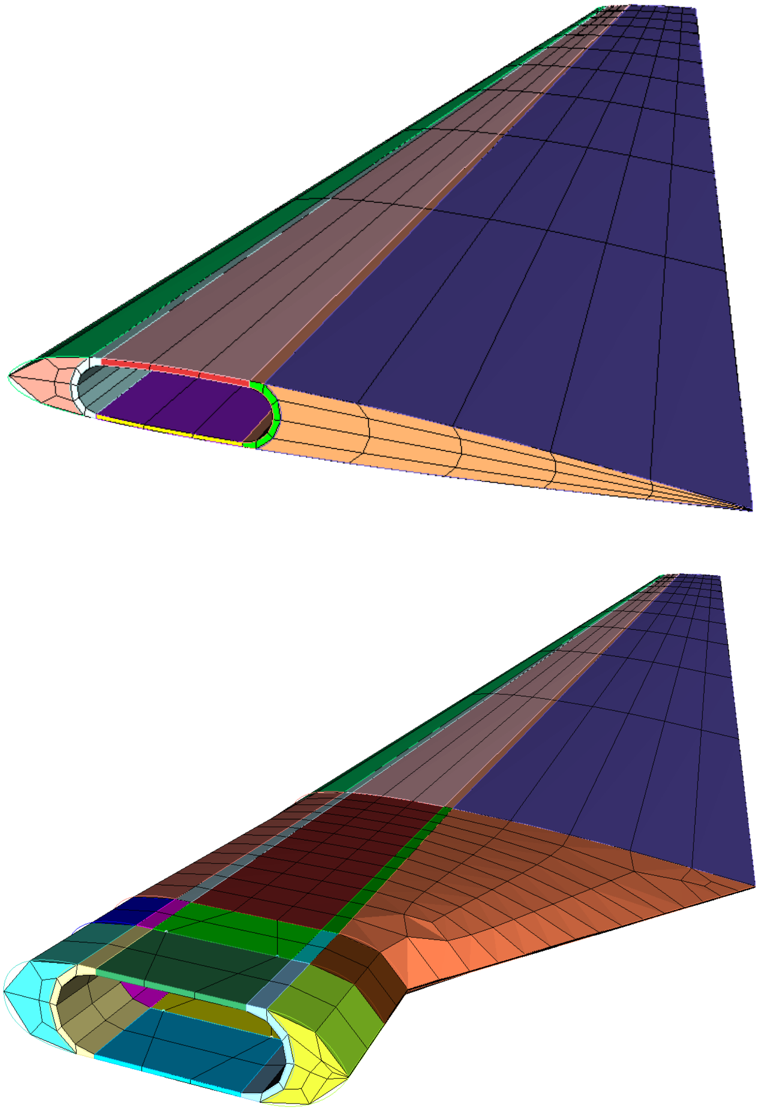


Figure 2.18 Volume meshing of a tubular spar blade: first the cross-section surface mesh is swept outward to mesh the outboard volumes (top) then it is swept inward to mesh the root volumes (bottom)

Note that this process does not always occur in a single step, with one cross-section being swept the length of the blade. If the internal geometry of blade changes significantly there may be several cross-sections that need their surfaces meshed independently at the ends of the volumes that need to be swept carefully. The root section may be especially difficult to mesh, requiring the blade cross-section mesh to be swept inboard piecemeal to ensure a good quality sweep.

2.5.3.4.3 Placing Nodes for Joints

When meshing, nodes may need to be positioned in precise locations to form joint connections in X3D. Selection of how to position nodes depends on the type of joint desired. There are three basic ways of selecting nodes to join parts: by element, by face, or individually. Examples of these three methods for identifying joint connection nodes are illustrated in Figure 2.19 for a notional proprotor yoke. Element joints are used to connect parts based on entire volumes; in the example, elements are used to "plug" a bolt hole, modeling the bolted connection without adding six separate flex parts. Selecting nodes based on faces allows the user to mate surfaces, possibly to represent bonded or clamped joints. Individual node selection is represented in this example by an axial joint, in which nodes lying along an axis are selected for connection to the joint part.

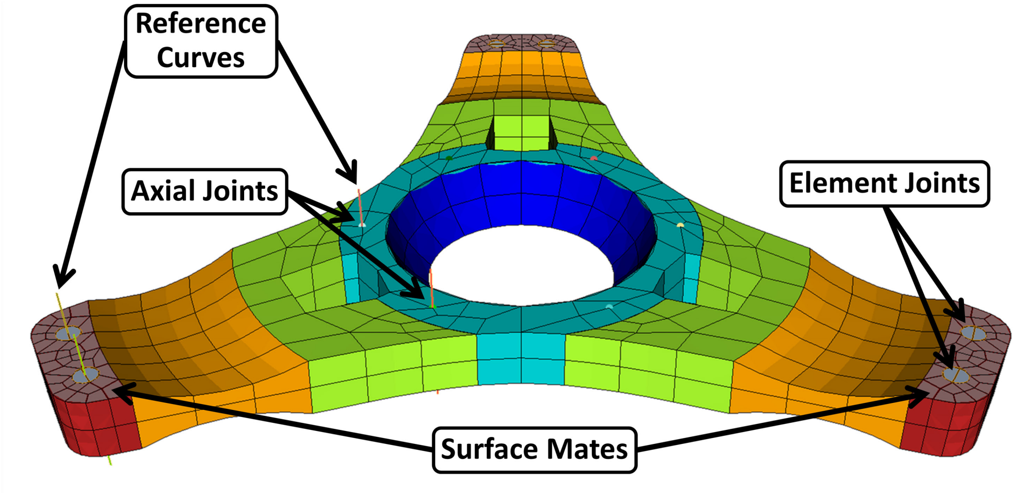


Figure 2.19 Notional proprotor yoke geometry demonstrating three methods of identifying nodes for joint connections: nodes selected by elements (element joints), nodes selected by face (surface mates), and nodes selected individually along a reference curve (axial joints)

Depending on the type of joint connection desired, there are different methods for positioning the nodes appropriately. Webcutting during the decomposition stage is a useful way of controlling the placement of nodes to achieve the desired joint connections. In the example above, the bolt "plugs" can be created by webcutting the yoke geometry with cylinders, forcing the mesh to fill the bolt holes precisely. These webcutting cylinders are defined in Cubit using reference curves imported from the CAD model (called out in Figure 2.19) as centerlines.

Another means of controlling the positioning of nodes is to imprint reference geometry on to the main model. Figure 2.20 shows an example of positioning nodes along an axis to represent bolt connections on a tubular spar blade. In this case, the geometry was webcut along the plane containing the two bolt axes during the decomposition phase, giving an internal surface that will support the mesh. Two reference curves which were imported from the CAD define the two bolt axes, and vertices are created where these curves intersect the spar. These vertices are then imprinted on the upper and lower surfaces of the spar caps. When the volumes are meshed, Cubit is now forced to create nodes at the imprinted vertices, thus ensuring nodes are placed where desired, as seen in Figure 2.21. Alternatively, the reference axes could have been directly imprinted on the interior surfaces of the geometry ("`imprint surface 125 135 with curve 188 189`"), thereby splitting the internal surfaces and forcing nodes along the bolt axes.

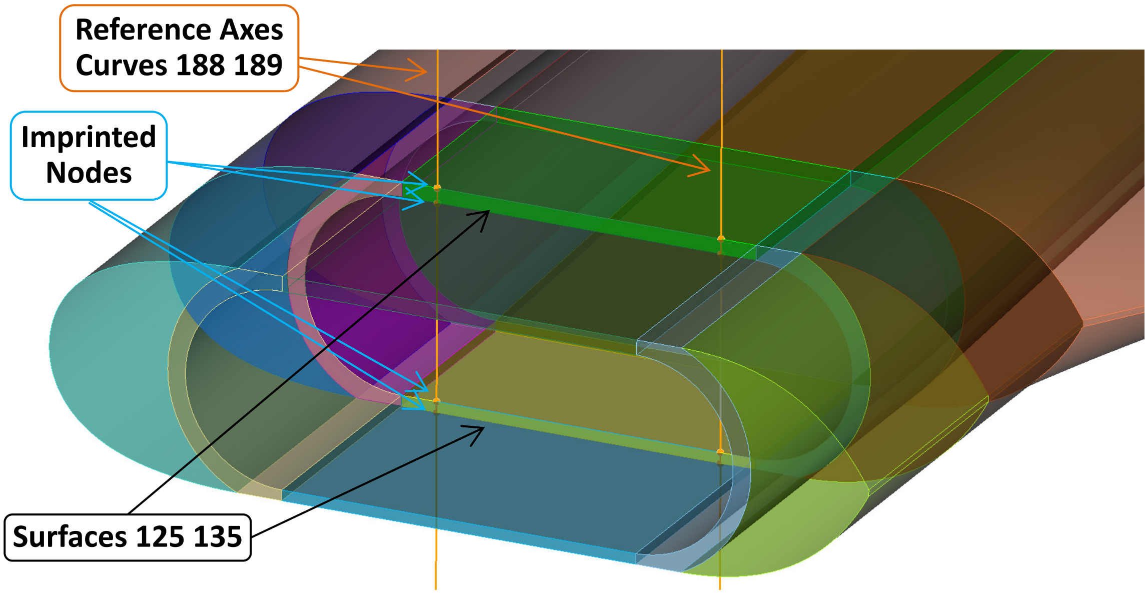


Figure 2.20 Root section of a tubular spar blade, demonstrating imprinting reference geometry in order to control node placement

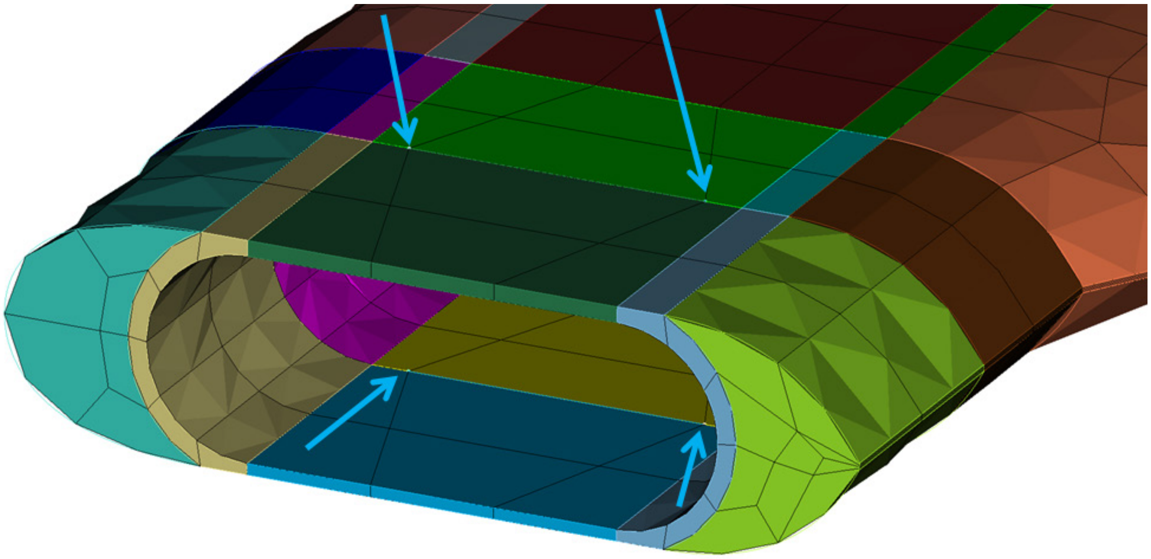


Figure 2.21 Root section of a tubular spar blade, showing how node placement was controlled by imprinting vertices created from the reference geometry in Figure 2.20

2.5.3.4.4 Checking Mesh Quality

As cross-sectional surfaces and volumes are meshed, their quality should be inspected. As discussed above in Section 2.5.2.2: Mesh Quality, Cubit has a variety of quality metrics built into the program. These metrics, however, are defined for Hex8 elements, and there are no benchmarks for Hex27 quality. That being said, it is still valuable and informative to inspect the Hex8 (and four-node quadrilateral Quad4) quality metrics. The default quality metric, obtained by the command "`quality volume ## draw mesh`" or "`quality surface ## draw mesh`" is shape. The shape metric describes how close a hex element is to an ideal hex, i.e. a cube, or how close a quad element is to a square. While helpful for identifying distorted elements, it will give poor results for elements with low aspect ratios, for example those used to mesh thin volumes coarsely. The use of Hex27 nodes eliminates the risk of locking with low aspect ratio elements; however, there may still be a relationship between the shape metric and solution accuracy.

The scaled Jacobian quality metric can be more useful in evaluating the mesh. It can be calculated using the commands "`quality volume ## scaled draw mesh`" and "`quality volume ## scaled draw mesh`". The scaled Jacobian measures how close to normal the edges of the elements are. For example a Quad4 that forms a perfect rectangle has a scaled Jacobian of 1.0, whereas a Quad4 that forms a triangle (three of the four nodes lie on a line) will have a scaled Jacobian of zero. A concave element (one or more internal angle greater than 180°) has a negative scaled Jacobian. It is important that the scaled Jacobian be greater than 0 for the FEA solution to be successful.

Although different than assessing mesh quality, topology checks fall under the quality banner in Cubit. This includes the coincident node check, which identifies any nodes that occupy the same coordinates in a volume to a specified tolerance. This will identify any surfaces that were not imprinted and merged properly and were therefore meshed independently. This is also likely to occur when copying mirrored volumes to take advantage of part symmetry (Section 2.5.3.2.4). The coincident node check can be run from the Command Panel, or using the command "`topology check coincident node volume all tolerance 1.0e-6 draw brief result group`".

2.5.3.5 Define Blocks, Sidesets, and Nodesets

Once the blade is meshed, blocks, sidesets, and nodesets must be defined. These are essential tools for communicating details about the part mesh with the X3D solver. Blocks are groups of solid elements and are used to define material properties. They are also necessary to change the mesh element type from Hex8 to Hex27 in Cubit. Sidesets are groups of faces and are used primarily to define aerodynamic surfaces. Nodesets are groups of nodes and serve various functions, including identifying joint connection points and cross-sections.

2.5.3.5.1 Blocks

Blocks in Cubit are groups of elements primarily used to identify mesh elements which form internal structural components. X3D uses these blocks to assign material properties within the structural analysis model. Blocks in X3D have IDs ranging from 201 to 299 and are numbered sequentially, with no skipped values. Every element must belong to a block, and each element can belong to only one block. Blocks may be assigned by volume, or by specific elements within a volume.

For example, consider the case of the tubular spar blade introduced in Figure 2.7. This blade model has six main structural components, each of which could be made out of a different material: a leading edge core, a trailing edge core, two spar caps, and two spar webs. Since the CAD model was created with multiple bodies based on material, these are each imported as a separate volume (Figure 2.22). Although the part is decomposed into many smaller volumes, the mesh of each corresponds to one of the original volumes (Figure 2.23). As such, blocks can be assigned by volumes using the command "`block 20# volume ## ## ## ##`". For this blade, there are six blocks, as illustrated in Figure 2.24.

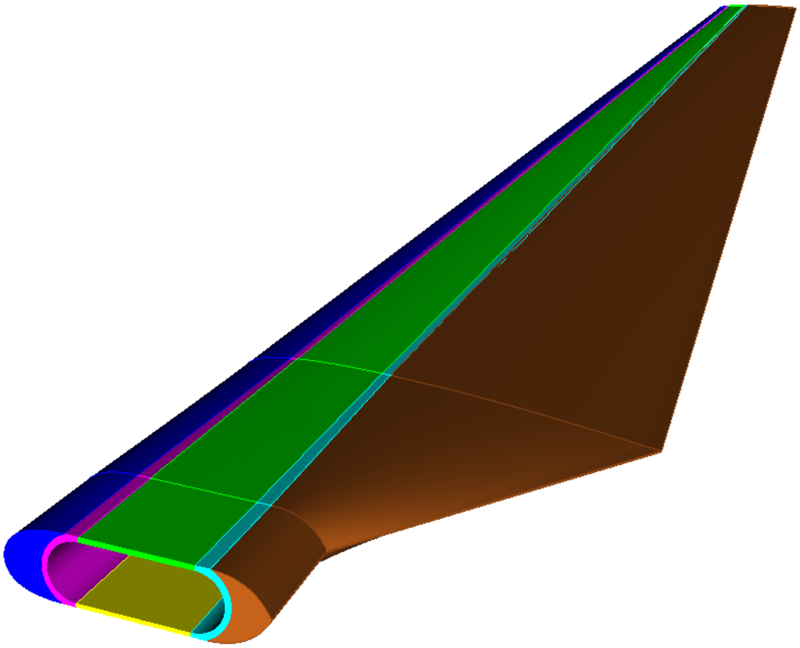


Figure 2.22 A notional tubular spar blade with no skin as imported into Cubit, showing six discrete volumes for each of the internal components

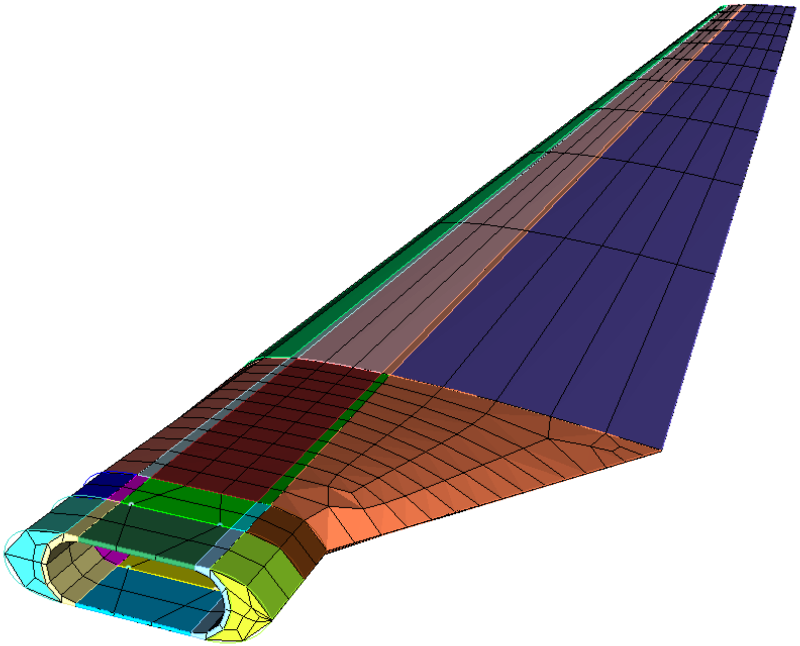


Figure 2.23 The tubular spar blade mesh, which follows the boundaries of the original six volumes in Figure 2.22 despite further decomposition

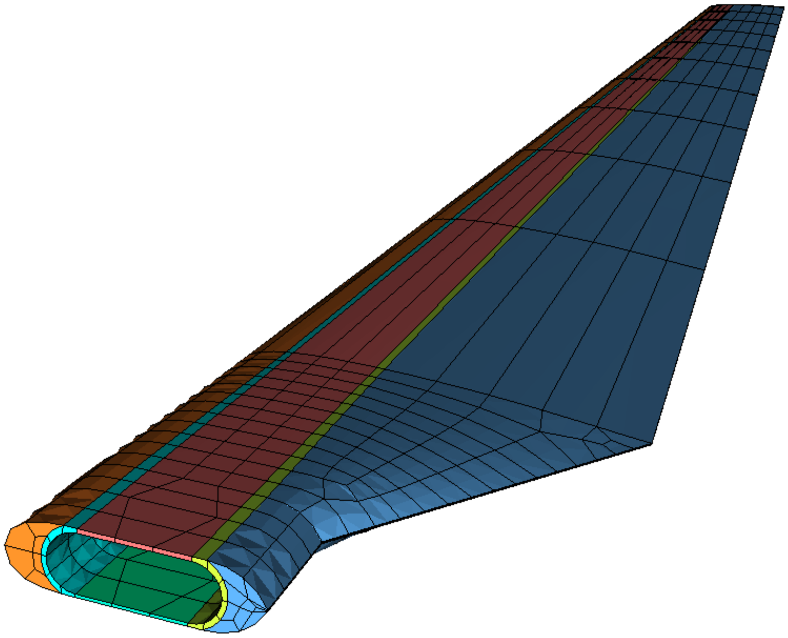


Figure 2.24 The tubular spar lade mesh with each of the six different blocks highlighted in a different color

Once the blocks have been identified, the user may provide them with material names within Cubit. These are optional descriptors which are not maintained when the mesh is processed for use in X3D. However, including them will conveniently initialize the block/material list in the processed mesh data .dat file, including allowing multiple blocks to share the same material (if material is not assigned to a block, it will be given a default material tag). For the example of this blade, consider the case where the four spar members (Blocks 201-204) are all the same composite fiberglass material, the leading edge core (Block 205) is structural foam, and the trailing edge core (Block 206) is an aramid honeycomb. The journal commands for defining the materials are:

```
create material name 'Mat_Glass'
create material name 'Mat_Foam'
create material name 'Mat_HComb'

block 201 to 204 material 'Mat_Glass'
block 205 material 'Mat_Foam'
block 206 material 'Mat_HComb'
```

Finally, blocks are used to define the element type. Cubit defaults to eight node hexahedral elements (Hex8) for its meshes, whereas X3D requires 27 node hexahedra (Hex27). Once the blocks are established, the command to change the element type to Hex27 must be included: "`block all element type Hex27`". The conversion process from Hex8 to Hex27 can sometimes result in errors when meshing concave surfaces, a known error in Cubit which is discussed in 2.5.4.1: Misplaced Internal Nodes on Concave Volumes.

2.5.3.5.2 Sidesets

Similar to a block, a sideset is a collection of element faces belonging to a part mesh. A sideset can have any number of faces, and a face can belong to multiple sidesets. Each face is defined by an element ID and a natural face ID from one to six. The nodes belonging to the specified element face can be extracted using the mapping for Hex27 elements shown in Figure 2.25.

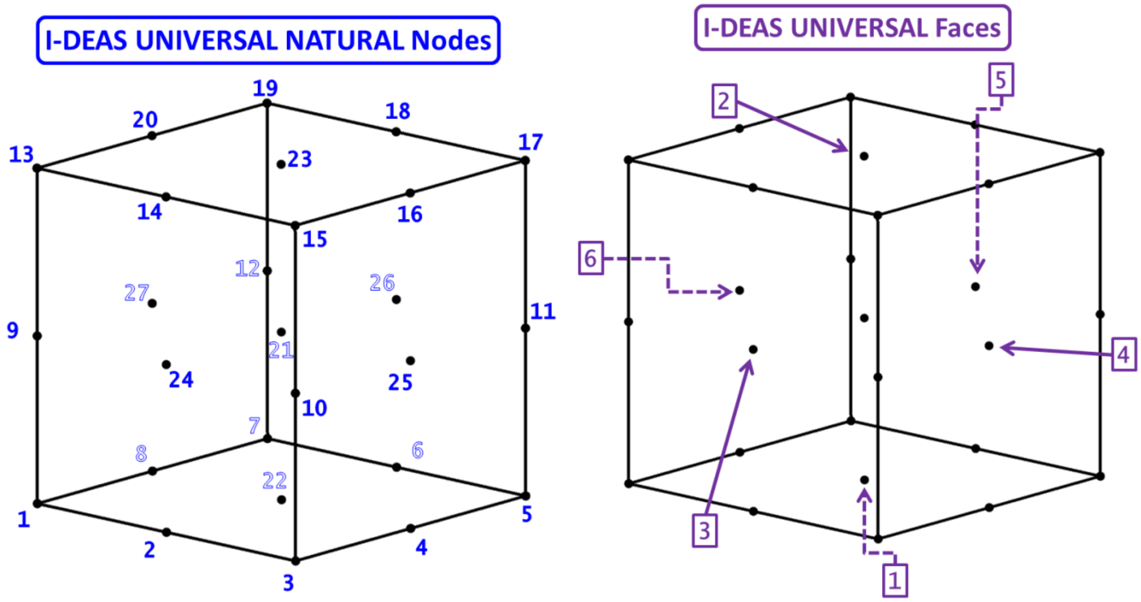


Figure 2.25 Mapping of node IDs to face IDs for a Hex27 element (Note: node IDs are based on the I-DEAS Universal .unv ordering used in this research, not the X3D standard Hex27; see Section 2.6.1.2: Natural and Global Node and Face IDs for more details)

Sidesets are useful for multiple purposes, and by convention are given a three digit ID in a different range depending on their application. Each class of sideset has a different first digit for its ID, and IDs within a class are assigned sequentially, with no skipped values (except the special case of SS300 and SS301 noted below).

Class 300 sidesets are those with IDs starting from SS300 and are defined as standard sidesets. They are used for defining part boundaries and aerodynamic interfaces, with potential for other applications. Sideset ID SS300 is reserved for element faces which define a boundary. Boundaries are surfaces marked as having zero deflection for structural testing of isolated parts, and are not used in assembled structural analysis models. If there is no boundary, the ID SS300 is left unassigned. Currently, SS300 is not used by X3D, and boundaries are defined using only nodeset NS400 (described below). Sideset ID SS301 is reserved for wetted aerodynamic interface faces, which are marked as being in contact with the air around the rotor for CFD coupling. If there are no wetted surfaces on the part, SS301 is left undefined. Sidesets with IDs greater than 301 may be used for other purposes in the future.

Class 500 sidesets, with IDs starting from SS501, are used to mark cross-sectional element faces. These are intended for use in analyzing stresses on cross-sections in the future if needed. Class 600 sidesets (numbered from SS601 up) have been defined for identifying faces on which to apply surface traction in the future if needed.

Class 900 sidesets are used to identify aerodynamic segments over a blade. These are used to apply low order lifting line aerodynamic forces to the structure in X3D and to apply aerodynamic forces calculated by CFD. Aerodynamic segments consist of radial bands containing all faces on the surface of the blade. Sideset ID SS901 belongs to the innermost aerodynamic segment, with ID increasing radially outward toward the tip. This is illustrated for the TRAM proprotor blade in Figure 2.26. Note that the very root and tip surfaces of the blade (the vertical surfaces normal to the radius) should not be included in the class 900 sidesets.

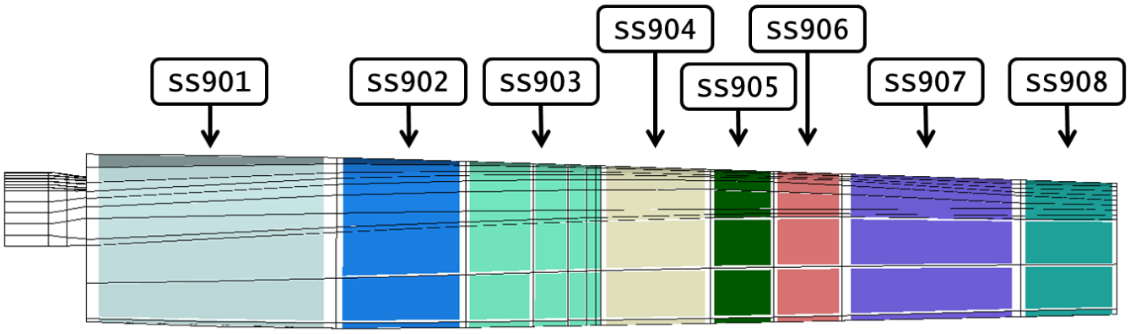


Figure 2.26 Aerodynamic segment sidesets for the TRAM proprotor blade

The method for specifying faces belonging to sidesets varies based on type and location. For a boundary sideset, which may be at the root end of a cantilevered beam, the boundary is likely composed of entire surfaces. In this case, the command "`sideset 300 surface ## ## ##`" will add all surfaces identified to SS300. In some cases, the boundary may consist of only specific faces within a surface. These can be selected manually by entering the face IDs (or using the GUI to select them), or using a filtered selection using APREPRO logical operators. For example, the command "`sideset 300 face in surface ## ## ## with (x_coord > {x_min}) and (x_coord < {x_max})`" will automatically add all faces in the identified surfaces to the sideset, as long as they fall within the boundaries stored in the APREPRO variables.

The aerodynamic interface sideset SS301 is always going to be defined by entire surfaces. Unlike the earlier SS300 example, however, individual faces needed to be added, so they can later be broken down into the aerodynamic segment 900 class sidesets. The command to do this is: "`sideset 301 face in surface ## ## ##`", which can also be used for SS300.

Aerodynamic segments are stored in the SS900 class sidesets. These may be defined by entering the face IDs, or selecting the appropriate faces in the GUI. Selection within the GUI can be aided by using the command "`draw face in sideset 301`" which will have Cubit display only aerodynamic interface faces, of which all 900 series sidesets are a subset. Further, if the radial mesh size is uniform (unlike the TRAM blade in Figure 2.26), the process can be automated using APREPRO loop functions. The following is an example of journal commands to automatically generate the 900 series sidesets from a uniform mesh:

```
#### Series SS900 (Aero Segments)
### Use loop to create one aero segment per radial element on blade
     surfaces

$ {SSetID    = 901}                  # ID of current sideset
$ {Sz_RadEl  = Blade_L / MInt_L}     # Size of radial elements
$ {AESurf_IB = Root_X}               # Inboard  radial boundary of aero seg
$ {AESurf_OB = AESurf_IB + Sz_RadEl} # Outboard radial boundary of aero seg

${Loop(MInt_L)}  # Loop over the aero segments
   ### Create sideset with proper faces
   sideset {SSetID} face in sideset 301 with (x_coord > {AeroSurf_IB}) and
     (x_coord < {AESurf_OB})

   ### Advance side set ID and aero surface bounds
   $ {SSetID    = SSetID + 1}           # ID of current sideset
   $ {AESurf_IB = AESurf_OB}            # Inboard  radial boundary of seg
   $ {AESurf_OB = AESurf_IB + Sz_RadEl} # Outboard radial boundary of seg

${EndLoop} # End loop over aero segments
```

In this example, the length of the blade is stored as the APREPRO variable `Blade_L`, the number of radial intervals as `Mint_L`, and the radial coordinate at the root of the first aero segment as `Root_X`. Based on this information, the journal automatically creates a separate aero segment for every radial element. It can be modified to make larger aero segments if desired.

2.5.3.5.3 Nodesets

Nodesets are collections of nodes within a part mesh. A node can belong to multiple nodesets. Standard nodesets are defined as those in the 400 class (numbered NS400 and up) and are used for listing nodes connected to joints. The specific ID NS400 is a reserved for boundary nodes. When NS400 is present in a mesh, X3D automatically recognizes it as a boundary condition and sets deflections to zero. This formulation is useful for static analysis of isolated parts. Standard nodesets with IDs from NS401 to NS450, are used to define nodes for joint connections. How nodesets are connected to joints is described in Section 2.6: Joint Definitions. Higher valued standard nodesets (NS451-499) are not currently used by X3D, and will be ignored by the solver. These unused nodesets can be taken advantage of for various diagnostic purposes, such as tracking how the IDs of a group of nodes changes during mesh renumbering (which is discussed in Section 2.6.1.4.2).

Like the standard sidesets discussed above, standard nodesets can be defined in one of several ways. The method chosen is based on the type of joints the nodes connect to, as discussed in Section 2.5.3.4.3: Placing Nodes for Joints. The simplest but most time consuming method is to manually select nodes in the GUI, resulting in a command of the form "`nodeset 40# node ## ## ##`". For joints that form surface mates, the command "`nodeset 40# node in surface ## ##`" can be useful. Additionally, if the user created a sideset SS300 boundary, the command "`nodeset 400 node in surface in sideset 300`" will extract the appropriate nodes. To find nodes within certain volumes or elements the command is simply "`nodeset 4## node in volume ##`" or "`nodeset 4## node in element ##`".

Finding nodes along an axis can be automated if the axis is parallel to one of the reference coordinate system axes. In that case, all of the nodes will share two of their coordinates. For example, if one wants to add all of the nodes along a vertical axis parallel to the mesh Z-axis, the X and Y coordinates for all nodes will be identical. The user can measure the X and Y coordinates of a vertex at the end of the axis and then use those values to select the desired nodes, with commands such as:

```
### Store X and Y coords of LE bolt vertex
$ {VertID = 203}         # Use vertex at top of the axis
$ {Xvert = Vx(VertID)}   # Store X coord of vertex
$ {Yvert = Vy(VertID)}   # Store Y coord of vertex

### Create nodeset using these coordinates
nodeset 401 node with ((x_coord < {Xvert} + {tol}) and (x_coord > {Xvert} -
  {tol})) and ((y_coord < {Yvert} + {tol}) and (y_coord > {Yvert} - {tol}))
  in volume all
```

In this example, the vertex at one end of the desired axis-defining curve has ID 203, and the journal commands find its coordinates (using functions `Vx` and `Vy`) and store them as APREPRO variables. The final command finds all nodes with matching X and Y coordinates within a certain tolerance stored in the “`tol`” APREPRO variable and adds them to nodeset NS401.

A sample proprotor yoke mesh in Figure 2.27 shows a typical collection of joint nodesets. Nodeset 400 represents a boundary along the inner surface of the part, and can be used for static testing or disabled for gimbaled rotor analysis, and is created using the command “`node in surface ## ##`”. Nodesets 401 to 403 represent joint connection nodes where blade equipment can be connected and, because they are made of entire blade bolt volumes, were created using “`node in volume ## ##`”. Finally, the nodeset 404 represents hub bolts which can be connected to a gimbal joint. Since these node all lie along axes parallel to the Z-axis, the nodes were selected using the APREPRO logical operators described above for selecting nodes with common X and Y coordinates.

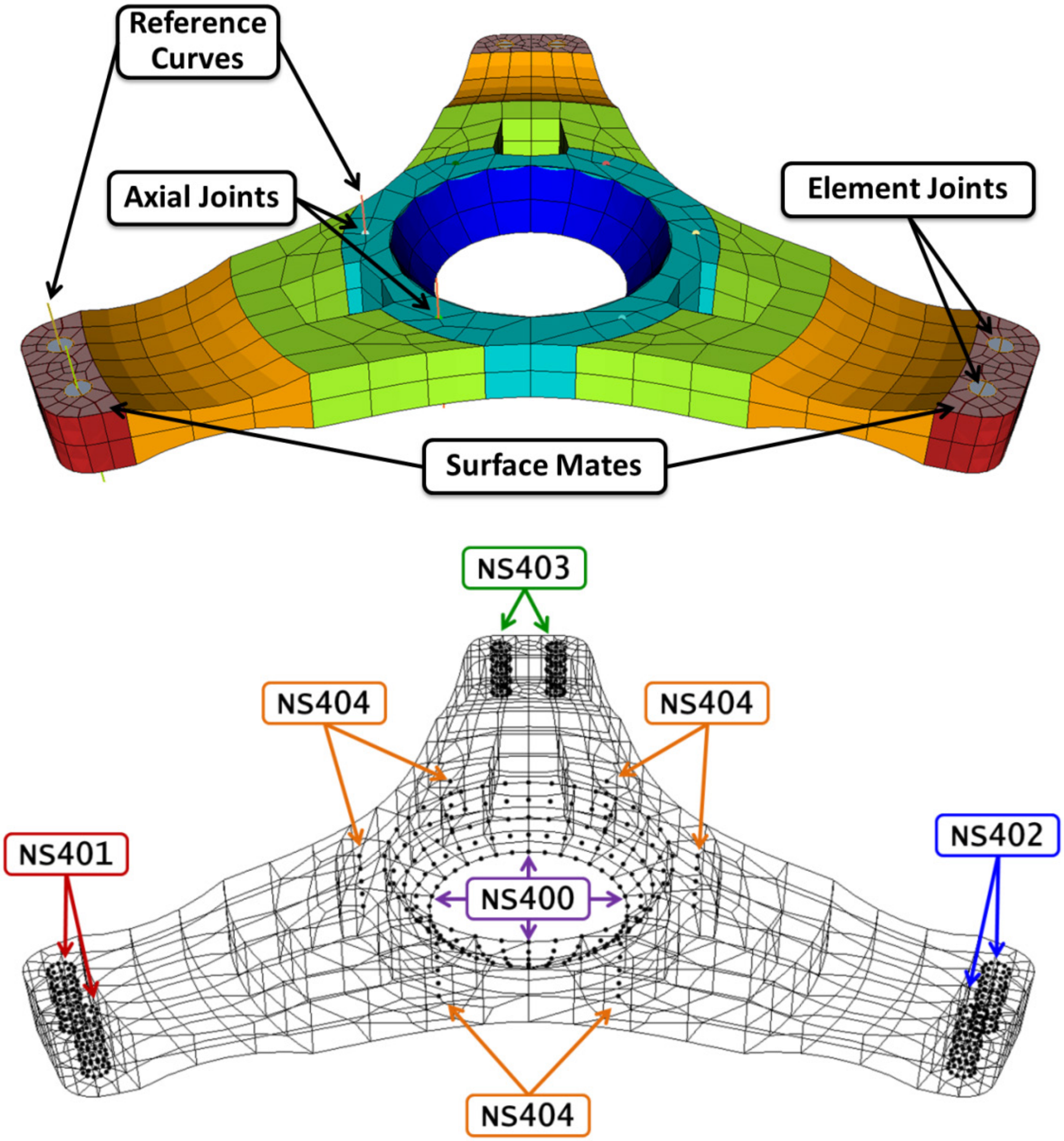


Figure 2.27 Sample proprotor yoke geometry demonstrating groups of nodes for joint connections (top) and their nodeset designations (bottom)

The 500 class nodesets (NS501 and up) are used to define cross-sections in a part for extraction of stress and strain information by the solver. Like class 500 sidesets, these nodesets are still under development.

The final three classes of nodesets are 700, 800, and 900. These are used to define blade cross-sections using four nodes: one on the leading edge, one on the trailing edge, one at the top of the thickness, and one at the bottom of the thickness. Although similar, each nodeset serves a different purpose. Class 700 nodesets, also known as "CN" for chord-normal, define deformation sections. Inside X3D, the relative deformed positions of the nodes are used to define a deformed airfoil section, the position of which is output in terms of deflections and rotations (u, v, w, θ_x, θ_y, θ_z). This allows equivalent beam-like deformations to be obtained from the 3-D blade mesh. Class 800 nodesets, also known as "SC" for structural cut, define cross-sections for the calculation of equivalent sectional blade loads. Class 900 nodesets, also known as "AE" for aerodynamic, identify aerodynamic cross-sections at the ends of the aerodynamic segments (SS900 class sidesets). Aero sections are used to communicate 2-D sectional airloads, when needed, to the structural solver in X3D. More information on interfacing class 700, 800, and 900 nodesets with X3D is provided in Section 2.6.1.1.

All three of these nodeset classes are defined the same way, and each consists of exactly four nodesets: class 700 is composed of NS701, NS702, NS703, and NS704; class 800 of NS801, NS802, NS803, and NS804; and class 900 of NS901, NS902, NS903, and NS904. Nodeset NS701 contains all deformation section nodes along the leading edge of the blade. NS702 defines nodes along the trailing edge. NS703 defines nodes along the upper surface, and NS704 defines nodes along the lower surface. An example of NS701 to NS704 for a tubular spar blade is shown in Figure 2.28.

Note that the motions of the deformation cross-sections at 3/4 chord are needed for unsteady aerodynamics calculations in X3D. Therefore, it is important that the chord line is defined precisely, with nodes selected at the leading edge (NS701) and trailing edge (NS702) picked accurately to lie exactly on either end of the chord line. The definition of the normal line (thickness-wise, defined between NS703 and NS704) is not critical to the solution, and need not be perpendicular to the chord line, despite the name. So long as the normal line lies in the plane of the cross-section it will suffice. That is to say, $\vec{c} \times \vec{n}$ is normal to the cross-section, which is generally true due to the radial nature of meshing in Cubit.

Likewise, NS801 to NS804 define the leading edge, trailing edge, upper surface, and bottom surface for SC sections. Unlike deformation sections, structural cut sections (NS801-804) are used to calculate equivalent sectional blade loads from 3-D strain data. Instead of being on the top and bottom skin of the blade and at the leading edge and trailing edge of the airfoil, they should instead be placed on the spar which carries the blade loads (Figure 2.29).

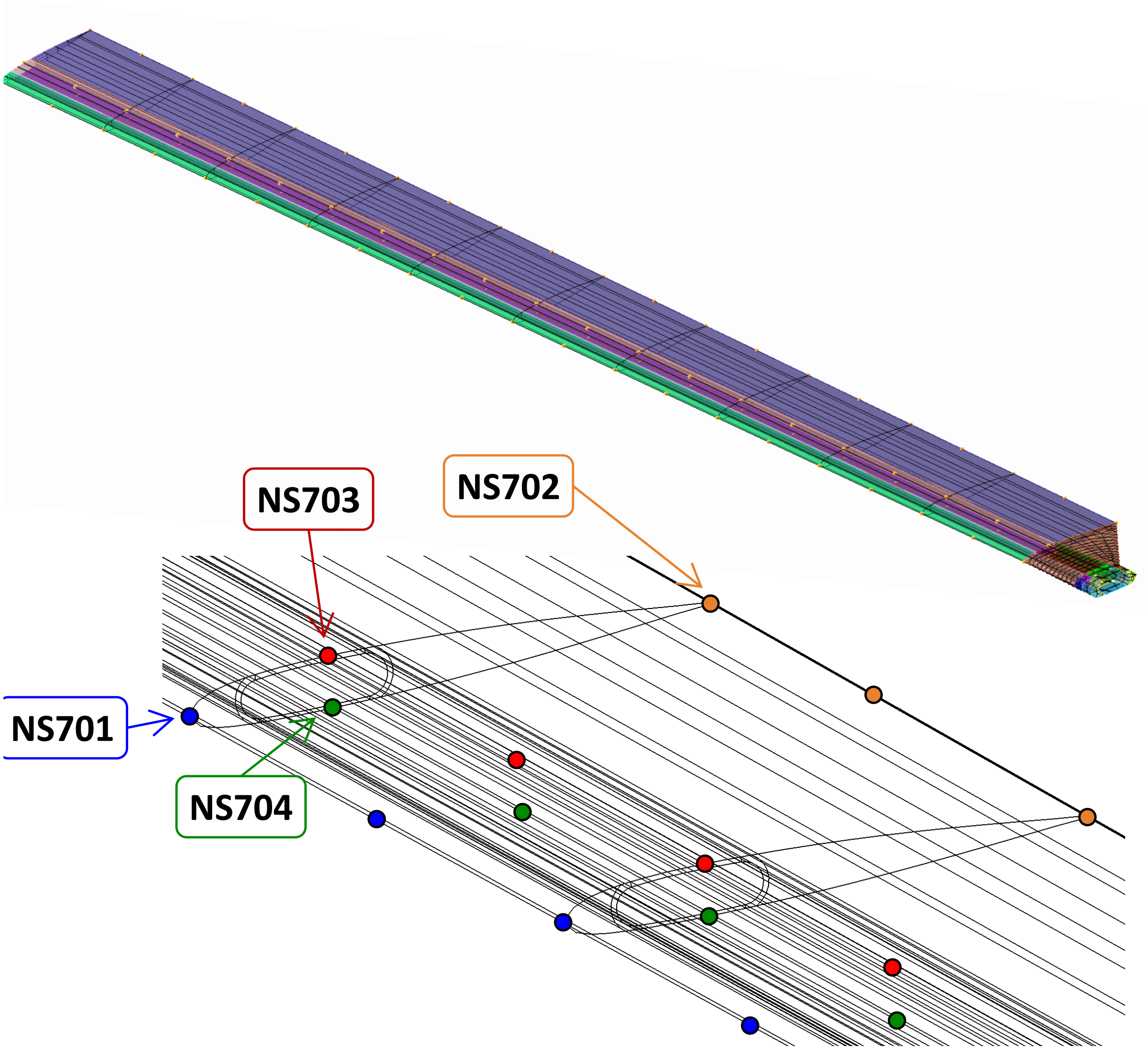


Figure 2.28 (Top) the mesh of a tubular spar blade with all sectional cut nodes (NS701 to NS704) highlighted and (bottom) a zoomed in mesh showing a few 700 series nodeset cross-sections: NS701 blue, NS702 orange, NS703 red, NS704 green

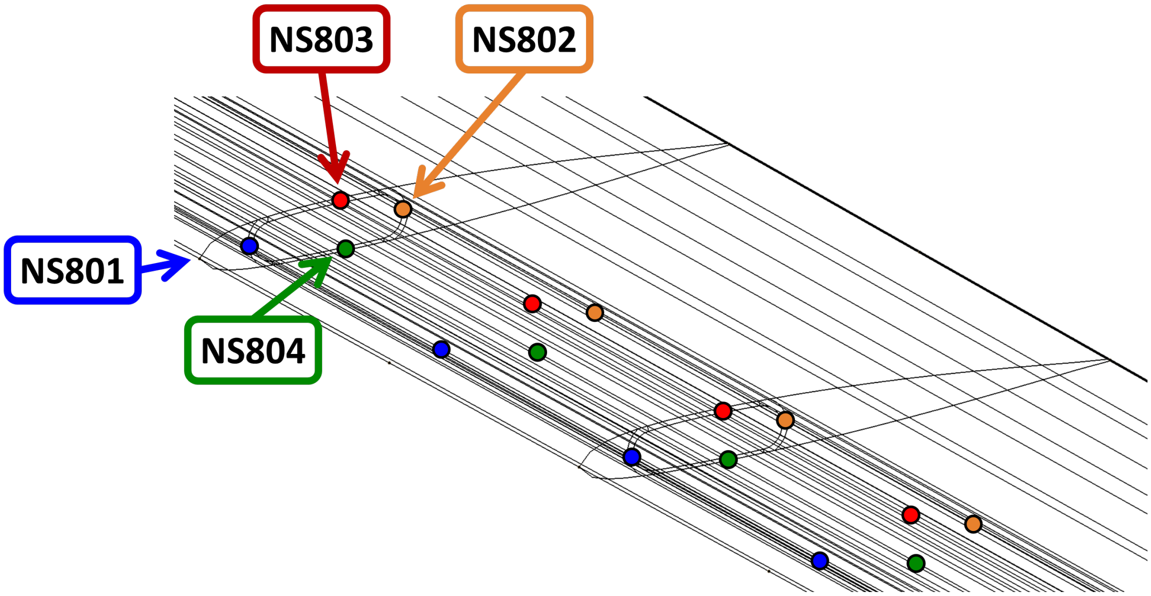


Figure 2.29 A zoomed in mesh showing 800 series structural cut cross-sections, nodes are chosen along the spar for sectional blade loads calculations: NS801 blue, NS802 orange, NS803 red, NS804 green

NS901 to NS904 define the leading edge, trailing edge, upper surface, and lower surface for the aerodynamic sections. These nodesets are used to determine sectional elastic deformations for the aerodynamic solver. The pitch angle of the aerodynamic section's chord line is used by the aerodynamics solver to calculate the angle of attack, thus it is important that the chord line is defined precisely, with nodes selected at the leading edge (NS901) and trailing edge (NS902) picked accurately to lie exactly on either end of the chord line. As with the deformation sections, the definition of the normal line (thickness-wise, defined between NS903 and NS904) is not critical to the solution, and need not be perpendicular to the chord line, despite the name.

All of these cross-section defining nodesets are optional, but will typically be used for blades. Depending on the needs of the designer, the 700, 800, and 900 class nodesets may be identical or different. For coarse meshes, it may be desired to have every node in the deformation section nodeset class 700. In this case, the process of selecting all nodes along the leading edge, trailing edge, upper surface, and lower surface can be automated. If the nodes fall along a curve associated with the geometry (e.g., a sharp trailing edge) the command "nodeset 70# node in curve ##" can be used to select the appropriate nodes. In many cases, however, there will not be a convenient curve. In these instances, there is a method to automate selection based on the elements containing the nodes.

This method for identifying nodes is best illustrated by an example. Consider the case of nodeset 701 for the tubular spar blade, seen above in Figure 2.28. As magnified in Figure 2.30 (top), this geometry has no curve defining its leading edge. However, it does have two elements at the leading edge (Figure 2.30, bottom), the shared edge of which contains the leading edge nodes. In order to select the proper nodes, the user needs to identify all elements which define the top of the leading edge and all elements which define the bottom of the leading edge; then, all nodes which are shared between them that also lie on the surface defining the leading edge can be added to the nodeset. In the case of the tubular spar blade example, the element faces in question are 22 and 27, and the leading edge surface is 72, as labeled in Figure 2.30.

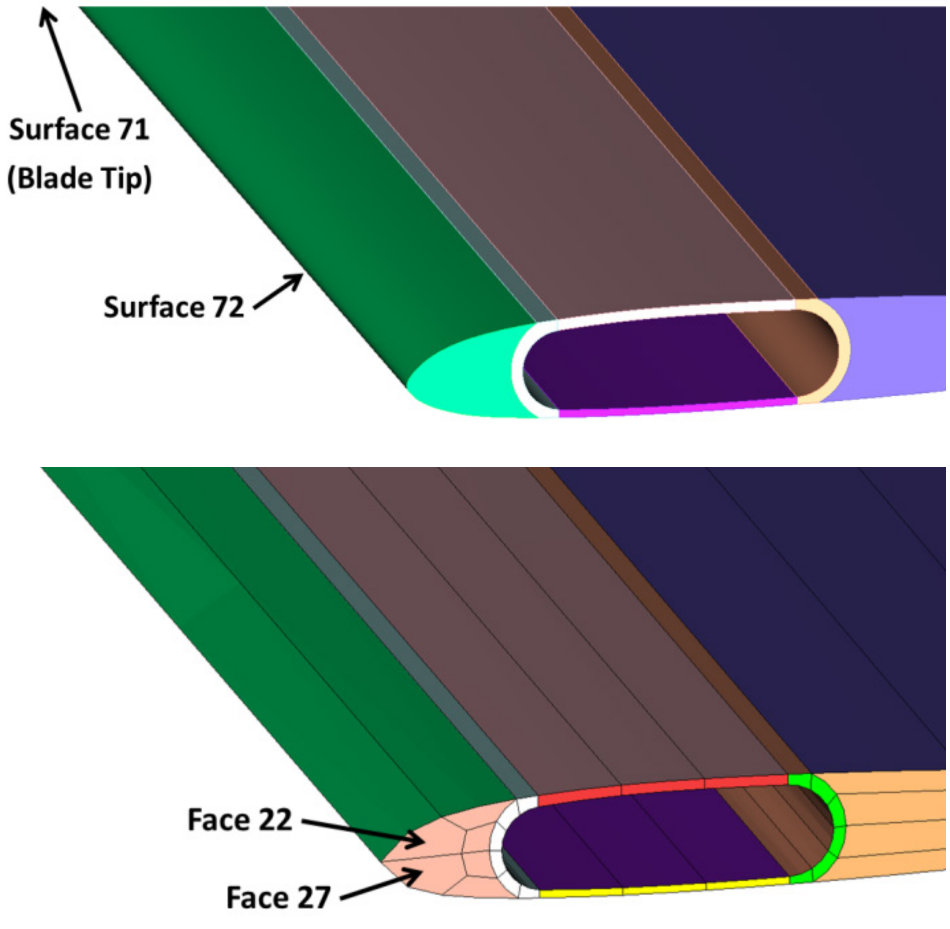


Figure 2.30 Example blade geometry (top) and mesh (bottom) for the creation of NS701, the leading edge of the sectional cut (CN); although there is no curve along the leading edge (top) there is an edge where the elements meet (bottom), and all nodes on this edge can be selected using propagation of the root faces

This can be accomplished by taking advantage of the “`group`” and “`propagate face`” commands within Cubit. The user can select an element face at the root of the blade (e.g., face 22 in the example above) and use the propagate face command to find all elements between it and the tip surface (surface 71, outside region shown in Figure 2.30). By doing this twice, the user will create two groups, one with all elements above the leading edge and another with all elements below. These groups can then be used to logically define the nodeset, as demonstrated in the commands:

```
### Create group to hold hexes temporarily
create group "El_Yes1"
create group "El_Yes2"

### LE of each deformation cross-section
$ {face1 = 22} {face2 = 27} {tgt_surf = 71} {out_surf = 72}
group El_Yes1 add hex propagate face {face1} target surface  {tgt_surf}
group El_Yes2 add hex propagate face {face2} target surface  {tgt_surf}
nodeset  701  node in hex in group El_Yes1 in hex in group El_Yes2 in
  surface {out_surf}
group all cleanout
```

If the 700 series nodeset includes all cross-sections within the mesh, it can be used as a basis for the selection of series 900. If the 900 series nodesets are identical to the 700 series, the commands are simply “`nodeset 901 node in nodeset 701`”, with similar entries for the other three NS900 class nodesets. The case may arise, however, where one of the high class nodesets consists of only a portion of those in NS700. For example, NS900 is used to define the ends of aero segments. For the TRAM blade example shown in Figure 2.31, the aero segments contain multiple mesh cross-sections.

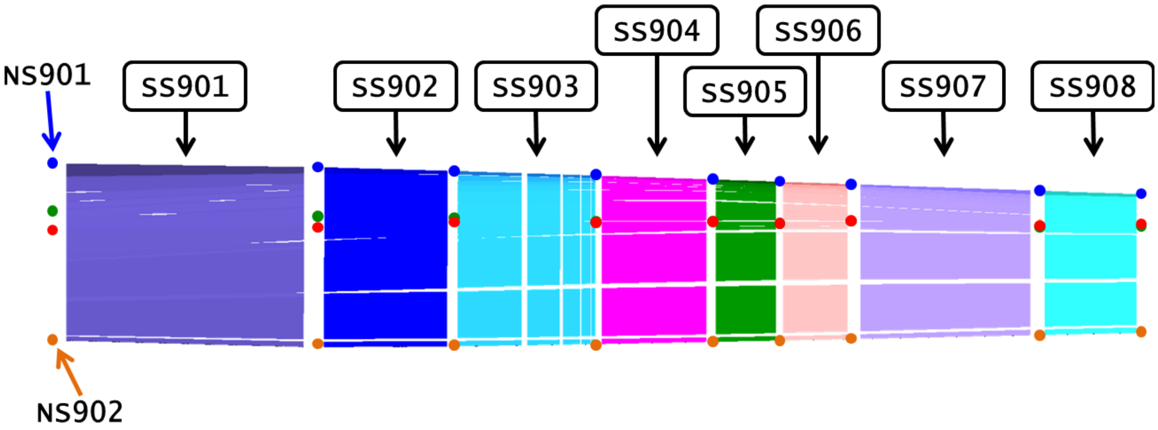


Figure 2.31 Aerodynamic segment sidesets and aerodynamic section nodesets for the TRAM proprotor blade

If nodesets 701 to 704 contain all of the leading edge, trailing edge, upper surface, and lower surface nodes, then nodeset 901 to 904 will be the subset of nodes which fall at the borders of the aerodynamic segments, as marked in Figure 2.31. The nodes at the blade root and tip will have be selected manually, or using a command such as "`nodeset 901 node in 701 with x_coord == {x_tip}`" where `x_tip` is equal to the radial location at the tip of the blade. The fact that the internal nodes all fall on the interface between two adjacent sidesets can be exploited for automatic selection using an APREPRO loop function, to select only nodes shared by adjacent aero segments:

```
${jj = 1} # Initialize loop index
${N_AerSeg_m1 = N_AerSeg - 1}
${Loop(N_AerSeg_m1)}  # Loop over the aerosegments
  nodeset  901 node in nodeset 701 in face in sideset {900 + jj} in face in
   sideset {900 + jj + 1}
  nodeset  902 node in nodeset 702 in face in sideset {900 + jj} in face in
   sideset {900 + jj + 1}
  nodeset  903 node in nodeset 703 in face in sideset {900 + jj} in face in
   sideset {900 + jj + 1}
  nodeset  904 node in nodeset 704 in face in sideset {900 + jj} in face in
   sideset {900 + jj + 1}
  ${jj++}
${EndLoop}
```

In this example, `N_AerSeg` is the total number of aero segments, and `N_AerSeg_m1` is the total number minus one. It must be defined separately because APREPRO does not allow algebra within the arguments of the `loop` command.

2.5.3.6 Examine and Export the Mesh

Once the mesh has been completed, the model designer must inspect it closely to ensure that all blocks, sidesets, and nodesets are defined as intended using commands such as "`draw block all`" and "`highlight node in nodeset 701 to 704`". The number of elements and nodes should be checked with the commands "`list node ids`" and "`list element ids`" or "`list model`". The command "`node visibility on`" can be used to confirm there are no misplaced internal nodes of Hex27 elements. Extra care should be taken when examining concave surfaces for internal Hex27 nodes that penetrate the boundaries of their parent element. This is a known problem in Cubit and is discussed in greater detail in Section 2.5.4.1: Misplaced Internal Nodes on Concave Volumes.

Once the user is confident the mesh is complete and accurate, it is exported as an I-DEAS Universal .unv file for processing. The command "`export Ideas "file_name.unv"`" will save the file. If exporting using the GUI, the "Export Using Cubit IDs" flag should be disabled to maintain consistent node numbering, discussed further in Section 2.6.1.2: Natural and Global Node and Face IDs. It is recommended that each export should be given a file name with a unique version tag appended at the end. The export command can be commented out and saved at the end of the journal file with notes, such as the mesh size, helping the user keep track of different mesh versions.

2.5.4 Advanced Concepts in Meshing

2.5.4.1 Misplaced Internal Nodes on Concave Volumes

The X3D solver uses 27 node solid hexahedral elements (Hex27) for FEM analysis of flexible structures. Cubit is capable of creating Hex27 meshes but its native element is an eight node hexahedron (Hex8). Cubit starts by creating a Hex8 mesh, the elements of which are, upon completion of the mesh, converted into Hex27 elements by adding additional internal nodes. For many meshes, this process works correctly. However, there is a known issue within Cubit when converting Hex8 to Hex27 for concave volumes with multiple elements through the thickness which causes internal nodes to be created outside the bounds of the geometry.

The circumstances leading to this error are best illustrated using an example. Consider two identical volumes with a crescent shape as depicted in Figure 2.32, one red (top) and one blue (bottom). The red volume will be meshed with Hex8 elements and the blue with Hex27 (note: when viewed in 2-D, Hex8 and Hex27 appear identical to Quad4 and Quad9 elements, respectively).

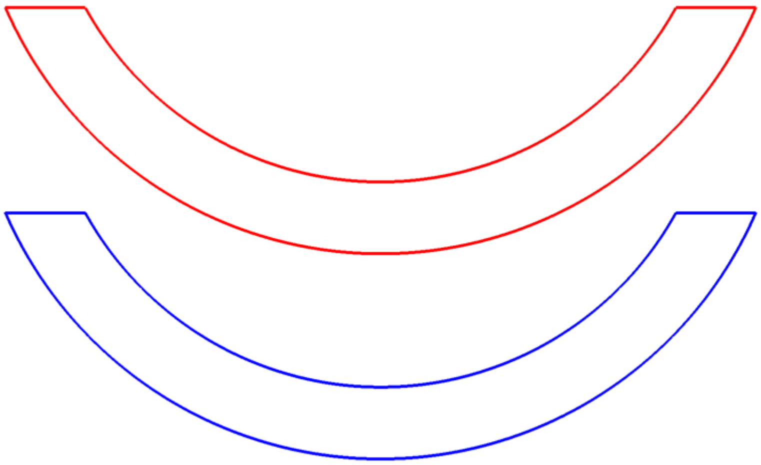


Figure 2.32 Two crescent shape extruded volumes to demonstrate Hex27 meshing of concave surfaces

When using a single element across the thickness of the volume, as shown in Figure 2.33, all of the nodes of the Hex8 mesh follow the geometry. When converting the Hex8's to Hex27's, Cubit begins with the corners of the Hex8's and then adds center-edge nodes. These center-edge nodes can be seen to follow the geometry of the curvature, as desired (called-out by green arrows in Figure 2.33). Finally, Cubit places an interior node at the center of the element; its position in the center of the curved volume's thickness is based on the corner and edge nodes. As expected, the Hex27 captures the geometry better than the Hex8, even in this Cubit rendering with straight edges (as opposed to X3D, in which the element edges would be curved due to the use of full second order interpolation functions, allowing the entire crescent to be captured exactly with a single element).

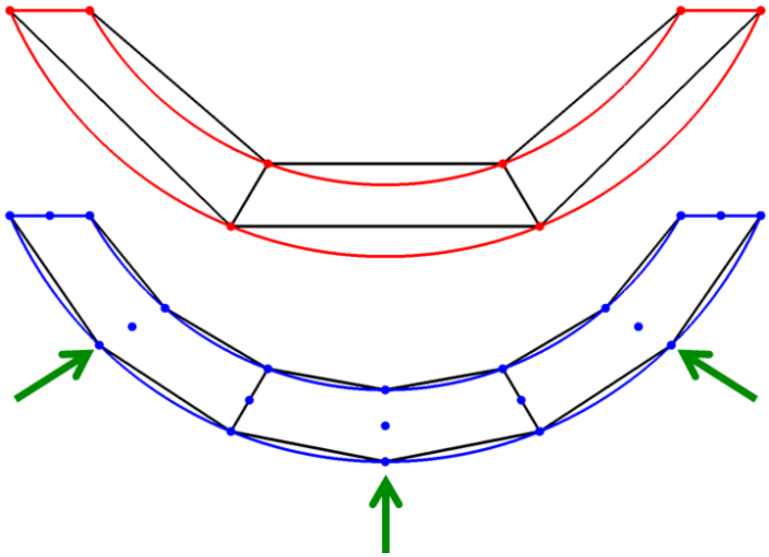


Figure 2.33 Meshing a concave volume with Hex8 (top, red) and Hex27 (bottom, blue) elements; one element through the thickness

Complications arise when additional elements are added through the thickness of the curved volume. Figure 2.34 depicts meshing with two elements across the thickness of the volume. In the Hex8 mesh, the nodes on the inside and outside of the circumference follow their curve's geometry, and the central node falls halfway between them, adequately representing the volume. When converting to Hex27 elements, Cubit once again begins with the Hex8 mesh and adds center-edge nodes. On the internal and external surfaces of the crescent, Cubit succeeds in linking the surface edges of the Hex27 mesh to the curved geometry, so the nodes successfully follow the geometry. On the interior of the crescent, however, Cubit links the internal element edges (called-out by red arrows in Figure 2.34) to the internal Hex8 element edges; it does not re-position them based on the surface geometry. This leads to very thick elements on the outside of the curve, and very thin elements on the inside, un undesirable artificial distortion.

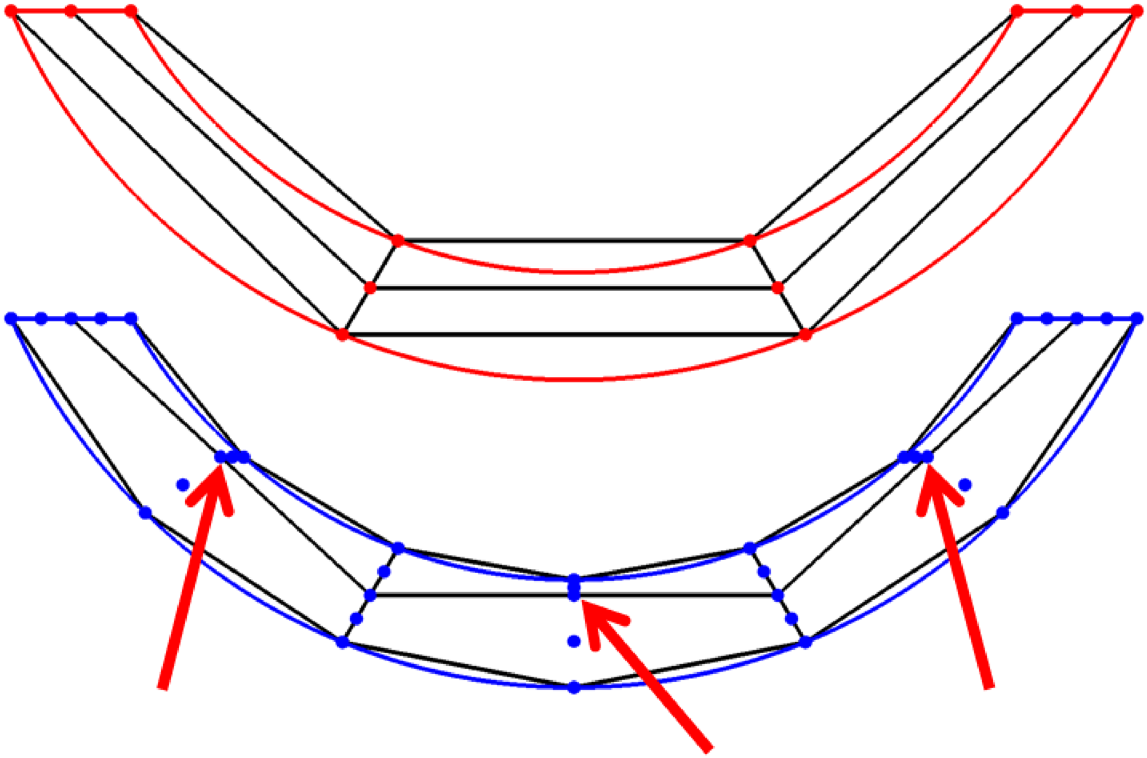


Figure 2.34 Meshing concave volumes with two elements through the thickness; Hex27 interior edges do not respond to the surface geometry

Taken further to five elements through the thickness of the volume, as depicted in Figure 2.35, the situation is aggravated. The internal Hex27 elements are all identical to the Hex8, but with extra nodes placed at the midpoints of the linear edges. Only the outside edge of the outermost element and inside edge of the innermost element are bound by Cubit to the surface of the crescent volume, so only the center-edge nodes of these curves follow the surface. This creates spurious elements on the concave portion of the volume, with center-edge and interior nodes that lie outside the boundaries of the innermost element (called-out by red arrows in Figure 2.35). These elements have negative Jacobians, though this fact is missed by the Hex8 Jacobian metric in Cubit (Section 2.5.2.2), highlighting the importance of visual mesh inspection.

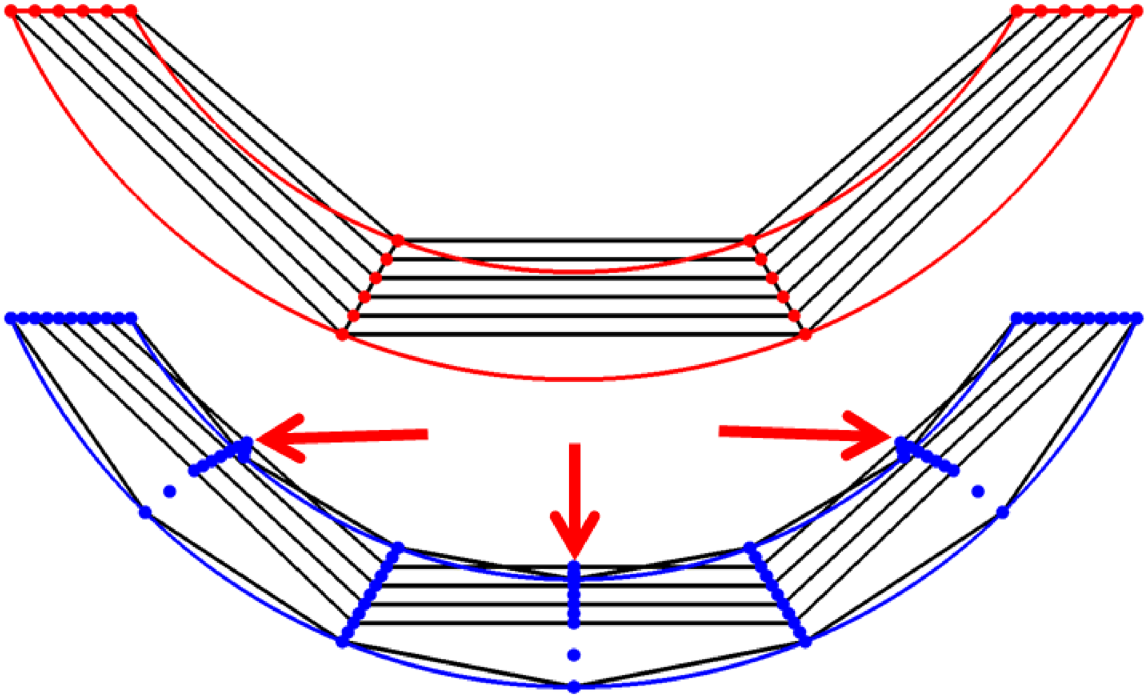


Figure 2.35 Meshing concave volumes with five elements through the thickness leads to interior nodes penetrating the surface of the innermost element

The interior node placement issue extends into the third dimension, as shown in Figure 2.36. Interior nodes can be seen penetrating through the interior surface of the geometry on the Hex27 mesh.

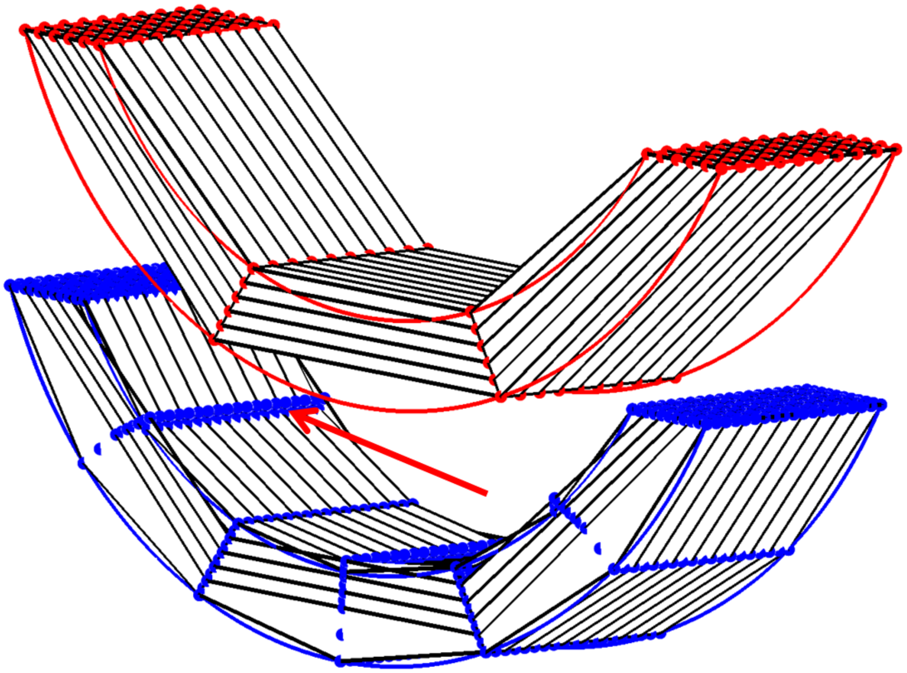


Figure 2.36 Viewed in 3 D, meshing concave volumes with five elements through the thickness leads to interior nodes penetrating the surface of the innermost element

There is no way of automatically detecting interior nodes that penetrate through the bounds of their Hex27 parent element in Cubit. X3D can detect the issue, which is reported as a negative volume Hex27 element error, though smaller excursions may not be detected.

This problem is known by the Cubit development team but there is no current solution that allows the interior nodes to recognize the surface geometry. Using the command "`node constraint off`" before the conversion to Hex27 will command all interior nodes to ignore the surface geometry, so all center-edge nodes will follow the original Hex8 edges. While this can keep elements from having misplaced interior nodes, it will also result in a worse capture of the geometry, one of the benefits of using Hex27 elements in the first place.

Instead, this issue should be avoided where possible. Improper interior node placement only becomes a problem when multiple elements are used to mesh concave geometry. Even then, poorly placed interior nodes only exceed the boundaries of their elements when the elements are thin. Thus, the issue can be avoided entirely by using only a single element through the thickness of any thin, concave volume.

In the event that thin, concave bodies must be meshed with multiple elements, the only remaining option is to manually reposition the nodes. This can be done with the command "`node ### ### ### move x # y # z #`", which will move a group of nodes as listed in 3-D space. This process can be labor intensive and time consuming for large meshes. Determining how far to move nodes can be made easier by utilizing the command "`measure between node ### node ###`" to estimate the shift required.

2.5.4.2 Tolerance Issues

One known problem when importing CAD models created using CATIA into Cubit is that CATIA has a lower internal tolerance than Cubit. Splines and curved surfaces imported from CATIA into Cubit may look identical but in fact be slightly different. This can lead to difficulty in imprinting and merging volumes, leading to un-meshable geometry.

As mentioned earlier (Section 2.3.2: Assemblies, Parts, and Bodies), CAD parts should contain multiple bodies for internal structures made out of different materials. When imported into Cubit, these will each be assigned a unique volume. For simple models this should not be a problem. For complex rotor blades, though, there may be cases where the boundaries between two volumes will be made up of curved surfaces that are identical in CATIA but upon import into Cubit are defined slightly different, due to the increased precision.

Using the heal geometry tools built into Cubit can solve some of these problems, but it can also often exacerbate them. If a problem arises, it can be useful to import the geometry with and without healing then compare the two Cubit .cub files to identify differences. The commands described in Section 2.5.3.2: Clean-up and Decompose Geometry can be used to search for small curves and loose vertices, allowing the user to identify which imported model works best. Errors with differences in geometry can sometimes be handled using forced merges, as described in the next section.

2.5.4.3 Troubleshooting Imprint/Merge Problems

With complicated geometry, a simple "`imprint all`" / "`merge all`" command pair may not be sufficient to merge the geometry. This may not become evident until meshing problems arise. To diagnose meshing problems, the "`is_merged`" property of the suspected problematic geometry in Cubit can be examined. If automatic imprint and merge fail, the best solution may be to manually imprint and merge volumes and surfaces piece by piece. Imprinting and merging issues are often caused by underlying geometry problems, e.g., small surfaces or spurious vertices, so if an error occurs it is important to check for these flaws and perform the necessary clean-up (Section 2.5.3.2: Clean-up and Decompose Geometry).

When imprinting fails, it may be due to different tolerances between the CAD model and the geometry as imported into Cubit. If this is the case, the tolerance of the imprint can be increased. This is done with the command "`imprint tolerant volume all`", which allows Cubit to imprint geometries that have differences too small to catch on the screen. However, it may create other problems by imprinting geometry that does not match, so clean-up of small geometry may need to be performed again (Section 2.5.3.2.1).

Instead of using “`imprint all`” or “`imprint tolerant all`”, the user can instead act part by part and attempt to imprint the geometry more selectively. The geometry should be treated section by section, for example from the root toward the tip and from the leading edge toward the trailing edge. Targeted imprint commands such as “`imprint volume ## ##`” and the more specific “`imprint curve ## on surface ##`” can ensure the imprinting is behaving as expected. If the imprinting command fails, other geometry modification commands may be necessary, for example splitting curves manually using the “`split curve ## at vertex ##`” command.

If merging still fails after checking over the imprinting, manual merging may be necessary. Merging specific geometric identities (“`merge surface ## ##`”) may be necessary. However, the merge command only works if the targeted geometric entities are identical within Cubit. There may be circumstances where geometry that the user knows should be identical and merged may not be recognized by Cubit. In this case the command “`merge surface ## ## force`” may be able to overcome whatever internal differences Cubit detects. However, this should only be used if all else fails. Note that all of the imprint and merge commands above can be used with all of the typical geometric entities (volume, surface, curve, vertex).

2.5.4.4 Meshing “Triangular” Volumes

The most difficult shapes to mesh are those that form a triangular profile. For example, consider a blade whose thickness decreases along its radius. At its tip it goes to zero thickness, which is to say the top and bottom surfaces of the blades meet. This forms a “triangular” profile when viewed from the trailing edge, with the blade having a thickness at the root but none at the tip. Such shapes are not amenable to mapping, and can only be swept across a “non-triangular” direction, in this example from the top of the blade to the bottom. However, depending on the structure inside the blade, top to bottom meshing may not be possible.

In such a case, there are two potential solutions. The first is to attempt to use webcutting to separate the problematic region of the blade, whether it is internal or at the tip. Then, instead of starting meshing surfaces at an inboard cross-section, the difficult section can be meshed across its non-triangular direction (e.g., from top-to-bottom) using `sweep`. The linking surface meshes Cubit creates on the cross-sections at the ends of the troubled region will be mapped by the `sweep` algorithm. If the meshes on these cross-section surfaces are acceptable, they can then be swept inboard or outboard to complete the blade mesh.

Although it may work, sweeping across the thickness in this manner can lead to poor quality. An example of this is presented in Figure 2.37. Although not exactly triangular, the trapezoidal, orange colored region at the trailing edge of the blade (called-out with an arrow) was swept from its top surface to its bottom, instead of radially. Evidence of this is seen in the top surface mesh, which is paved. Sweeping across the thickness leads to a mapped surface at the root end of the volume. This surface mesh has very poor quality; although only three elements can be seen in at the trailing edge root in Figure 2.37, there is a fourth element with almost zero thickness at the tip of the call-out arrow. In this case, a better mesh could have been obtained by using fewer elements to mesh the trailing edge wall of the tubular spar.

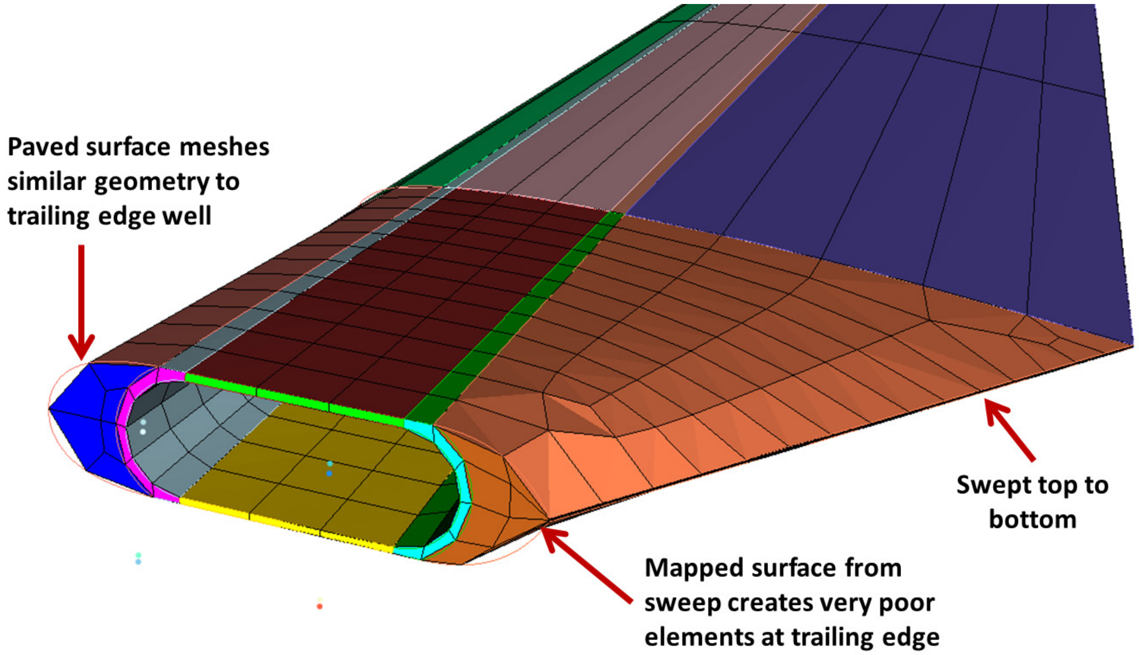

Paved surface meshes similar geometry to trailing edge well
Swept top to bottom
Mapped surface from sweep creates very poor elements at trailing edge

Figure 2.37 Example of a volume meshed across its thickness (orange colored area), resulting in poor mesh quality on the mapped root surface

The other potential solution is to simply cut off the tip of a triangular volume, thereby creating a trapezoidal volume which can be swept from root to tip. In the example of a blade whose thickness goes to zero at the tip, this would take the form of simply webcutting the tip of the blade using a vertical plane positioned just inboard of the radius of the rotor. Then, the tip volumes can be deleted. Since the very tip of a rotor blade is not structurally significant, this may be a good solution, though it can affect the mass, centrifugal force, and CFD aerodynamics if too much is removed. In the case where internal structures form triangular regions, say a spar web tapering to zero thickness, this method may create a viable mesh but could cause other problems, such as a non-physical void in the blade structure.

2.6 Structural Analysis Model

Once the structural analysis representation (SAR) showing the relationship between all of the parts in the rotor system has been completed and the flex parts have been meshed appropriately, all parts are assembled into the final structural analysis model (SAM). The SAM forms a complete, X3D-ready, model of the rotor system for input to the solver. This step combines the part definitions from the structural analysis representation, the appropriately processed individual flex part mesh data files, the joint definitions, and material properties. It was an objective of this work to standardize the mesh and joint file formats for 3-D FEA/multibody analysis of rotor blades, as well as creating tools to generate those files; this section presents the resulting file standards and describes the data file creation utilities.

2.6.1 Mesh Processing

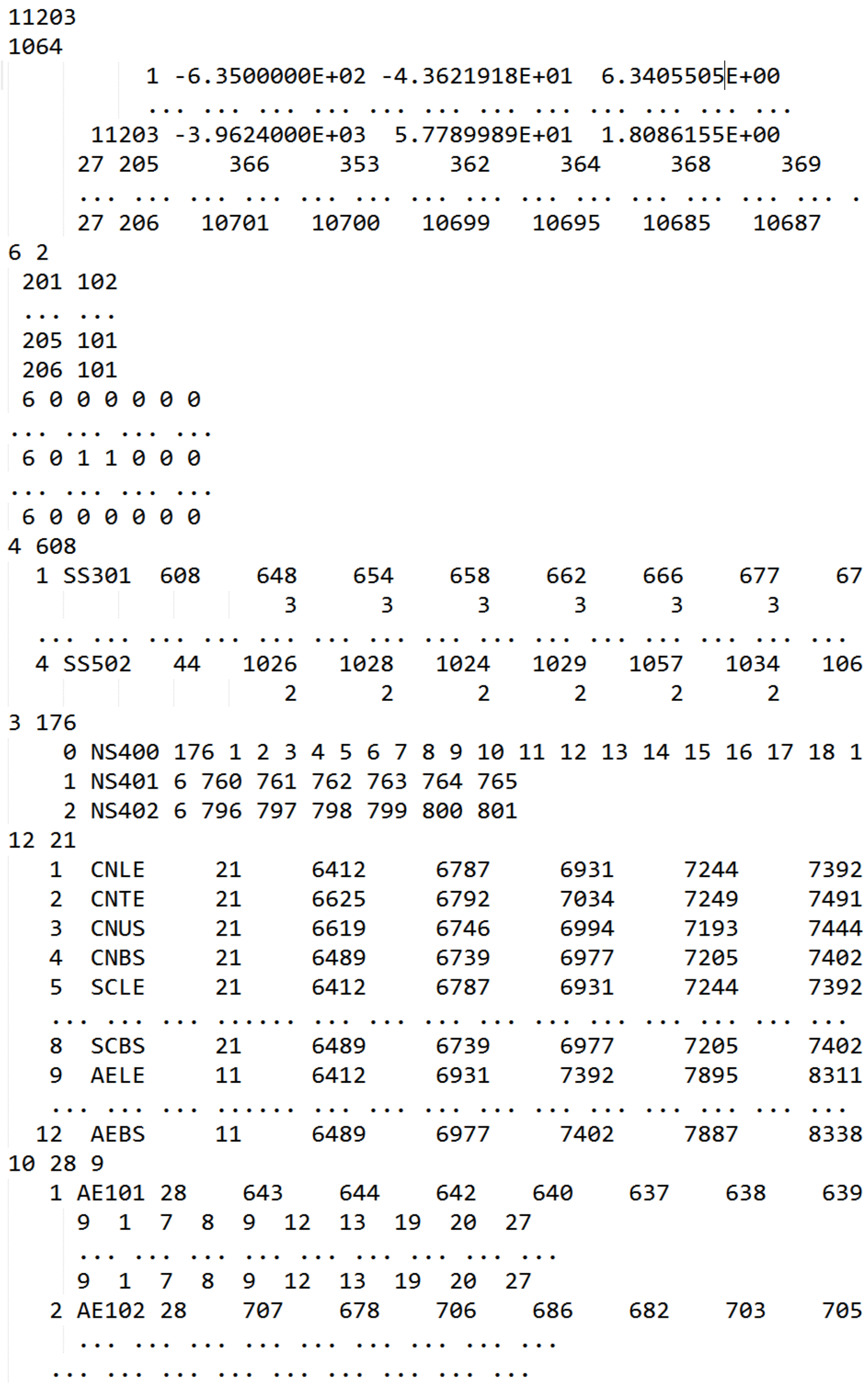
Flexible part meshes exported by Cubit in the I-DEAS Universal (.unv) file format need to be re-processed into the X3D mesh data file (.dat) standard for analysis. Each unique flexible part needs a .dat mesh file. Developed as part of this research effort, the `MeshProc.m` MATLAB function performs the necessary conversion. In addition to converting the format of the file, the `MeshProc.m` function can also reorder global node and element IDs, reducing model bandwidth and improving solution efficiency. This section presents the X3D mesh data file .dat standard and discusses the `MeshProc.m` conversion tool created as part of this work.

2.6.1.1 X3D Mesh Data File Format (.dat)

Each flexible component within the rotor needs its own mesh data file for analysis in X3D. These are ASCII files that contain all necessary mesh information. They do not contain material property data, which is provided in the SAM input file, but do identify the material blocks for each element. A ".dat" extension is typically used to identify mesh files, but any ASCII file can be ready by X3D.

Figure 2.38 shows a sample mesh data file for a rotor blade, and is intended to illustrate the structure and function of the standard. There are nine sections in a mesh data file, which are identified in Figure 2.39 by a number and the color of their annotations. This section describes the contents and size of each section, using variables for line counts where mesh dependent. Values in parentheses refer to those for the sample mesh in Figure 2.38 and Figure 2.39. This section refers to blocks, nodesets, and sidesets, more information on which can be found in Section 2.5.3.5.

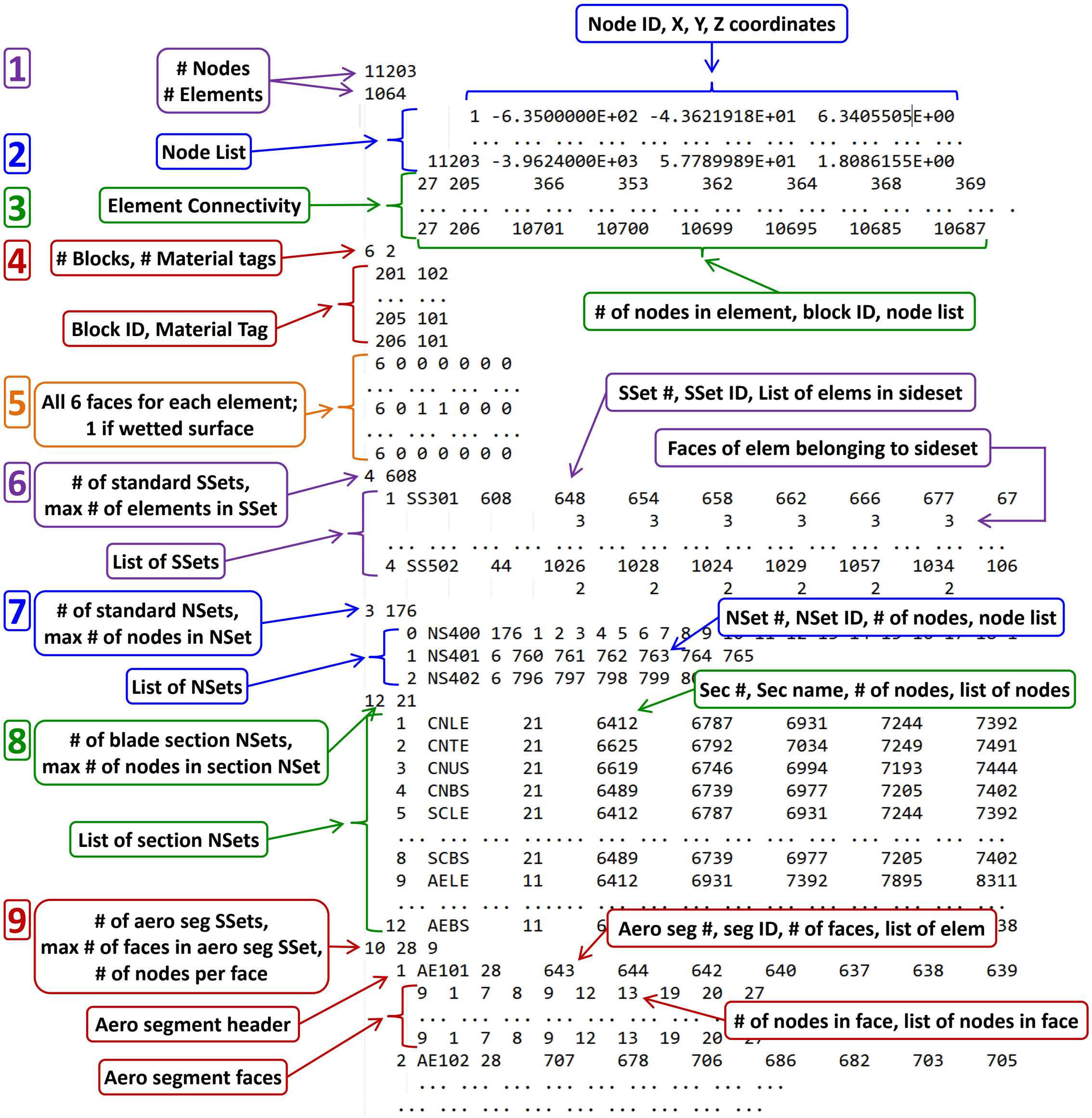
X3D ignores indentation and extra spacing, however, the indentation shown in the sample mesh data file is recommended for readability, and is generated automatically by the `MeshProc.m` script. Most fields in the mesh data file contain unsigned integers, with several strings where noted, except for the node coordinates in section 2 which are floating point numbers.



```
11203
1064
          1 -6.3500000E+02 -4.3621918E+01  6.3405505E+00
          ... ... ... ... ... ... ... ... ... ... ... ...
      11203 -3.9624000E+03  5.7789989E+01  1.8086155E+00
     27 205     366     353     362     364     368     369
     ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... .
     27 206   10701   10700   10699   10695   10685   10687
6 2
 201 102
 ... ...
 205 101
 206 101
 6 0 0 0 0 0 0
... ... ... ...
 6 0 1 1 0 0 0
... ... ... ...
 6 0 0 0 0 0 0
4 608
  1 SS301  608    648    654    658    662    666    677    67
                     3      3      3      3      3      3

  ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
  4 SS502   44   1026   1028   1024   1029   1057   1034   106
                    2      2      2      2      2      2
3 176
    0 NS400 176 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 1
    1 NS401 6 760 761 762 763 764 765
    2 NS402 6 796 797 798 799 800 801
12 21
   1  CNLE     21      6412     6787     6931     7244     7392
   2  CNTE     21      6625     6792     7034     7249     7491
   3  CNUS     21      6619     6746     6994     7193     7444
   4  CNBS     21      6489     6739     6977     7205     7402
   5  SCLE     21      6412     6787     6931     7244     7392
   ... ... ... ...... ... ... ... ... ... ... ... ... ... ...
   8  SCBS     21      6489     6739     6977     7205     7402
   9  AELE     11      6412     6931     7392     7895     8311
   ... ... ... ...... ... ... ... ... ... ... ... ... ... ...
  12  AEBS     11      6489     6977     7402     7887     8338
10 28 9
   1 AE101 28    643    644    642    640    637    638    639
     9  1  7  8  9  12  13  19  20  27
     ... ... ... ... ... ... ... ... ... ...
     9  1  7  8  9  12  13  19  20  27
   2 AE102 28    707    678    706    686    682    703    705
     ... ... ... ... ... ... ... ... ...
   ... ... ... ... ... ... ... ... ...
```

Figure 2.38 An abridged .dat mesh data file for a rotor blade to illustrate the file standard; ellipses (…) indicate skipped lines; lines are truncated at 60 characters for readability



```
11203
1064
       1 -6.3500000E+02 -4.3621918E+01  6.3405505E+00
       ... ... ... ... ... ... ... ... ... ... ... ...
   11203 -3.9624000E+03  5.7789989E+01  1.8086155E+00
  27 205      366      353      362      364      368      369
  ... ... ... ... ... ... ... ... ... ... ... ... ... ... .
  27 206    10701    10700    10699    10695    10685    10687
6 2
 201 102
 ... ...
 205 101
 206 101
 6 0 0 0 0 0 0
... ... ... ...
 6 0 1 1 0 0 0
... ... ... ...
 6 0 0 0 0 0 0
4 608
  1 SS301   608     648     654     658     662     666     677     67
                      3       3       3       3       3       3

  ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
  4 SS502    44    1026    1028    1024    1029    1057    1034    106
                      2       2       2       2       2       2
3 176
    0 NS400 176 1 2 3 4 5 6 7 8 9
    1 NS401 6 760 761 762 763 764 765
    2 NS402 6 796 797 798 799 8
12 21
   1   CNLE      21      6412      6787      6931      7244      7392
   2   CNTE      21      6625      6792      7034      7249      7491
   3   CNUS      21      6619      6746      6994      7193      7444
   4   CNBS      21      6489      6739      6977      7205      7402
   5   SCLE      21      6412      6787      6931      7244      7392
   ... ... ... ...... ... ... ... ... ... ... ... ... ... ...
   8   SCBS      21      6489      6739      6977      7205      7402
   9   AELE      11      6412      6931      7392      7895      8311
   ... ... ... ...... ... ... ... ... ... ... ... ... ... ...
  12   AEBS      11      6                                       38
10 28 9
   1 AE101 28     643     644     642     640     637     638     639
    9   1   7   8   9   12   13   19   20   27
    ... ... ... ... ... ... ... ...
    9   1   7   8   9   12   13   19   20
   2 AE102 28     707     678     706     686     682     703     705
    ... ... ... ... ... ... ... ... ...
   ... ... ... ... ... ... ... ... ...
```

1
# Nodes
# Elements
Node ID, X, Y, Z coordinates
2
Node List
3
Element Connectivity
# of nodes in element, block ID, node list
4
# Blocks, # Material tags
Block ID, Material Tag
5
All 6 faces for each element; 1 if wetted surface
SSet #, SSet ID, List of elems in sideset
Faces of elem belonging to sideset
6
# of standard SSets, max # of elements in SSet
List of SSets
7
# of standard NSets, max # of nodes in NSet
NSet #, NSet ID, # of nodes, node list
List of NSets
Sec #, Sec name, # of nodes, list of nodes
8
# of blade section NSets, max # of nodes in section NSet
List of section NSets
9
# of aero seg SSets, max # of faces in aero seg SSet, # of nodes per face
Aero seg #, seg ID, # of faces, list of elem
Aero segment header
# of nodes in face, list of nodes in face
Aero segment faces

Figure 2.39 An abridged .dat mesh data identical to that in Figure 2.38, with annotations illustrating the nine different sections of the file standard (color coded) and the contents of each line

]

Section 1: Mesh Size; Number of lines: 2

The first section consists of two lines, each containing one integer. The first line provides the number of nodes in the mesh, `N_Nd` (11203), and the second the number of elements, `N_El` (1064).

Section 2: Node Definitions; Number of lines: `N_Nd`

The second section contains a list of nodes. Each node gets its own line, making the section `N_Nd` (11203) lines long. Each line must specify the mesh global node ID, followed by its three spatial coordinates X, Y, and Z in the part's local coordinate frame. This local part coordinate frame is defined when creating the original CAD model, and its position in the global rotor system coordinate frame is defined in the structural analysis representation (Section 2.4). The default units in X3D are meters but any unit of length may be used, with an appropriate scaling factor later provided through the X3D SAM input file.

Section 3: Element Definitions; Number of lines: `N_El`

The third section contains a list of elements and defines mesh connectivity. Each element gets its own line, making the section `N_El` (1064) lines long. Each line describes one element, the first corresponding to element 1, the second to element 2, and so on. The first value in each line is the number of nodes in the element, `El_Size` (27). Currently, only Hex27 meshes are supported by X3D. The second value gives the ID of the block to which the element belongs (for more information on blocks see Section 2.5.3.5.1), which range sequentially from 201 up to 200 + `N_Blocks`, the total number of blocks in the part.

The remaining `El_Size` (27) values list the global node IDs which define the element. The global node IDs are listed in order of their natural node IDs within the element, which is to say the first global node ID listed corresponds to node #1 in the element, and the second is element node #2, and so on. More discussion of node ordering is provided in Section 2.6.1.2: Natural and Global Node and Face IDs.

Section 4: Block Material Definitions; Number of lines: `1 + N_Blocks`

The fourth section lists the blocks and their associated material tags. Material tags are three digit IDs which link a block in the mesh to a set of material properties in the X3D SAM input file. The association of blocks with materials is defined during meshing (Section 2.5.3.5).

The first line of the section contains two integers, the number of blocks in the part, `N_Blocks` (6), followed by the number of material tags, `N_Mat` (2). This is followed by `N_Blocks` lines, each of which consists of two integers containing a block ID followed by its material tag. There has to be at least one material tag, 101, with additional tags indexing from the first (102, 103, …). Material tags cannot skip values (no 103 without a 102) though they may be assigned to blocks in any order (in the sample provided, 102 is assigned to a block before 101). Every block assigned in the element connectivity section must be listed, and every block must have a material tag. Multiple blocks may share a single material tag, and there may be no more material tags than blocks.

Section 5: Aerodynamic Interface Faces; Number of lines: `N_El`

The fifth section identifies which element faces lie on wetted surfaces, i.e. are exposed to the flow. It is used by the CFD interface within X3D, and uses data from the aerodynamic interface sideset SS301. This section is `N_El` lines long (1064), with one line for every element. Each line starts with the number of faces for the element (6). The number of faces is followed by one value for each face of the element in their natural ID order (Section 2.6.1.2). If the element face is exposed to the flow around the rotor, it is assigned a value of 1; if not it is left as zero. Typically, most elements will have the default value of "`6 0 0 0 0 0 0`".

Section 6: Standard Sideset Definitions; Number of lines: `1 + 2*N_SSetStd`

The sixth section defines the standard sidesets within the mesh, those with IDs from SS301 to SS399 and SS501 to SS599. Similar to the aerodynamic interface tags described above, standard sidesets are defined in terms of element faces. The first line of the section gives the number of standard sidesets, `N_SSetStd` (4), followed by the max size of a sideset in terms of element faces (608).

The sideset header line is followed by a list of sideset definitions, each two lines long for a total of `2*N_SSetStd` lines. The first line of the sideset definition begins with its number (1, 2, …, `N_SSetStd`) followed by a string containing its name (e.g., `'SS301'`). After the sideset name is the number of element faces contained in the sideset, `N_Faces` (608 for `SS301`, 44 for `SS502`), followed by a list of `N_Faces` element IDs. Note that a single element may contain multiple faces which belong to the sideset, in which case its element ID will appear more than once.

The second line of the sideset definition contains `N_Faces` integers, valued between 1 and 6. The value indicates which face ID of the corresponding element belongs to the sideset. The `MeshProc.m` function automatically indents the face ID entries so they line up with their corresponding element ID. Face assignments and natural node IDs are discussed in more detail in Section 2.6.1.2.

The boundary sideset SS300, whether or not it exists, is not included in this section, and is therefore not input into X3D. Sideset SS301, containing the wetted surface information, is included in this section of the mesh data file. This duplicates the information contained in the aerodynamic interface definition, section 5 of the mesh data file, but in a more compact form because it does not list non-wetted faces. Although it is currently unused, it is left in the mesh data file for potential use in later versions of X3D.

Section 7: Standard Nodeset Definitions; Number of lines: `1 + N_NSetStd`

The seventh section defines standard nodesets, those with IDs from NS401 to NS499. The first line of the section gives the number of standard nodesets, `N_NSetStd` (3), followed by the number of nodes in the largest standard nodeset (176). The nodeset header line is followed by `N_NSetStd` (3) lines defining each of the standard nodesets. The first value in the line gives the nodeset number, and is followed by a string defining the nodeset name (`NS400`, `NS401`, and `NS402`). The next value is an integer containing the number of nodes in the nodeset, `N_NSNodes` (176, 6, and 6). Finally, `N_NSNodes` IDs are listed, corresponding to the global node IDs as defined at the beginning of the mesh data file.

Note that the nodeset numbers start at zero in this example. This is because the boundary nodeset NS400 is defined. If NS400 was not defined during meshing, the first nodeset NS401 would still be given nodeset number one, with zero skipped.

A circumstance may arise where a single mesh may be used for multiple types of analysis, and a boundary nodeset NS400 is only needed some of the time. In this case, the nodeset can be included in the mesh data file, and disabled when not in use. NS400 can be deactivated by manually editing its name so it does not match the format expected, for example replacing it with “NS4XX”. Other nodesets besides NS400 can also be disabled in the manner if not needed, for example, a nodeset containing nodes at the tip of a blade for static testing can be disabled for trim studies. Standard nodesets with IDs greater than 450 (NS451–NS499) will also be listed in this section, but are currently ignored by the solver. These nodesets can be used for various diagnostic purposes, for example tracking how the IDs of a group of nodes changes during mesh renumbering.

Section 8: Cross-section Nodeset Definitions; Number of lines: 1 + N_NSet789

The eighth section defines the cross-sectional nodesets in class 700, 800, and 900. Class 700 nodesets define CN (chord and normal) lines, which are used to define beam-like cross-sections needed to output sectional beam-like deformations; class 800 define SC (structural cut) sections, which are used for determining equivalent beam-like sectional blade loads; and class 900 define AE (aerodynamic) sections, which are used to determine blade sections where aerodynamic loads are imposed. Each of these classes contain exactly four sidesets if defined, (e.g., NS701, NS702, NS703, and NS704), therefore the number of cross-section nodesets, N_NSet789, will be 0, 4, 8, or 12.

Section 8 begins with a header line, listing the number of cross-section nodesets (12) and the maximum size of a cross-sectional nodeset (25). This header entry is followed by the definitions of the sectional nodesets, grouped by class. Each of the classes of nodesets is printed as a group of four lines. Each line has a similar format, starting with a sectional nodeset number, followed by a sectional nodeset name, the number of nodes in the nodeset, and a list of the node IDs. This format is similar to the standard nodeset definition but the names are different. Instead of being identified by nodeset ID (e.g., “NS702”) they have unique names (e.g., “CNTE”), presented in Table 2.4.

Table 2.4 Description of cross-section nodesets and their names

Nodeset ID	Nodeset Name	Description
NS701	CNLE	Deformation section, Leading Edge (**C**hord-**N**ormal)
NS702	CNTE	Deformation section, Trailing Edge (**C**hord-**N**ormal)
NS703	CNUS	Deformation section, Upper Surface (**C**hord-**N**ormal)
NS704	CNBS	Deformation section, Bottom Surface (**C**hord-**N**ormal)
NS801	SCLE	**S**tructural **C**ut section, Leading Edge
NS802	SCTE	**S**tructural **C**ut section, Trailing Edge
NS803	SCUS	**S**tructural **C**ut section, Upper Surface
NS804	SCBS	**S**tructural **C**ut section, Bottom Surface
NS901	AELE	**AE**rodynamic section, Leading Edge
NS902	AETE	**AE**rodynamic section, Trailing Edge
NS903	AEUS	**AE**rodynamic section, Upper Surface
NS904	AEBS	**AE**rodynamic section, Bottom Surface

Section 9: Aero Segment Sideset Definitions; Number of lines: `Variable`

The ninth, and final, section defines the “aero segments” contained in the 900 class sidesets (SS901 and up), which are used by the fluid structure interface (FSI) within X3D. The first line of the section has three values: the number of aero segment sidesets `N_SS900` (10), the max number of faces in an aero segment sideset (28), and the max number of nodes in the face, which is 9 as X3D only supports Hex27 elements at this time.

The heading line is followed by `N_SS900` (10) subsections, each defining one of the aerodynamic segments. The first line of the segment subsection is a header, and begins with the aero segment number (`1, 2, ..., N_SS900`), followed by the name of the aero segment. Unlike standard sidesets, the aero segment sideset name is not identical to its sideset ID. Instead, it takes the prefix AE and an ID starting from 101. Sideset SS901 thus becomes aero segment AE101, SS902 becomes AE102, and so on. After the sideset name is an integer defining the number of element faces in the sideset, `N_Faces` (28, 28).

The number of faces is followed by a list of that many element IDs. Like the standard sidesets, if an element has more than one face contained in the aero segment, its ID will be repeated.

The aerodynamic segment subsection header is followed by a list of `N_Faces` (28) lines defining the nodes within the element faces that lie on the aero segment surface; the first line identifies a face in the first element ID in the aerodynamic segment subsection header, the second line corresponds to the second element, and so on. Each line begins with the number of nodes on the face (9, for Hex27) followed by the natural node IDs in the element reference frame. The natural node IDs are based on the face within the element, as discussed in greater detail in Section 2.6.1.2: Natural and Global Node and Face IDs. This aerodynamic subsection structure is then repeated for every aero segment.

2.6.1.2 Natural and Global Node and Face IDs

The 27 node hexahedral element (Hex27) is defined using an isoparametric formulation, based on a cube with 27 nodes: one at each corner (8), one at the midpoint of each edge (12), one at the center of each face (6), and one at the center of the element volume (1). While this is true of all Hex27 elements, the conventions for identifying nodes within the element may vary depending on the source which created the element. The number of a node within an element is known as its natural, or local, ID. This ordering of nodes within an element must be defined in the SAM input to X3D.

Cubit assigns IDs as nodes are created by its meshing algorithms, so there is no typical natural node ID within Cubit. When exporting the mesh as an I-DEAS Universal .unv file, however, the natural node ordering is standardized. The standard natural node

IDs for an I-DEAS Universal element are shown in Figure 2.40. X3D utilizes a different standard, shown in Figure 2.41.

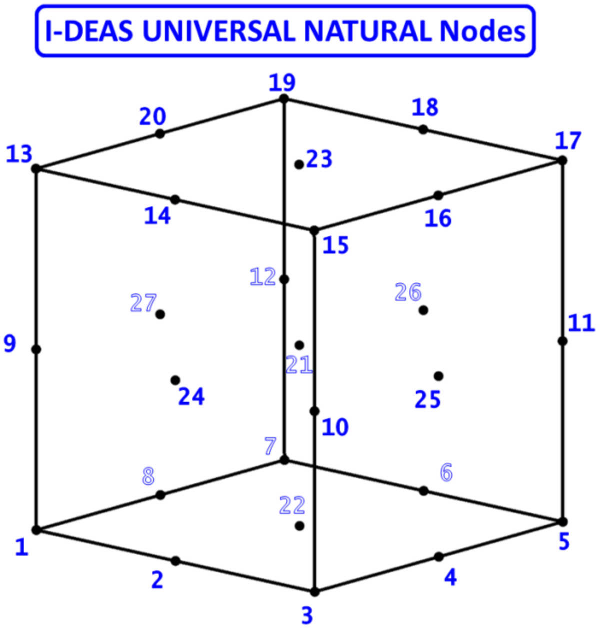


Figure 2.40 Natural node IDs: I-DEAS Universal

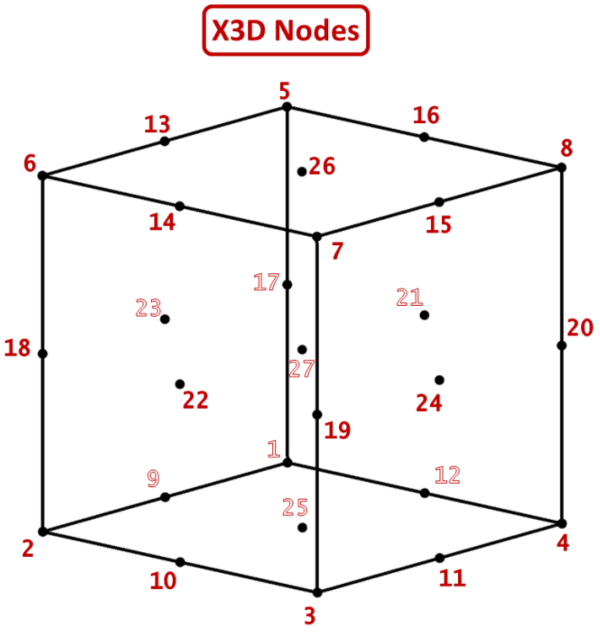


Figure 2.41 Natural node IDs: X3D

Natural node IDs from the I-DEAS Universal .unv mesh files are preserved by the `MeshProc.m` processor, even when global node ID renumbering to reduce mesh bandwidth is enabled (Section 2.6.1.4.2). Consequently X3D mesh data files generated from Cubit using the workflow described here follow the I-DEAS Universal standard in Figure 2.40. When defining the SAM input file, it is necessary to define a natural node ID conversion, which allows X3D to properly parse the elements in the mesh data file (discussed in Section 2.6.5). The natural node ID relation between I-DEAS Universal and X3D is provided in Table 2.5.

Table 2.5 Mapping of Hex27 natural node IDs between I-DEAS Universal .unv and X3D

UNV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
X3D	2	10	3	11	4	12	1	9	18	19	20	17	6	14	7	15	8	16	5	13	27	25	26	22	24	21

Similarly, the face IDs associated with an element vary between I-DEAS Universal and X3D. Figure 2.42 compares the face numbering for I-DEAS Universal Hex27 elements and that for X3D. The orientations of the faces in this figure match with the nodal IDs in Figure 2.40 and Figure 2.41. Table 2.6 compares corresponding face IDs in I-DEAS Universal versus X3D, and Table 2.7 provides the four corner nodes which bound each face in the two standards.

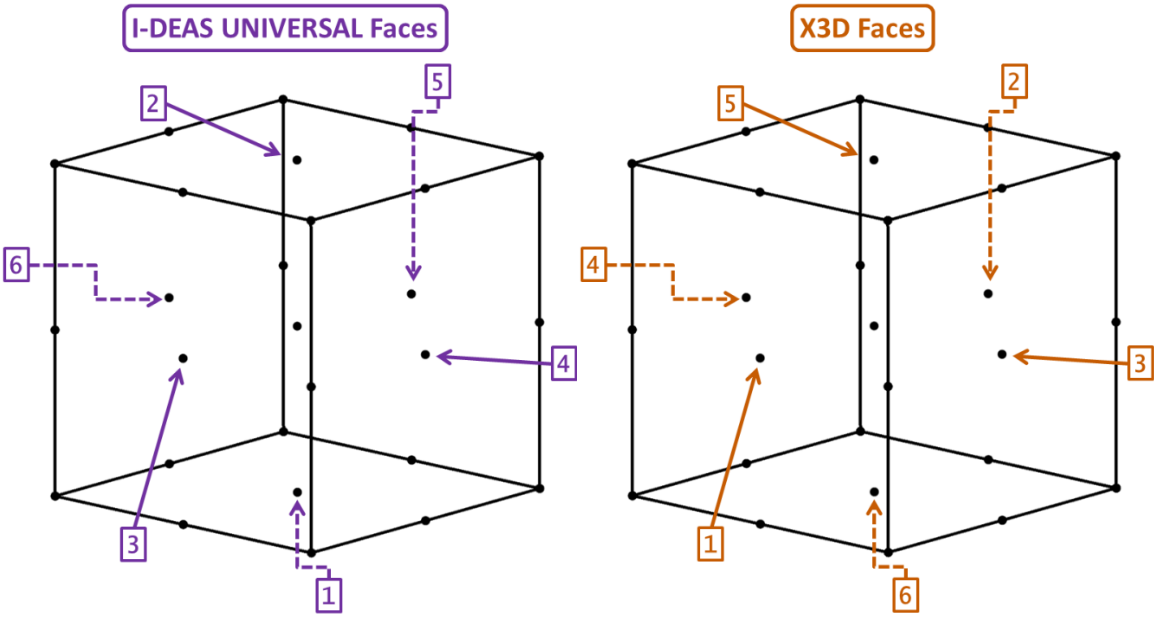


Figure 2.42 Corresponding face IDs for I-DEAS Universal and X3D Hex27 elements

Table 2.6 Mapping of Hex27 face IDs between I-DEAS Universal .unv and X3D formats

UNV	1	2	3	4	5	6
X3D	6	5	1	3	2	4

Table 2.7 Corner nodes which define faces in I-DEAS Universal and X3D formats (colors of natural node IDs indicate corresponding faces based on Table 2.6)

Node IDs: IDEAS-Universal	Face ID	Node IDs: X3D
1, 3, 5, 7	1	2, 3, 6, 7
13, 15, 17, 19	2	1, 4, 5, 8
1, 3, 13, 15	3	3, 4, 7, 8
3, 5, 15, 17	4	1, 2, 5, 6
5, 7, 17, 19	5	5, 6, 7, 8
1, 7, 13, 19	6	1, 2, 3, 4

2.6.1.3 Global Node IDs and Bandwidth

When executed on a single processor, the X3D solver uses skyline storage to reduce the size of sparse matrices by only storing non-zero entries. This method is efficient for finite element models since they tend to form banded stiffness matrices, with non-zero terms falling near the diagonal. The efficiency of the method degrades, however, as the height of the skyline increases, which is to say when data is stored far from the diagonal. Thus, efficient structural solution requires that nodes in close spatial proximity are numbered close together, minimizing the bandwidth of any given element. The bandwidth for an element is defined by the difference between the highest and lowest global node ID number included in the element:

$$Bandwidth = MaxNodeID - MinNodeID \qquad 2.1$$

Furthermore, to allow decomposition of the blade into multiple domains for parallel computing, it is required that element numbers are also grouped together based on their location.

When Cubit creates a mesh, it assigns IDs to nodes in the order they are created by the meshing algorithms; the first node gets the ID 1, the second node ID 2, and so on. While this makes sense from the perspective of the meshing software, it can lead to high bandwidth storage. This is illustrated with an example mesh of a brick shaped object. First the root surface is meshed, as shown in Figure 2.43. Since the default mesh element in Cubit is the Hex8, it begins with a Quad4 face, and the first nodes it creates are given global node IDs 1-4. To mesh the rest of the volume, Cubit subsequently meshes the far surface of the block, giving the nodes on the opposite end IDs 5-8, as shown in Figure 2.44. The linking surfaces are then meshed, one curve at a time. In the example case, that results in groupings of nodes with global IDs 9-11, 15-17, and 18-20 as shown in Figure 2.45, completing the Hex8 mesh. The final step is conversion from Hex8 nodes to Hex27, with Cubit assigning IDs based on node creation order in its internal algorithm. Once again, the ordering is not optimal, leading to high bandwidth storage.

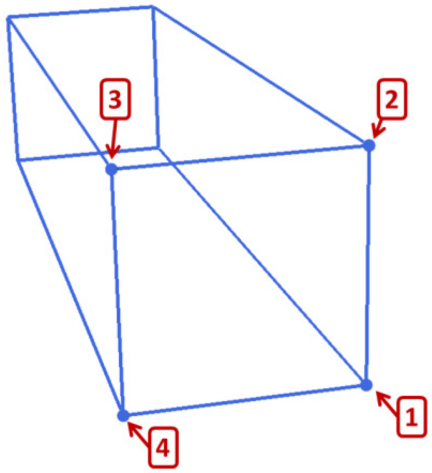


Figure 2.43 Cubit node ID assignment, part 1: The root surface is meshed first, creating node IDs 1-4

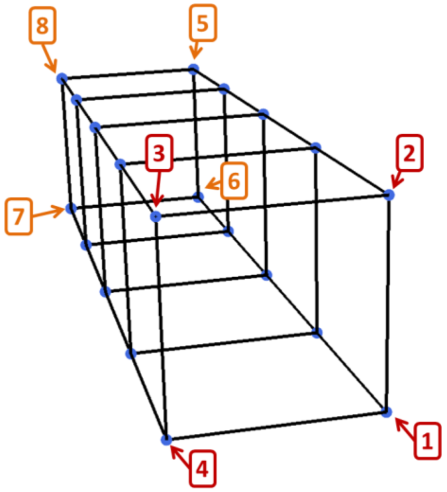


Figure 2.44 Cubit node ID assignment, part 2: Next, the far end is meshed, creating node IDs 5-8

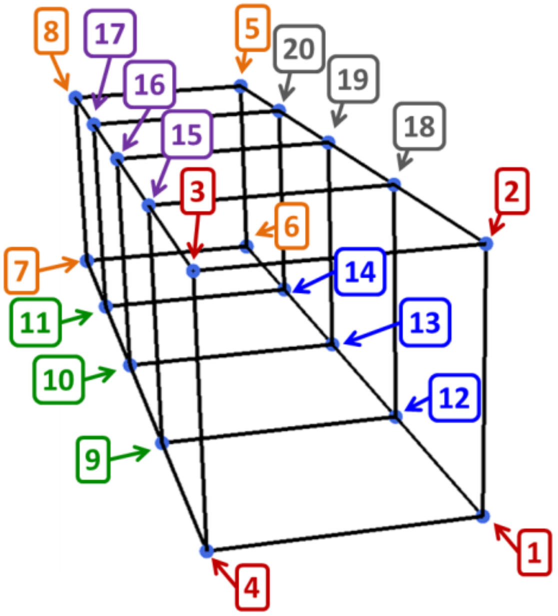


Figure 2.45 Cubit node ID assignment, part 3: Linking curves are meshed next, leading to low bandwidth elements at the ends; conversion to Hex27 elements adds interior nodes as a final step, further degrading bandwidth of every element

Rotor geometry tends to be strongly radial, swept from root to tip. This leads to node ordering in a pattern similar to that above, with bandwidth that gets progressively worse as the number of radial elements is increased. The bandwidth issue can be mitigated by renumbering the nodes based on their geometric positions.

As with nodes, Cubit assigns hex IDs as the Hex8 elements are created. Unlike nodes, when a volume is swept the hex IDs increase from the source surface to the target surface. However, when volumes/elements are added to blocks, Cubit adds an element ID on top of the default hex ID. Whereas hex IDs are assigned as the elements are created, element IDs are grouped by block, then numbered based on the hex IDs within the given block. It is the element ID which gets output into the I-DEAS Universal file by Cubit.

Since most blades have multiple blocks for different internal geometry elements, the element IDs can be scattered. For the sake of partitioning, the elements may also be renumbered. Node and element renumbering is performed by default within the `MeshProc.m` mesh processing script, and is discussed in Section 2.6.1.4.2.

2.6.1.4 The `MeshProc.m` Mesh Processing Script

The `MeshProc.m` mesh processor is a MATLAB script which converts I-DEAS Universal .unv files into the X3D mesh data .dat file format. The script is compatible with mesh files created in Cubit using the workflow described above. By default, the processed mesh is output in the X3D .dat mesh data file format standard defined above in Section 2.6.1.1, which is compatible with X3D versions 1.4.3 and later.

The `MeshProc.m` MATLAB script is formatted as a function with three primary inputs: the name of the .unv mesh file to process (without the extension), the directory in which it is stored, and an array "`DimOrder`" which defines the directional order in which to renumber the mesh, which is discussed in further detail in Section 2.6.1.4.2. The file name and directory are both strings, and `DimOrder` is a three integer vector.

In addition to the three main inputs, there are two optional inputs. The first, "`PlotOn`", is used to activate optional mesh plotting. When `PlotOn` is set to 1, the nodes are plotted using the MATLAB `plot3d` function, with each block given a different color. Nodes belonging to nodesets and sidesets are plotted with indicative markers, helping the user verify that the mesh in the Universal .unv file was read correctly. It is recommended that plotting should be activated for all new meshes. The plotting routine included in `MeshRenumber.m` is relatively inefficient and can be slow for large meshes (on the order of one minute for meshes with ~10K nodes). Optionally, `PlotOn` may be set to 2 in order to plot node positions without identifying block, sideset, or nodeset, speeding up plotting considerably. If no input is provided, `PlotOn` is set equal to 0 by default, disabling plotting.

The second optional input is "`RenumberMesh`". When equal to 1, which is the default setting, the mesh processor will call a function to renumber the node and element IDs within the mesh, thereby reducing bandwidth. This option is recommended but it can be disabled by setting `RenumberMesh` equal to zero as an input.

2.6.1.4.1 Mesh Processor Code Structure

The `MeshProc.m` script begins by reading in the I-DEAS Universal .unv mesh file. The .unv data file is organized into several datasets, each marked by a three or four digit flag. The relevant dataset flags are listed in Table 2.8.

Table 2.8 I-DEAS Universal .unv mesh dataset flags

Flag	Dataset
1716	Material definitions
2411	Node definitions
2412	Element definitions
2477	Nodesets
790	Sidesets

The processor reads through the .unv file until it finds one of the recognized flags. Then, based on the type of dataset encountered, it will read the lines within the section to load the required data. Documentation on all of the datasets except 2411 (nodesets) can be found online [120].

The first dataset encountered is 1716, containing material definitions. This section provides details on any material definitions created in Cubit. The properties extracted include the material ID (which indexes from 1), the name of the material created in Cubit, and the material properties (modulus of elasticity, Poisson's ratio, shear modulus, and mass density). These properties are saved in a cell array "`MaterialsList`". Material properties are not used in the X3D mesh data file, and therefore are not necessary, but they are left in the mesh processor for potential future use. Dataset 1716 is repeated in the .unv mesh data file once for every material used in the mesh.

Next is dataset 2411, containing node definitions. Nodes are defined by a node ID and three double precision coordinates, X, Y, and Z in the part reference frame. The I-DEAS Universal .unv mesh data file lists nodes in order of their Cubit global node IDs, which are assigned in the order the nodes were created by Cubit, as discussed in Section 2.6.1.2. The node definitions are stored in the `NodeMatrix` array, which is an `[N_Nd x 4]` size matrix (recall, `N_Nd` is the total number of nodes in the mesh). There should only be one dataset 2411 in the .unv mesh file.

After node definitions is dataset 2412, containing element definitions. The elements are listed in order by their Cubit element IDs, which under certain circumstances may not match their hex IDs, as discussed in Section 2.6.1.3. For each element, the definition provides the Cubit hex ID, the block ID to which the element is assigned, the material tag for the given block, the color of the element (unused in X3D but saved by `MeshProc.m` for potential later use), and the number of nodes in the element (currently only Hex27 elements are compatible with X3D). This information is then followed by a list of global node IDs which define the element. The positions of the nodes within this list define their natural node IDs in the I-DEAS Universal standard ordering (Section 2.6.1.2). The element deformations are stored in the `ElemMatrix` array, an [`N_El x 32`] size matrix (recall, `N_El` is the total number of elements in the mesh). Although material tags in I-DEAS Universal .unv files start from one and index up, `MeshProc.m` converts mesh tags to 101, 102, etc. as required by X3D (Section 2.6.1.1). Blocks which did not have their materials assigned during meshing are given a default material ID of 100, and these will need to be updated manually by the user.

Element definitions are followed by nodesets, found in I-DEAS Universal dataset 2477. In the .unv mesh data file, nodesets are defined by a number, a name, a size, and a list of nodes. The names of all nodesets are stored as strings in the `NSetNames` cell array and nodeset definitions are stored in the `NSetMatrix` cell array. Each cell in `NSetMatrix` defines the nodeset in the corresponding entry of `NSetNames`. Each cell of `NSetMatrix` is a single row vector contain unsigned integers defining the nodeset number (in order from one to the total number of nodesets in the mesh), the size of the nodeset in terms of number of nodes, and a list of mesh global node IDs which belong to the nodeset.

Finally, sidesets are defined in dataset 790. Each sideset has a name, stored in the cell array `SSetNames`. The sidesets are defined by corresponding entries in the `SSetMatrix` cell array. Each entry in the `SSetMatrix` cell array contains a matrix which identifies the element faces belonging to the sideset. Each row in the matrix identifies a different face, giving a sideset number, a face number within the sideset, an element ID, a face ID (from 1 to 6), and then a list of the global node IDs which comprise the face (9 nodes for Hex27 elements). The element IDs and face IDs are output to the X3D .dat mesh data file, global node IDs are not.

Once the mesh data has been extracted from the source I-DEAS Universal .unv mesh data file, the mesh is renumbered (unless explicitly disabled). If the plotting option is turned on, the mesh is then plotted. The bandwidth of the mesh is calculated and printed to the display screen. Finally, the mesh is output to an X3D .dat mesh data file with the same original file name and new extension (e.g., "`mesh1.unv`" is processed into "`mesh1.dat`"). Overwrite protection will detect if the file name already exists, and if so prompts the user to overwrite, enter a new file name, or abort output.

2.6.1.4.2 Mesh Renumbering

Unless explicitly disabled using an input option, the `MeshProc.m` MATLAB will call a function "`MeshRenumber.m`" to renumber the global node and element IDs to reduce bandwidth. This function takes as inputs the mesh definition arrays defined in the previous section as well as the renumbering order: `NodeMatrix`, `ElemMatrix`, `SSetMatrix`, `NSetMatrix`, and `DimOrder`. It outputs updated versions of the four mesh definition arrays with global node and element IDs reordered.

The renumbering scheme works by re-sorting global node IDs based on nodal coordinates. It does this by calling the `sort` and `sortrows` functions within MATLAB three times, once across each coordinate dimension of the node matrix. The variable `DimOrder` contains a three value array, which specifies the order in which to sort the coordinates, as well as the direction. The three values within the array must be ±1, ±2, and ±3. The values 1, 2, and 3 correspond to the X, Y, and Z directions respectively. The order of these values indicates the order in which the nodes will be sorted. The sign indicates the direction, with positive indicating ascending node order based on the coordinate and negative indicating descending order.

For example, if `DimOrder = [-3, 1, 2]`, the nodes will be sorted first over the Z axis in descending order, so all nodes with the highest Z values are given the lowest global node IDs. Next, they will be sorted in ascending order based on their X coordinate, and finally in ascending order based on their Y coordinate. This is illustrated in Figure 2.46. Before renumbering (left half of figure), the nodes in the labeled face have their original IDs, assigned by Cubit in the order they were created. After reordering (right half of figure), the nine root nodes are given the nine lowest global IDs (1 to 9) because the first sorting direction was -3, and these nodes are all located in equal Z position at the extreme positive tip of the volume. Next, `DimOrder(2) = 1`, so the nodes are sorted in the ascending X direction (bottom to top in the figure), thus the three nodes at the bottom are given the lowest IDs within the subset (1 to 3), the middle nodes the next highest (4 to 6), and the top nodes the highest (7 to 9). Finally, the nodes are sorted in ascending order along the Y axis, so within each subset (1 to 3, 4 to 6, 7 to 9) the nodes on the right hand side of the figure (-Y) receive the lowest IDs, followed by the nodes along the center, and then the nodes at the left hand side (+Y). The next nine nodes (IDs 10 to 18), are those in the corresponding positions one step closer to the root of the volume, as indicated by the label for node 10.

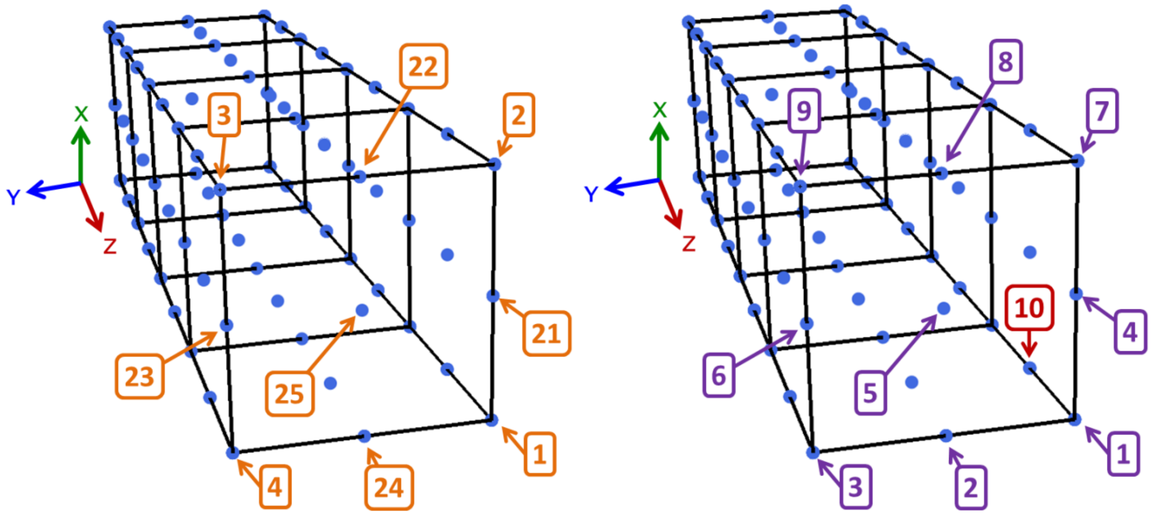


Figure 2.46 Example of mesh global node IDs before (left) and after (right) the reordering function `MeshRenumber.m`

After sorting the global node IDs, the function replaces all global node ID references within the `ElemMatrix` with the updated values. Note that the natural node IDs (the order they are listed within the element) are preserved. Next, to aid in partitioning, the elements themselves are renumbered. This is achieved by sorting the elements based on the lowest global node ID they contain. Finally, all global node and element IDs within the sideset matrix and all node IDs within the node matrix are updated.

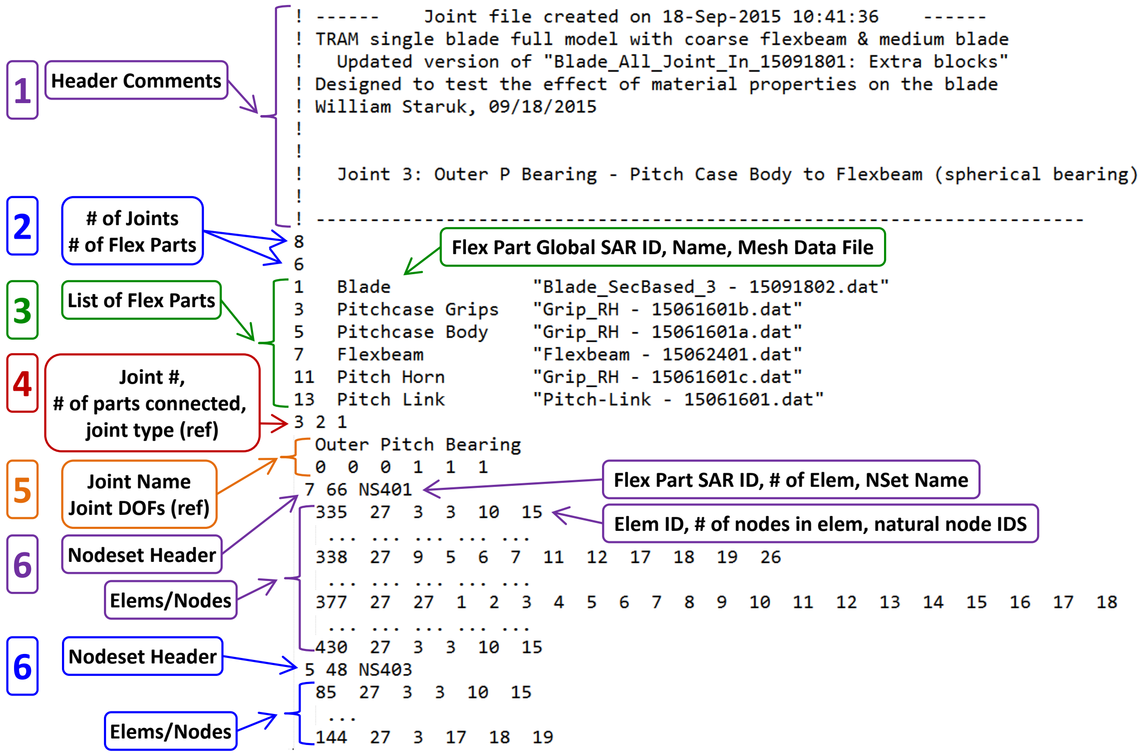
Sorting by elements first, and then nodes, might lower bandwidth even further. However, the renumbering function provided does yield reductions in bandwidth for meshes of slender bodies, with one dominant radial dimension, which is well suited to rotor analysis. For the notional tubular spar rotor mesh example seen in Figure 2.18 and elsewhere in this chapter, renumbering the nodes decreases the maximum element bandwidth by a factor of eight, and the average bandwidth by a factor of three.

2.6.2 Joint Definitions

Joints are idealized parts representing connections within the structural analysis model (SAM) which connect other flexible parts and allow arbitrary relative motion between one other to form the complete multibody assembly. The definitions of the system topology and the decision of which components to model as joints are found in the structural analysis representation (SAR, Section 2.4), but typically joints represent bearings, welds, sensors, and actuators. For the structural analysis model (SAM), each joint in X3D must be defined using the .j.dat joint file data format. These joint data files are created by the `JointDef.m` MATLAB script, which was also developed as part of this research effort. In addition to the joint data file, each joint has certain properties which must be defined in the SAM input file, including type, degrees of freedom, stiffness, damping, commanded motion, or forcing functions.

2.6.2.1 X3D Joint Data File Format (.j.dat)

Like the mesh data file, the input standard for X3D joints is an ASCII file format. The “j.dat” extension is used to identify mesh files, but any ASCII file can be read by X3D. Figure 2.47 shows a sample joint data file and is intended to illustrate the structure of the joint data file standard.



```
! ------     Joint file created on 18-Sep-2015 10:41:36     ------
! TRAM single blade full model with coarse flexbeam & medium blade
!   Updated version of "Blade_All_Joint_In_15091801: Extra blocks"
! Designed to test the effect of material properties on the blade
! William Staruk, 09/18/2015
!
!
!   Joint 3: Outer P Bearing - Pitch Case Body to Flexbeam (spherical bearing)
!
! -----------------------------------------------------------------------
8
6
1   Blade              "Blade_SecBased_3 - 15091802.dat"
3   Pitchcase Grips    "Grip_RH - 15061601b.dat"
5   Pitchcase Body     "Grip_RH - 15061601a.dat"
7   Flexbeam           "Flexbeam - 15062401.dat"
11  Pitch Horn         "Grip_RH - 15061601c.dat"
13  Pitch Link         "Pitch-Link - 15061601.dat"
3 2 1
Outer Pitch Bearing
0   0   0   1   1   1
 7 66 NS401
335   27   3   3   10   15
 ...  ...  ...  ...  ...
338   27   9   5   6   7   11   12   17   18   19   26
 ...  ...  ...  ...  ...
377   27   27   1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17   18
 ...  ...  ...  ...  ...
430   27   3   3   10   15
 5 48 NS403
85   27   3   3   10   15
 ...
144   27   3   17   18   19
```

Figure 2.47 An abridged and annotated.j.dat joint data file illustrating the six different sections of the file standard (color coded) and the contents of each line; ellipses indicate skipped lines, lines truncated at 78 characters for readability

There are six sections in a joint data file, with the sixth section repeating once for each nodeset attached to the joint. These sections are identified in Figure 2.47 by a number, and by the color of their annotations. As with mesh data files, X3D ignores indentation and extra spacing in joint data files. However, the indentation shown in the sample mesh data file is recommended, and is generated automatically by the `JointDef.m` script. All fields in the joint data file are either unsigned integers or strings.

Section 1: Header Comments; Number of lines: 10

The first ten lines of the joint data file contain a header of comments, indicated by exclamation points at the start of each line. This header can be generated by `JointDef.m`, and may contain notes about the structural analysis model for which it was created and the specific joint in question.

Section 2: SAM Part Counts; Number of lines: 2

The first line after the header gives the total number of joints, `N_Joints` (8), within the structural analysis model. The following line gives the total number of flex parts, `N_Flex` (6). These counts are determined when establishing the structural analysis representation.

Section 3: List of Flex Parts; Number of lines: `N_Flex`

The third section lists all of the flex parts in the model in order of their global structural analysis representation part IDs. Each flex part is given one line. First, the global part ID is given (not to be confused with the flexible part ID), followed by a name for the flex part, and the name of the mesh data file for the part. The mesh data file name is given for reference, and allows the model designer to match the joints to the appropriate mesh files from which they were created.

Section 4: Joint Header; Number of lines: `1`

Sections two and three are identical for all joints in a given SAM, but the fourth section begins the definition of a specific joint. It consists of three values, the joint ID (3), the number of parts connected to the joint (2), and the joint type (1). Note that the ID provided in this line is the joint type ID and not the global SAR part ID. The joint type is currently only a reference, with boundaries being given a type value of zero and standard kinematic joints connecting parts a type value of one. This information is currently unused by X3D but can be utilized in the future.

Section 5: Joint Information; Number of lines: 2

The next two lines provide more details on the joint. First is the name of the joint, which can aid the model designer in identifying the component being modeled. This is followed by a six value array giving the degrees of freedom of the joint. Note that the joint name and degrees of freedom in the j.dat joint data file are only given as references at this time; X3D takes joint names and degrees of freedom from the structural analysis model input file (Section 2.6.5).

Section 6: Connected Parts; Number of lines: Variable

The sixth section of the mesh data file defines the elements and nodes connected to the joint. It is repeated for every flex part connected to the joint. If a joint connects three flex parts, the connected nodeset section will be repeated three times. In the example in Figure 2.47, the section appears twice: once for the flexbeam and once for the pitchcase.

Each connected part definition begins with a header line giving the global part ID of the part being connected (7, 5), which can be compared to the list of flex parts in section 3 to ensure the proper part is referenced. The part ID is followed by the number of elements which have nodes connected to the joint, `N_El` (66, 48), and the name of the nodesets which are connected to the joint (NS401, NS403). If a single flex part has multiple nodesets connected to the joint, they will be listed sequentially by `JointDef.m` (e.g., `# # NS401 NS402`), although this is not necessary for X3D. The nodeset names are given only for reference, to assist the model developer in verifying the joint.

The nodeset header line is followed by `N_El` lines identifying the elements and nodes belonging to the nodeset in question. Each line begins with the element ID (335 …

338 …) of the element within the flex part. The element ID is followed by the total number of nodes in the element (27, only Hex27 elements are supported) and the number of nodes within the element which belong to the nodeset and therefore connect to the joint, `N_ElNodes` (3 … 9 … 27). This is followed by a list of `N_ElNodes` natural node IDs. The natural node IDs correspond to the position of the node within the element connectivity matrix as defined in the X3D .dat mesh data file, described in Section 2.6.1.1.

2.6.2.2 The `JointDef.m` Joint Definition Script

The `JointDef.m` joint definition utility is a MATLAB script which generates X3D joint data files (.j.dat) for analysis. It is capable of creating all joints for a structural analysis model at once, and does this by loading in various mesh data files and finding connected nodesets as identified by the user. Joint definitions require many inputs, which are stored in a separate joint input .m MATLAB script file. The only inputs for `JointDef.m` are the file name and directory for the joint input file. The outputs are joint data files, as many as are specified in the joint input file.

2.6.2.2.1 Joint Definition Input File Format

The joint definition script `JointDef.m` calls a joint input file to establish all input variables needed to define the joints. The joint input file is not a function, but a MATLAB script, which stores all variables in the active workspace when run. The joint input file for the bearingless rotor model introduced in Section 2.4.2.2 (Figure 2.8, Table 2.3) is provided in Appendix II as a template. Appendix III offers a second sample joint input file which creates joints for a composite box beam for static tip loading.

First the details of the output are provided. The first variable is JFDir, a string providing the directory in which to save the joint files. A master header variable for all of the joints is defined, containing ten strings which will each be written as a line in the header of every joint file. Additionally, JHeadLine stores a single integer, specifying a certain line to be overwritten by a joint-specific header line. Next, a version tag which will be appended to the file name of each joint created must be provided.

The next section of the joint input file specifies the topological information from the structural analysis representation. It gives the total number of flex parts in the model and the number of joints. It also gives the type of hex element used in the mesh files in terms of number of nodes, which will always be "27" as other element types are not yet supported. The dimensionality of the model is also specified; currently only "3" is supported, indicating a 3-D model.

Next the flex part meshes are listed. The directory in which the mesh files are found is stored in MFDir; the example provided uses the current working directory but any other directory can be provided. Then the flex parts in the model are listed in order of their type IDs (flex part F1 comes first, followed by F2). For each flex part, the mesh data file name (without the .dat extension) and a short reference name are given as strings. The global SAR part ID must also be specified for each flex part, as in Table 2.3.

Finally each joint is defined in order of their joint type IDs (joint J1 comes first). Each joint is given an output name for its .j.dat joint data file, and a brief description which is printed in the joint data file. Then the joint-specific header line is provided as a string; in the example all are the same, printing the joint ID and the joint description. Next, the number of flex parts connecting to the joint is defined, as well as the type of joint (boundary or standard) and its degrees of freedom. Lastly the J_NSets cell array entry for the joint is filled out, listing the nodesets attached to the joint.

Each joint has its own cell in J_NSets. These entries are themselves cell arrays, with one entry for each flex part attached to the joint. For each flex part attached to the joint, the flex part number (matching the index of the mesh file names provided earlier) and the name of its nodesets which are connected to the joint are provided. In the example in the appendix, only a single nodeset is attached to a joint per flex part. If there are multiple nodesets in a single flex part attaching to the joint, it will be listed in the joint input file with all nodeset names:

```
J_NSets{J_ID}{1} = {2,'NS401',’NS402’};
```

2.6.2.2.2 Joint Definition Code Structure

The joint definition script JointDef.m creates X3D joint data files in the .j.dat format standard. It begins by running a joint input file, as described above, the name and location of which are specified as inputs. The script reads the pertinent contents of each mesh data file: the node list, the element connectivity matrix, and the nodesets, skipping all other data which is not relevant to joint definitions.

Once the mesh has been read, the joints are processed. The script loops over all of the joints, checking the entries of J_NSets to find the nodesets which are connected to each joint. It finds the list of the global node IDs belonging to the nodeset and searches for all elements which contain the node in question using the MATLAB find function. A matrix is built of every hit found and sorted, providing a list of element IDs and their natural node IDs which belong to the nodeset in question. All of this information is stored in a cell array JDefs which, for each joint and for each flex part, contains the element IDs, the total number of nodes in the element, the number of nodes in the element which belong to the nodeset, and the natural node IDs which belong to the nodeset.

The final step in the process is to write the joint data files. Each joint gets its own file, with a name “JFName_ver_tag.j.dat” where JFName and ver_tag are variables that were defined in the joint input file. Overwrite protection will check if the name of the joint file being written already exists and, if so, gives the option to abort output, provide a new name for the joint, overwrite the joint in question, or overwrite all joints.

2.6.3 Joint Properties

Although the joint data file includes a line for degrees of freedom, it is intended as a reference only. Joint properties are defined separately in the structural analysis model input file. Available joint properties include kinematic degrees of freedom, stiffness, and damping.

Each joint has three types of degrees of freedom, each provided in the form of a six-integer vector, TJDOF, PJDOF, and FJDOF. Each value in these vectors is either 1 for free or 0 for fixed. The first three values of each vector correspond to translation about the deformed X, Y, and Z axes of the joint respectively, and the last three correspond to rotations about the same axes; the position and orientation of the joint axes in its undeformed state are defined in the SAM input file (Section 2.6.5).

The first type of DoF vector, TJDOF, defines the degrees of freedom in which the joint is free to move. For a locked joint, the TJDOF vector will be [0 0 0 0 0 0]; for a spherical bearing, it will be [`0 0 0 1 1 1`]; and for a vertical slider (prismatic joint) it will be [`0 0 1 0 0 0`].

In addition to basic kinematic degrees of freedom, joint deformation may be commanded using prescribed degrees of freedom `PJDOF`. Prescribed degrees of freedom are those for which the position is commanded, allowing X3D to send specific actuation stroke to a given joint. These command inputs are intended to model actuation, with X3D applying a force on the joint to create the desired deflection:

$$F_{com} = kx_{com} \qquad 2.2$$

where F_{com}, the force applied to the degree of freedom, is based on the stiffness of the joint (k) and the command input (x_{com}). An example of the use of prescribed degrees of freedom is the joint at the bottom of a pitch link representing the connection to a swash plate. X3D prescribes vertical motion of this joint to control the pitch of the blade for trim. In this case, if the joint is treated as a spherical bearing and a vertical slider, its `TJDOF` vector will be [`0 0 1 1 1 1`] and its `PJDOF` vector will be [`0 0 1 0 0 0`].

The joint degrees of freedom can be also be forced using `FJDOF`. Any degrees of freedom identified as being forced will be given a direct constant or harmonic forcing by X3D. This allows the solver to send specific actuation force to a given joint For example, consider a simulation of three point bending. A model of a beam can be developed and connected to three joints. The two joints at the end might be simply supported, and a third joint at the center of the beam will have a forced vertical degree of freedom, represented by `TJDOF` and `FJDOF` vectors which are both [`0 0 1 0 0 0`]. The X3D inputs to apply forces are discussed in Section 2.6.5.

Joints can also be given linear stiffness in any of their degrees of freedom, including cross-coupling. This stiffness is provided by a six-by-six stiffness matrix, `KJDOF`. If a joint is not allowed a particular degree of freedom, it is treated by X3D as rigid in that direction. If a joint is allowed a degree of freedom, however, the stiffness will be obtained from the appropriate entries in `KJDOF`. Values of joint stiffness are given in N/m or N/rad depending on whether the stiffness is for translation or rotation. Similarly, joints can be given a linear damping matrix `CJDOF`, defined identically to `KJDOF`.

2.6.4 Material Definitions

The final consideration in creating a structural analysis model (SAM) is material modeling. Within each flexible part mesh, elements are assigned to groupings known as blocks. Each block is associated with a material tag, which X3D uses to assign the appropriate material properties, including mass density (ρ), elastic modulus (E), shear modulus (G), and Poisson's ratio (v). These properties must be determined by the user and provided to X3D using the structural analysis model `SAM.input` file.

2.6.4.1 Material Properties and Assumptions

Elastic properties for each material tag can be specified in multiple ways within X3D, each identified by a specific flag as summarized in Table 2.9. The simplest option is `ISO`, for isotropic materials, which only requires the elastic modulus and Poisson's ratio. Composite materials are often orthotropic in nature, and their inputs can be provided using the flag `ORTH`. These materials need nine independent elastic coefficients, three elastic moduli, three shear moduli, and three Poisson's ratios. With most composite materials, not all of the orthotropic properties will be provided. In this case there are two sets of transverse isotropic assumptions which can be made using the flag `ORTHTISO` or

ORTHTISO2. For fully anisotropic materials the flag ANISO allows a full six-by-six stiffness matrix (C) to be provided to the solver. All materials require mass density and all units are SI: ρ in kg/m^3, E and G in N/m^2.

Table 2.9 Available material properties input assumptions in X3D

Flag	Description	Required Properties	Assumptions
ISO	Isotropic	E, nu	
ORTH	Orthotropic	E1, E2, E3 G12, G23, G31 nu12, nu23, nu31	

ORTHTISO	Orthotropic, Transversely-Isotropic in 2-3 (YZ) plane [v1]	E1, E2
		G12, nu12
			E3 = E2
			G31 = G12
			G23 = 0.5*(C22-C23)
			nu13 = nu12, nu23 = nu21

ORTHTISO2	Orthotropic, Transversely-Isotropic in 2-3 (YZ) plane [v2]	E1, E2
		G12, nu12
			E3 = E2
			G31 = G12
			G23 = G12
			nu13 = nu12, nu23 = nu12

ANISO	Fully Anisotropic	C11, C12, ..., C16 ..., ..., ..., ... C61, C62, ..., C66	

In addition to specifying the material properties, orientations can be provided. Material orientation is important for modeling the behavior of anisotropic materials appropriately. Orientation is provided in terms of three Euler angles defined about the local reference frame of the part for which the material is being defined, with a provided rotation order. This orientation can be used to specify ply angle for a composite. Another possible use is to rotate the properties of an orthotropic material to model composites oriented perpendicularly, such as the walls and caps of a composite box beam.

It is important to note that X3D currently does not have the capability to automatically transform material properties in relation to element orientation. Consider, for example, a rectangular twisted beam (Figure 2.48). The beam is made up of a single woven ply of ±45° carbon fiber, twisted along its axis and then cured. The material properties of the beam are that of the orthotropic composite material, however, the orientation of the composite material should change along the length of the beam; the primary and transverse material axes should remain tangential and normal to the beam cross-section, respectively. Currently, X3D cannot perform such a calculation automatically. Instead, the only way to model the pre-twisted material properties is to create multiple blocks along the span of the blade, and give each one a unique material tag, as depicted in Figure 2.48. Then, when defining each material tag, the model designer can assign each one an identical set of elastic properties with a tailored rotation angle about the beam axis (P_x in Figure 2.48). The development of a means for automatically rotating material properties based on element orientation is desired, and an area in need of further development.

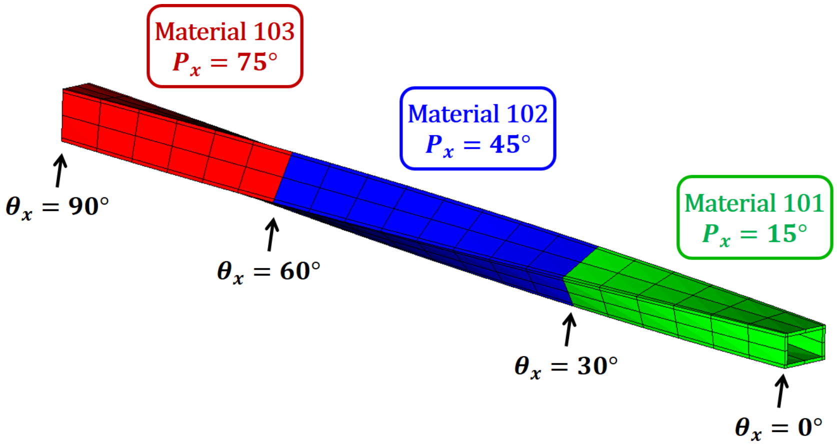


Figure 2.48 Example of assigning materials to a twisted beam to account for material angles

2.6.4.2 Composite Material Homogenization

To minimize the number of degrees of freedom in a structural analysis model, it is desirable to have fewer total elements in the mesh. This can present a challenge when modeling composite materials. A decision point is set by the depth of material modeling desired; the limit to which material properties must be resolved determines the mesh size. Modern composites routinely contain many thin layers, thousands of woven fibers, randomly dispersed mixtures, and possibly even millions of embedded nanotubes. Effective modeling is needed to capture the gross – generally anisotropic – 3-D elastic behavior of such materials. For layered plies, the limit is set by the requirement to calculate inter-laminar stresses. For such cases the modeling must either directly resolve the mesh to this limit or provide a model (analogous to turbulence models in fluids) to recover the stresses from those obtained on a coarser mesh.

In the case of modern composite rotors, for a coarser mesh, a single Hex27 element will likely comprise a laminate made of many layers of different types of material. In order to model such an element, an effective modulus method (also known as the homogenized modulus or smeared modulus method) is adopted [127, 128, 129]. Due to the 3-D nature of this problem, classical laminated plate theory is no longer an adequate technique, and so a modified method presented by Chou, Carleone, and Hsu [130] is used. This method assumes that normal stresses perpendicular to the plane of the layers and the shear stresses in the planes perpendicular to the layers are equivalent and uniform, as are the normal strains parallel to the layers and the shear strains in the plane of the layers. All other stresses and strains are volume averaged. This method ensures that the stresses at the lamina interfaces are in equilibrium, and that the strains are such that the material can remain bonded.

The effective modulus method employed builds 3-D material properties (a full 6-by-6 stiffness matrix for use with the X3D `ANSIO` material tag) from properties of many individual plies. This is not required for 3-D analysis, as each ply can be treated with its own element, but when used allows a coarser mesh, leading to a smaller problem size. Experimental validation of this method is presented in Section 3.1.1.

2.6.5 X3D Structural Analysis Model Input File (`SAM.input`)

The structural analysis model (SAM) is defined within X3D by the `SAM.input` file. This is an ASCII FORTRAN namelist file which points X3D to the appropriate mesh and joint data files, assigns material properties, and defines joint properties. This section describes the contents and functionality of the `SAM.input` file. A full sample structural analysis model input file for a composite box beam undergoing static tip loading is provided for reference in Appendix IV. As with the other X3D input files, `SAM.input` uses a special FORTRAN namelist input structure. This allows the same variable names to be used repeatedly for different sections of the input file, allowing for the input format to be easily scaled up to include any number of parts.

Structural analysis models in X3D are broken into subsystems and components. A wind tunnel model of a rotor will have a single subsystem, the rotor, comprised of a single component, a blade assembly, which gets repeated once for each blade. A single main rotor helicopter could potentially have subsystems for the main rotor, the tail rotor, and the fuselage. The first portion of the `SAM.input` file defines the number and types of subsystems within the part, and the number of components in each. This is followed by component definitions, which provide the total number of parts within each component.

Consider as an example the simplest case of a rotating cantilever beam undergoing static deflection from a tip load (full SAM.input file found in Appendix IV). Such a structural analysis model would have three total parts: the joint clamping the root, creating a cantilever condition; the beam, a flexible part; and the tip joint representing the loading condition. Such a model would have a single subsystem, with the sole type `ROTOR` assigned for now, with a single component comprised of three parts:

```
&NLDEF  model='SAM',  action='INIT',  class='SUBSYSTEM', &END
&NLVAL
    STYPE = 'ROTOR',           ! Subsystem type
    IDS = 1,                   ! Subsystem ID
    IDR = 1,                   ! Rotor ID
    NCOMPONENT = 1,            ! # Components in subsystem
&END
&NLDEF  action='ENDINITSUBSYSTEM',  &END

&NLDEF  model='SAM',  action='INIT',  class='COMPONENT', &END
&NLVAL
    IDS = 1, IDR = 1, IDB = 0, IDC = 1,    ! Specify component
    NPART = 3,                             ! Number of parts in component
&END
&NLDEF  action='ENDINITCOMPONENT',  &END
```

After the subsystem definition, the list of parts is initialized (`action = ‘INIT’`), using the information developed for the structural analysis representation. Each part is given a class, `FLEX3D` for flex parts and `JOINT` for joint parts. Each part is given a name, a global structural analysis representation part ID (`IDP`), a type (‘`F`’ or ‘`J`’), and a type ID (`IDF` or `IDJ`). The number of parts connecting to the one being defined is specified, and connecting part IDs are listed. For the static beam test, the box beam will be the only flex part and is connected to both other parts, and the joints will each connect only to the beam:

```
&NLDEF  model='SAM',  action='INIT',  class='FLEX3D',  &END
&NLVAL
    NAME = 'UNIF BOX BEAM',       ! Flex part name
    IDP = 1, TYPE = 'F', IDF = 1, ! Global part ID, part type, flex part ID
    NCONNPARTS = 2,               ! Number of connected parts
     CONNPARTS = 2,3              ! Part IDs for connecting parts
&END

&NLDEF  class='JOINT',  &END
&NLVAL
```

```
    NAME = 'CLAMP INBOARD',       ! Joint part name
    IDP = 2, TYPE = 'J', IDJ = 1, ! Global part ID, part type, joint part ID
    NCONNPARTS = 1,               ! Number of connected parts
     CONNPARTS = 1,               ! Part IDs for connecting parts
&END

&NLDEF  class='JOINT',  &END
&NLVAL
    NAME = 'LOAD OUTBOARD',       ! Joint part name
    IDP = 3, TYPE = 'J', IDJ = 2, ! Global part ID, part type, joint part ID
    NCONNPARTS = 1,               ! Number of connected parts
     CONNPARTS = 1,               ! Part IDs for connecting parts
&END
&NLDEF  action='ENDINITPART',  &END
```

Note that in this case, both joint parts connect only to the beam. Joints, however, require a minimum of two connections in X3D. Since only one was provided, the solver will correctly assume that the other end is connected to the reference frame, allowing the root joint to clamp the beam and the tip load to act on the beam. There may be times where a joint connects multiple parts to the reference frame, for example to model a gimbaled rotor hub. In this case, the number of connecting parts is 2, but an additional input must also be provided to alert the solver to the boundary connection. This is accomplished by introducing an additional variable flag "ISBOUND":

```
&NLDEF  class='JOINT',  &END
&NLVAL
      NAME = 'GIMBAL',
        IDP = 1, TYPE = 'J', IDJ = 1,
        NCONNPARTS = 3,
         CONNPARTS = 2, 3, 4,
        ISBOUND    = 1,
&END
```

By setting the optional boundary flag to one, X3D will connect this joint to the reference frame. Because X3D uses the namelist file input format, the variable ISBOUND will be reused by the next joint defined. Therefore it must be set equal to 0 if there are any more joints defined after the boundary. The order of part definitions in this section does not matter, as long as the part and type IDs are correctly assigned, so any boundary joints can simply be defined last to avoid errors with incorrect boundaries.

Next, each part must be defined (`action = ‘DATA`). Flexible parts and joints must be pointed towards the appropriate mesh and joint data files respectively. Positions (`RXP`) and orientations (`CXP`) must be defined for both flexible part grids and joints in terms of the global coordinate system (Section 2.4: Structural Analysis Representation).

Flexible parts must be defined first, followed by joints, since joints need flex part mesh information to establish proper connections. Flex parts have their own set of unique inputs. The `GRIDFILE` input points X3D toward the proper mesh file to use. An input for mesh scaling (`GRSCALE`) is provided to correct the mesh units to meters (discussed in Section 2.5.2.4) and the inputs `H27NORDER` and `H27FORDER` are used to correct for the difference in natural node ID and face ID orders between the I-DEAS Universal Hex27 element and the X3D Hex27 element (Section 2.6.1.2). The vector `RXP` provides a translation from the subsystem origin to the origin for the flex part. IFGR is a flag which indicates if the flex part reference frame has an orientation different from that for the subsystem and how it will be provided. Options for orientation specification include using a 6x6 rotation matrix (CXP) or a series of Euler rotation angles (`GRANGLE`) and rotation order (`GRORDER`). If a blade has multiple parts with identical meshes (three blades, for example), the same mesh data file can be referenced for each one, but each will need its own orientation and possibly position vector.

```
&NLDEF  model='SAM',  action='DATA',  class='FLEX3D', &END
&NLVAL
    IDC = 1, IDP = 1,                         ! Component and part ID
    GRIDFILE = '../grids/SProps_UnifBox_15111601.dat', ! Mesh data file
    FILEFORM = 0,                           ! Mesh format flag
    GRSCALE  = 0.001,                       ! Dimension scaling [mm]
    IGRAXIS  = 0,                           ! Mesh axis flag
    IFGR     = 0,                           ! Mesh rotation flag
    RXP      = 0.,0.,0.,                    ! Part position vector
    IH27NORDER = 1,                         ! Flag to correct natural node IDs
        H27NORDER =  2,10, 3,11, 4,12, 1, 9, ! Natural node ID conversion
                    18,19,20,17, 6,14, 7,15,
                     8,16, 5,13,27,25,26,22,
                    24,21,23,
    IH27FORDER = 1,                         ! Flag to correct natural face IDs
        H27FORDER = 6,5,1,3,2,4,            ! Natural face ID conversion
    NNODALF = 0,                            ! # of nodal forces
    NNODALM = 0,                            ! # of nodal masses
    DFC = 0.5,                              ! # Aero forcing location
&END
```

In addition to the generic part inputs, joints also have their kinematics defined in the `SAM.input` file. Recall, each joint is given three six-value vectors, `TJDOF`, `PJDOF`, and `FJDOF`, which describe their kinematic degrees of freedom (DoF), prescribed DoF, and forced DoF, respectively (see Section 2.6.3: Joint Properties for more information). For the case of the cantilevered beam, the root joint will have no degrees of freedom, hence all six values will be zero, and the tip can have all degrees freedom, hence all six values are one. The tip joint is also given a vertical forcing using the `FJDOF` vector. Once a forced degree of freedom is defined, harmonic joint forces can be applied. `NHJF` defines the number of harmonic joint forces to apply (one, for a steady tip load), and the `HJF` vector provides the magnitude of the force in newtons:

```
  &NLDEF  class='JOINT',  &END
  &NLVAL
          IDC = 1, IDP = 2,                      ! Component and part ID
          GRIDFILE = '../grids/J_RootClamp_15111601.j.dat',  ! Joint data file
          JFORM = 'EULER',                       ! Joint formulation
          JTYPE = 'GENERIC6',                    ! Joint type
          TJDOF = 0,0,0,0,0,0,                   ! Degrees of Freedom
          PJDOF = 0,0,0,0,0,0,                   ! Prescribed DoF
          FJDOF = 0,0,0,0,0,0,                   ! Forced DoF
      RXP = 0., 0., 0.,                          ! Part position
      CXP = 1.,0.,0.,0.,1.,0.,0.,0.,1.           ! Orientation rotation matrix
```

```
&END

&NLDEF  class='JOINT',  &END
&NLVAL
        IDC = 1, IDP = 3,                      ! Component and part ID
        GRIDFILE = '../grids/J_TipLoad_15111601.j.dat',  ! Joint data file
        JFORM = 'EULER',                       ! Joint formulation
        JTYPE = 'GENERIC6',                    ! Joint type
        TJDOF = 1,1,1,1,1,1,                   ! Degrees of Freedom
        PJDOF = 0,0,0,0,0,0,                   ! Prescribed DoF
        FJDOF = 0,0,1,0,0,0,                   ! Forced DoF
        NHJF  = 1,                             ! Forces: # of harmonics
        HJF   = 0., 0., 5.0e3, 0., 0., 0.,     ! Joint forces: Magnitude
        KJDOF = 0., 0., 0., 0., 0., 0.,        ! 6x6 K matrix
                0., 0., 0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0.,
    RXP =  3.0, 0., 0.,                        ! Part position vector
    CXP = 1., 0., 0., 0., 1., 0., 0., 0., 1.   ! Orientation rotation matrix
&END
```

Alternatively to FJDOF, flex parts can also be given nodal forces directly, using the NNODALF (number of nodal forces) option. It functions in a manner similar to the harmonic joint forces described above, but acts on specific nodes within the mesh. For the example of the cantilevered beam with tip loading, this feature offers an alternative formulation of the structural analysis model. Instead of considering a beam and two joints, a model with a single flex part and no joints can be used. If the beam mesh data file has a boundary nodeset NS400 defined at the root, this will be registered by X3D as a boundary, fixing the nodes in place and obviating the need for a root joint. Nodal forces can then be applied directly to nodes at the tip of the beam, eliminating the need for a tip joint. In this case, the SAM.input file would contain the flex part definition:

```
&NLDEF  model='SAM',  action='DATA',  class='FLEX3D', &END
&NLVAL
    IDC = 1, IDP = 1,
    GRIDFILE = '../grids/SProps_UnifBox_15111601.dat',
    FILEFORM = 0,
    GRSCALE  = 0.001,
    IGRAXIS  = 0,
    IFGR     = 0,
    RXP      = 0.,0.,0.,
    IH27NORDER = 1,
        H27NORDER =  2,10, 3,11, 4,12, 1, 9,
                    18,19,20,17, 6,14, 7,15,
                     8,16, 5,13,27,25,26,22,
                    24,21,23,
    IH27FORDER = 1,
        H27FORDER = 6,5,1,3,2,4,
    NNODALF = 4,
        NODALFNODES = 1967, 1932, 2005, 1970,
        NODALF = 0., 0., 50.0, 0., 0., 0.,
                 0., 0., 50.0, 0., 0., 0.,
                 0., 0., 50.0, 0., 0., 0.,
                 0., 0., 50.0, 0., 0., 0.,
    NNODALM = 0,
    DFC = 0.5,
&END
```

Note that in this example the load is split between four nodes, all located at the tip of the beam and each receiving 50 N. However, the load in this case may not be 200 N, the sum of the four nodal forces. X3D applies nodal forces to the specified nodes each time they appear in an element, so if a node is shared by multiple elements the force will be multiplied, so the input must be divided appropriately.

For example, consider the mesh shown in Figure 2.49, representing the end of the beam. If all nodes exist on the center of a face of their parent element, like the nodes labeled 1, 2, 3, and 4 in the figure, each node belongs to only one element, so the beam will experience four times the nodal force: 200 N. If each of the nodes appear on an outside corner shared by two elements (A, B, C, D) the load will be applied twice to each node, for a total of 400 N. If the nodes lie on internal corners (α, β, γ, δ) the load will on each node will be applied four times, yielding 800 N. This variability makes careful selection of nodes important when using nodal forces, whereas when using joint forces the loads are applied exactly once, making for a more intuitive analysis. Furthermore, joint based loading more closely resembles an experimental set up where a true point force is nearly impossible to achieve.

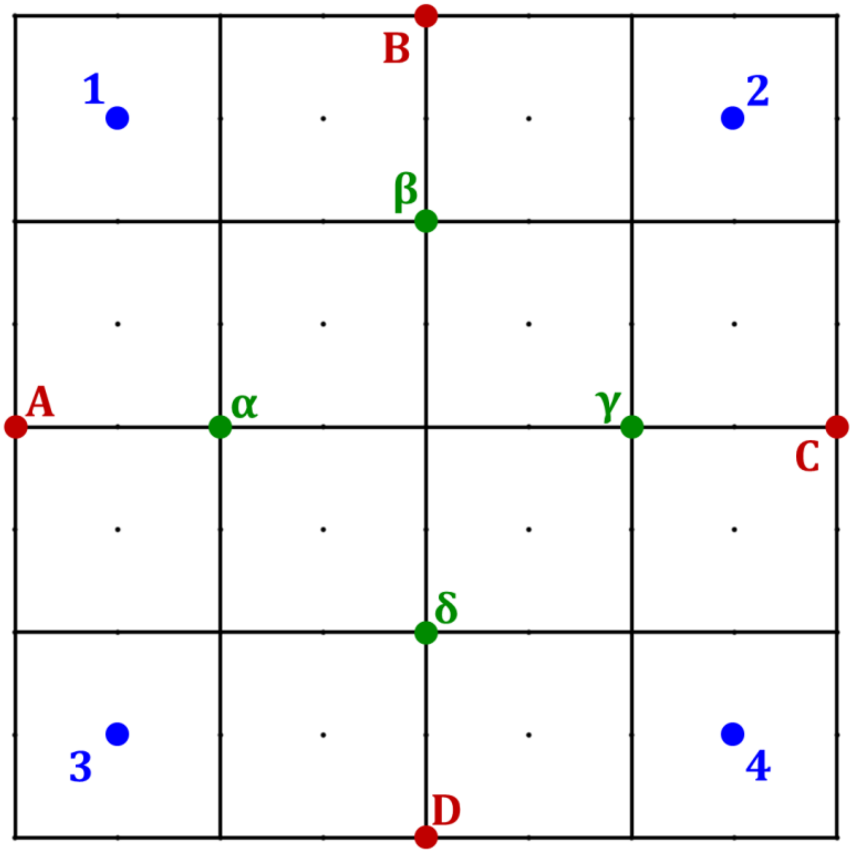


Figure 2.49 Potential nodal force loading positions at the end of a 4x4 element beam mesh

Finally, the `SAM.input` file also defines material properties for each flex part. These material definitions are based on the material tags (`IMAT`) assigned to the elements by the mesh data file. Material properties definitions must be placed immediately after the corresponding flex part because tags can be repeated on multiple flex parts (for example, every flex part will have tag 101, but it may represent different materials on each part). Examples of some of the material definition options described in Section 2.6.4 are provided below.

```
&NLDEF  class='MAT3D', &END
&NLVAL
 IMAT = 101, TYPE = 'iso', RHO = 2810., E = 71.7e9, nu = 0.33,
&END

&NLDEF  class='MAT3D', &END
&NLVAL
 IMAT   = 102, TYPE = 'orth', RHO = 1444.,
 E1     =  55.0000e09, E2   =   2.0000e09, E3   = 0.0,
 G12    =   0.5000e09, G23  =   0.0,       G31  = 0.0,
 nu12   =   0.340,     nu23 =   0.0,       nu31 = 0.0,
&END

&NLDEF  class='MAT3D', &END
&NLVAL
 IMAT = 103, TYPE = 'orthtiso2', RHO = 1444.,
 E1   = 2.0000e+10, E2    = 1.0000e+10,
 G12  = 4.0000e+08,
 nu12 = 3.0000e-01,
 PANGLE = 90., 0., -45.,
 PORDER = 3,1,0,
&END

&NLVAL
 IMAT = 104, TYPE = 'aniso', RHO = 2309.89,
 C11 = 9.e9, C12 = 7.e9,  C13 = 3.0e9, C14 = 1.e8, C15 = 0.,    C16 =  0.,
             C22 = 9.e9,  C23 = 3.e09, C24 = 2.e8, C25 = 0.,    C26 =  0.,
                          C33 = 1.e10, C34 = 3.e7, C35 = 0.,    C36 =  0.,
                                     C44 = 6.0e9,  C45 = 0.,    C46 =  0.,
                                                   C55 = 5.e9,  C56 = -1.e9,
                                                                C66 =  5.e9,
 PANGLE = 0, 0., 0.,
 PORDER = 3,0,0,
&END
```

Note that the in the examples above, the second material (102), which is orthotropic, has no ply orientation provided. Because it is not listed, X3D will default to setting the material E1 axis parallel to the global X axis and its E2 axis parallel to the global Y axis.

The third material (103) is transversely isotropic and has a ply angle (`PANGLE`) of 45° about the Z axis and 90° about the Y axis specified. The order of these rotations is set out by `PORDER`, and proceeds first about the global Z axis (3) followed by the global Y axis (1). In this case, it represents a composite material which is rotated to a ply orientation angle of 45°, then rotated on its side, e.g., for use in the vertical web of a composite box beam.

Finally the fourth material has no ply rotation angle, but unlike material 102 this is specified by setting PANGLE equal to a vector of zeroes. The reason this must be specified is the namelist input style used by X3D; if a variable is not set, the last value used will be taken. While the default is all zeroes for PANGLE, this was changed when material 103 was defined, requiring it to be reset with material 104. Subsequent materials would assume no ply angle until one specifies otherwise.

Another important point about material properties is that they all are defined relative to the subsystem reference frame, not the part reference frame. For a rotor this is a rotating frame which assumes the X direction points toward the blade tip. If four blades were to be modeled using a single orthotropic material with its E1 axis oriented radially toward the tip, each blade would need a different ply orientation angle. The first blade would not need any rotation, as with material 101 above, but the subsequent blades would need a ply angle of 90°, 180°, and 270° about the Z-axis.

2.7 Data Post-Processing

Using full 3-D finite element analysis (FEA) for computational structural dynamics (CSD) predictions results in a wealth of data. Three-dimensional deformations, strains, and stresses are predicted at every node in the mesh. While this offers opportunities to analyze data not available from 1-D beam analysis, it makes comparison to conventional blade structural data difficult. To remedy this issue, specific data post-processors were developed as part of this research to convert 3-D structural predictions into equivalent 1-D beamlike forms for comparison of blade structural properties (bending and torsional stiffness) and sectional blade loads (bending and torsion moments).

2.7.1 Extraction of Beam-Like Sectional Properties

When using 1-D beam structural analysis, the blade is represented using distributed masses and sectional bending stiffness (EI) and torsional stiffness (GJ). These values are typically measured experimentally for existing blades as a basis for analysis. A means of extracting these equivalent, beam-like, sectional properties from CAD-based 3-D models is necessary to validate blade models versus measured data, and to compare predictions from X3D to those from beam-based comprehensive analysis.

Extracting equivalent sectional properties can be achieved by performing static analysis of the model using X3D. The Euler-Bernoulli beam equation is used to determine the beam equivalent EI, or the St. Venant torsion theorem is used to determine GJ. For bending stiffness, Euler-Bernoulli theory dictates that the beam bending moment (M) is approximately equal to the bending stiffness (EI) multiplied by the second derivative of the deflection *w*:

$$M = EI\frac{d^2w}{dx^2}. \qquad 2.3$$

By assuming the bending stiffness is constant over an interval, the bending stiffness can be evaluated by integrating twice the interval:

$$EI = \frac{M}{w'_{x2}-w'_{x1}}. \qquad 2.4$$

where w'_{x1} and w'_{x2} are the bending slope of the beam segment at either end of the segment. When a known tip force is applied to the blade, the bending moment M is known and static analysis in X3D provides the bending slope as an output, allowing sectional bending stiffness of the structural analysis model to be determined between any cross sections of nodes. Similarly for torsion:

$$M = GJ\frac{d\phi}{dx}\text{, and } GJ = \frac{M}{\phi_{x2}-\phi_{x1}} \qquad 2.5$$

where M is the torsional moment and ϕ is the angle of twist, so applying a known tip torque allows determination of the sectional torsional stiffness. This method is analogous to the experimental determination of blade sectional properties using the mirror method, as described in [131].

Validation of the sectional properties extraction method was performed on four simple test cases: a uniform hollow box beam, a tapered box beam, a stepped box beam, and a hollow tapered circular tube. All of the test beams were given the properties of Aluminum 7075-T6 and have geometry as shown below (Figure 2.50 and Figure 2.51).

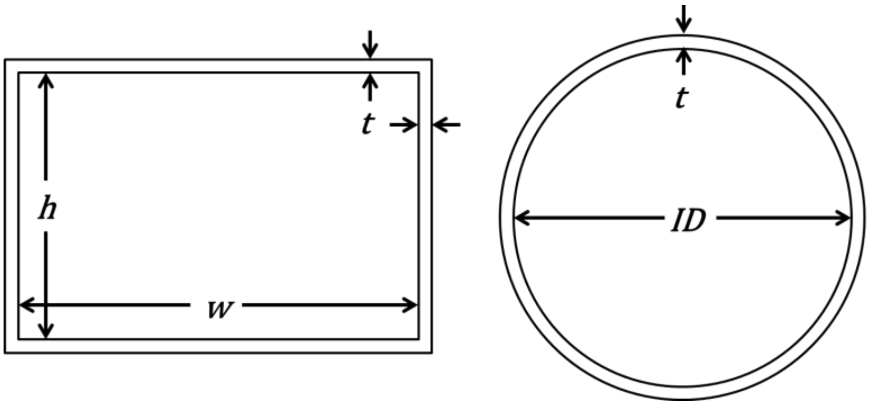
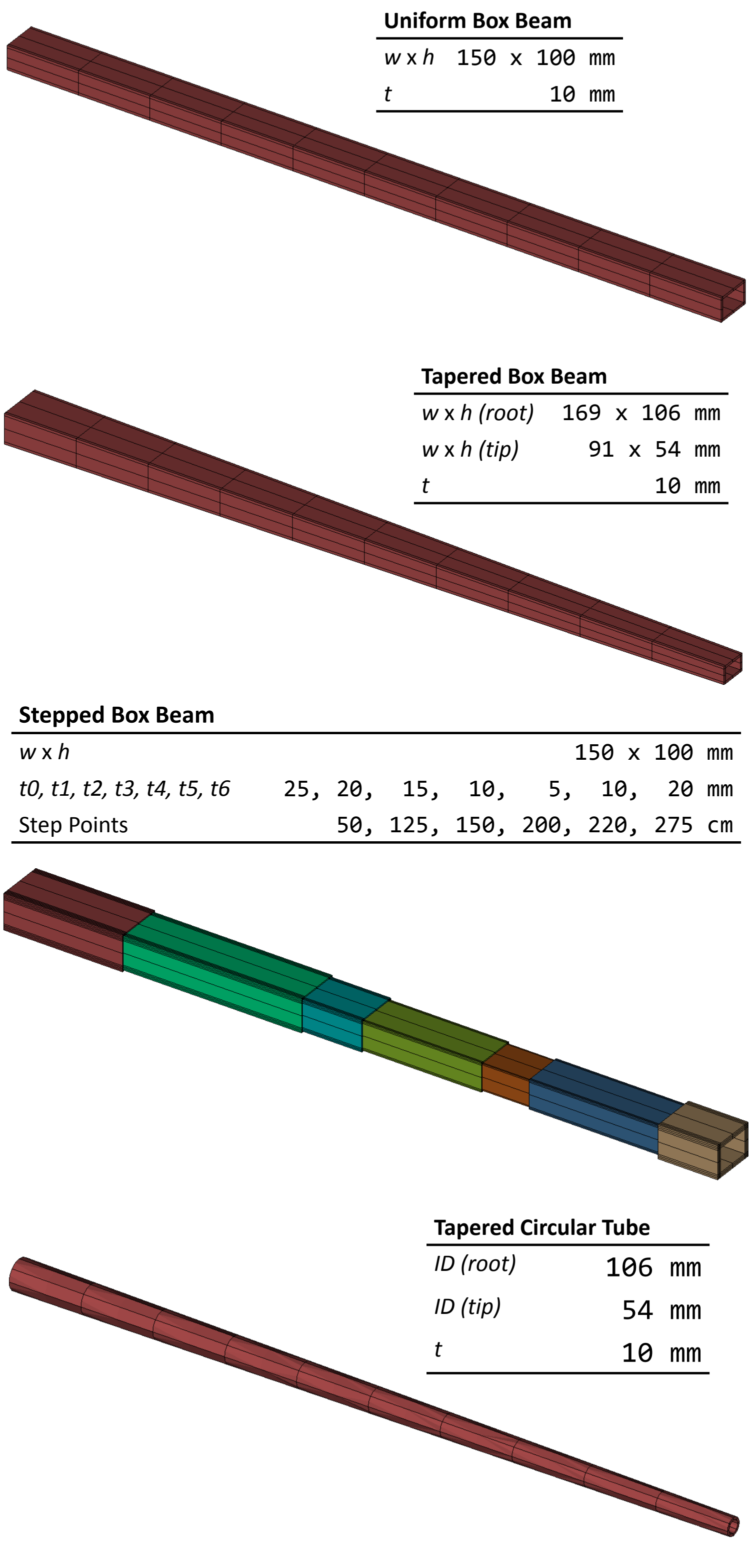


Figure 2.50 Cross-sectional dimensions for 3 D models of box and tube beams



Uniform Box Beam

w x *h*	150 x 100 mm
t	10 mm

Tapered Box Beam

w x *h (root)*	169 x 106 mm
w x *h (tip)*	91 x 54 mm
t	10 mm

Stepped Box Beam

w x *h*	150 x 100 mm
t0, t1, t2, t3, t4, t5, t6	25, 20, 15, 10, 5, 10, 20 mm
Step Points	50, 125, 150, 200, 220, 275 cm

Tapered Circular Tube

ID (root)	106 mm
ID (tip)	54 mm
t	10 mm

Figure 2.51 Geometry of uniform, tapered, and stepped box beam and tapered tube beam used for validation of sectional properties extraction

As can be seen in Figure 2.52, Figure 2.53, and Figure 2.54, calculation of sectional bending stiffness for 3-D structural models in X3D is very accurate, with less than 2% error, for the uniform, tapered, and stepped box beams respectively. This demonstrates the ability of the method to handle non-linear and discontinuous geometries. Torsional stiffness is also well predicted, as demonstrated by analysis of the tapered tube (Figure 2.55).

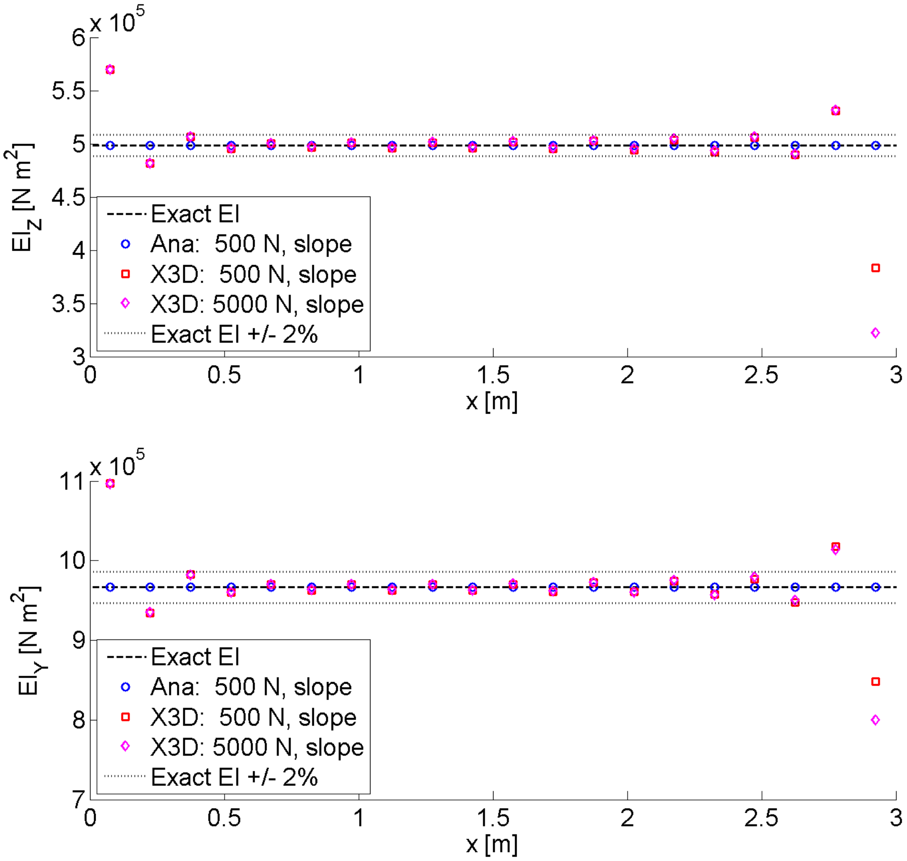


Figure 2.52 Bending stiffness of a uniform box beam, X3D versus analytical calculations

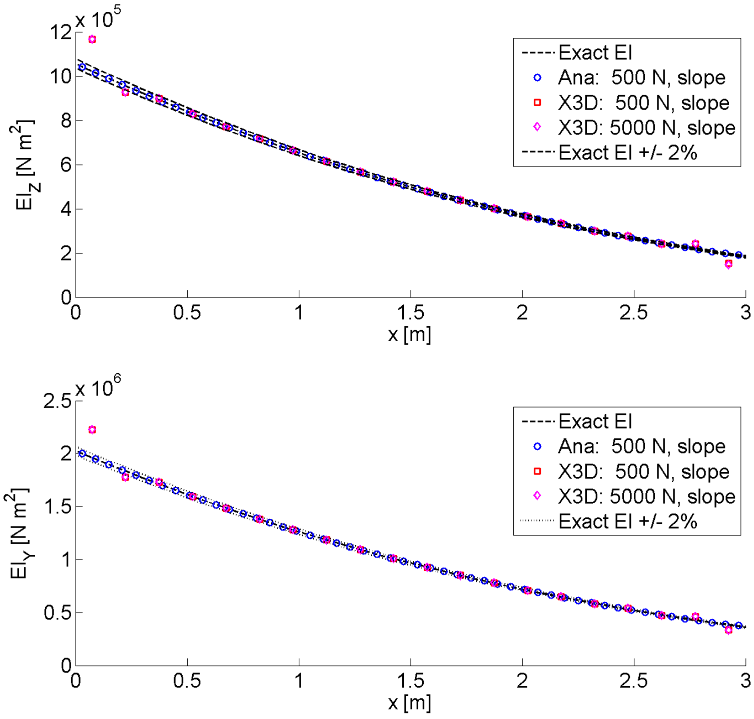


Figure 2.53 Bending stiffness of a tapered box beam, X3D versus analytical calculations

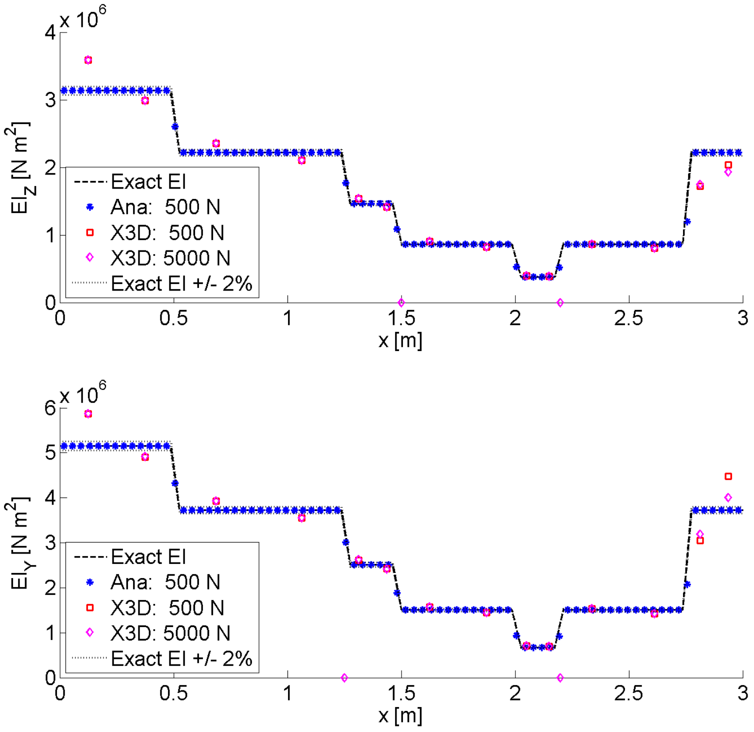


Figure 2.54 Bending stiffness of a stepped box beam, X3D versus analytical calculations

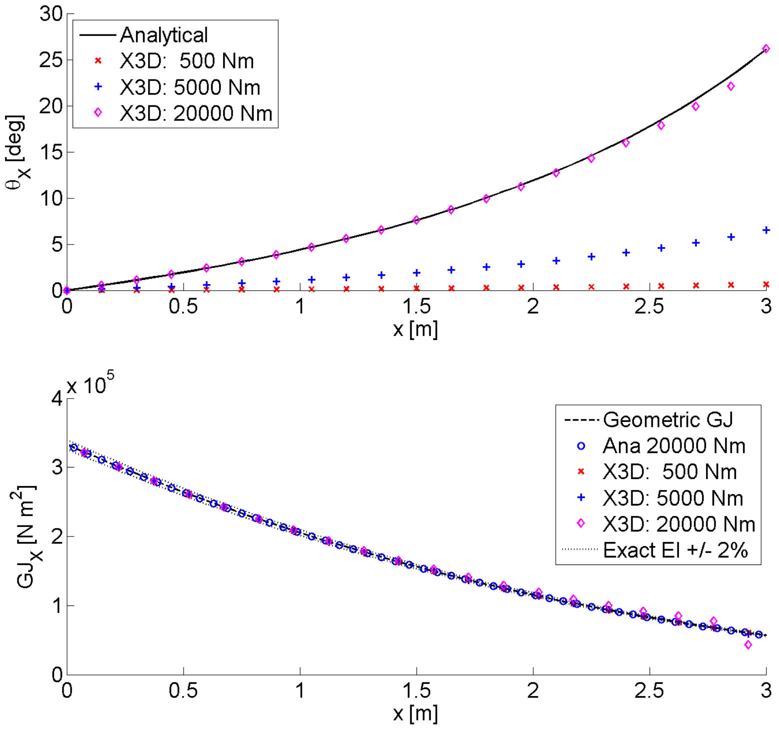


Figure 2.55 Torsion deflection and stiffness, tapered tube, X3D versus analytical calculations

For twisted blades and beams, there is an inherent structural coupling between the flap and lag degrees of freedom that results from the principal axes of the beam not being aligned with the global reference frame axes. Based on analytical deformations of a twisted beam [132], this method for extracting sectional bending stiffness can be extended to a structurally coupled blade. Instead of considering a single flap stiffness and lag stiffness, a coupled formulation of the Euler-Bernoulli theory can be expressed in the global part reference frame as:

$$M_z = EI_{zz}v_y'' + EI_{zy}w_y'' \qquad 2.6$$

$$M_y = EI_{yz}v_z'' + EI_{yy}w_z'' \qquad 2.7$$

In equation 2.6, M_z is the lag bending moment about the part z-axis, caused by an applied chordwise tip force F_y. The in-plane bending curvature created by this chordwise tip force is v_y'' and the flapwise bending curvature caused by the lag force is w_y''. Equation 2.7 describes flapping motion due to an applied vertical tip force F_z. EI_{zz} and EI_{yy} are the sectional bending stiffness about the part z-axis (lag) and y-axis (flap) respectively, while EI_{yz} and EI_{zy} are the stiffness products which create flap/lag coupling. In a symmetric beam, these products will be zero, eliminating the coupling and yielding the independent equations for an uncoupled beam.

Integrating these equations once, putting them in matrix form, and solving for stiffness yields:

$$\begin{bmatrix} EI_{zz} & EI_{yz} \\ EI_{zy} & EI_{yy} \end{bmatrix} = \begin{bmatrix} \Delta v_y' & \Delta w_y' \\ \Delta v_z' & \Delta w_z' \end{bmatrix}^{-1} \begin{bmatrix} \int M_z & 0 \\ 0 & \int M_y \end{bmatrix}, \text{ or} \qquad 2.8$$

$$\boldsymbol{EI} = \boldsymbol{\Delta\delta}^{-1} \int \boldsymbol{M}. \qquad 2.9$$

Here, $\Delta v_y'$ represents the difference in lag bending slope due to a chordwise tip force (v_y') between two deformation sections; $\Delta w_y'$ is difference in flap bending slope due to chordwise tip force; and $\Delta v_z'$ and $\Delta w_z'$ are the differences in bending slope caused by a flapwise tip force. These bending slopes are given as an output of X3D, allowing calculation of the coupled sectional beam stiffness matrix $\boldsymbol{EI}$.

Alternatively, the sectional beam stiffness can be defined in the local coordinate frame, which describes the primary axes of beams with symmetric sections. This requires rotation of the flap and lag bending slopes (v' and w') and the internal bending moments

(M_z and M_y) into the local coordinate frame. Performing this double rotation gives the equation:

$$\begin{Bmatrix} EI_n \\ EI_c \end{Bmatrix} = \begin{bmatrix} \Delta v_y' \cos^2\theta + \Delta w_y' \sin\theta\cos\theta & \Delta v_y' \sin^2\theta + \Delta w_y' \sin\theta\cos\theta \\ \Delta w_z' \sin^2\theta + \Delta v_z' \sin\theta\cos\theta & \Delta w_z' \cos^2\theta + \Delta v_z' \sin\theta\cos\theta \end{bmatrix}^{-1} \begin{Bmatrix} \int M_z \\ \int M_y \end{Bmatrix} \quad 2.10$$

In this version, EI_n is lagwise bending stiffness about the normal (thickness) axis of the cross-section, and EI_c is flapwise bending stiffness about the chordwise axis of the section. For validation of both the global and local format of the sectional stiffness equations, a twisted version of the uniform box beam introduced earlier was considered. Figure 2.56 shows the twisted beam, which goes through 90° of twist from its root to its tip. This beam was created by lofting the root section to the tip using helical guide curves, creating a constant-width, linearly twisted beam (the importance of properly describing the width and twist of a beam is discussed in reference [133]). Figure 2.57 shows that both the global and local sectional bending stiffnesses predicted using X3D match the geometrically calculated values (the small discontinuity at mid-span in the local bending stiffness is due to a trigonometric singularity at the 45° point).

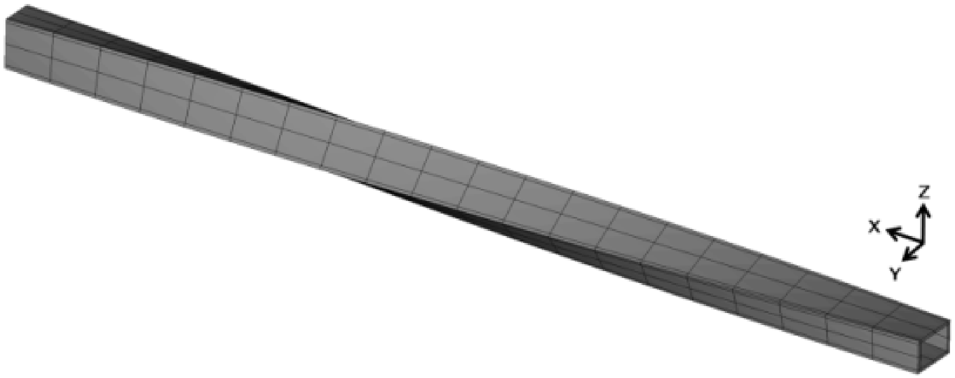


Figure 2.56 Drawing of the twisted beam mesh

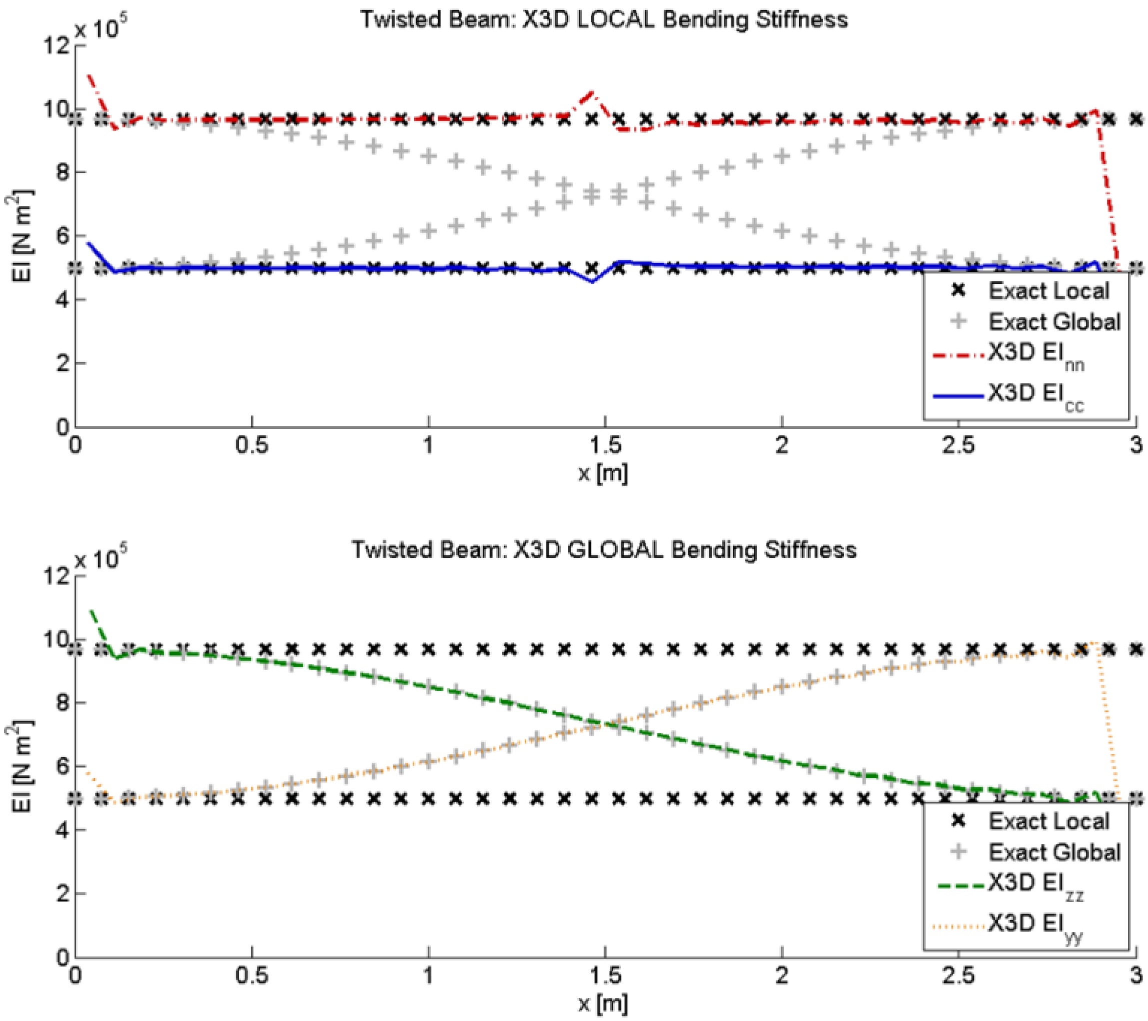


Figure 2.57 Bending stiffness of the twisted box beam calculated using X3D compared to analytical calculations

2.7.2 Extraction of Beam-Like Sectional Blade Loads

Whereas 3-D analysis generates detailed stresses and strains over the entire rotor, as depicted in Figure 2.58, experiments provide structural loads in the blade calibrated as sectional bending and torsion moments. These sectional blade loads are also predicted by conventional beam-based comprehensive analysis. There is no simple way to compare these sectional loads to predictions other than integrating sectional stresses as a post processing step. A more elegant solution is desirable to validate the X3D structural predictions.

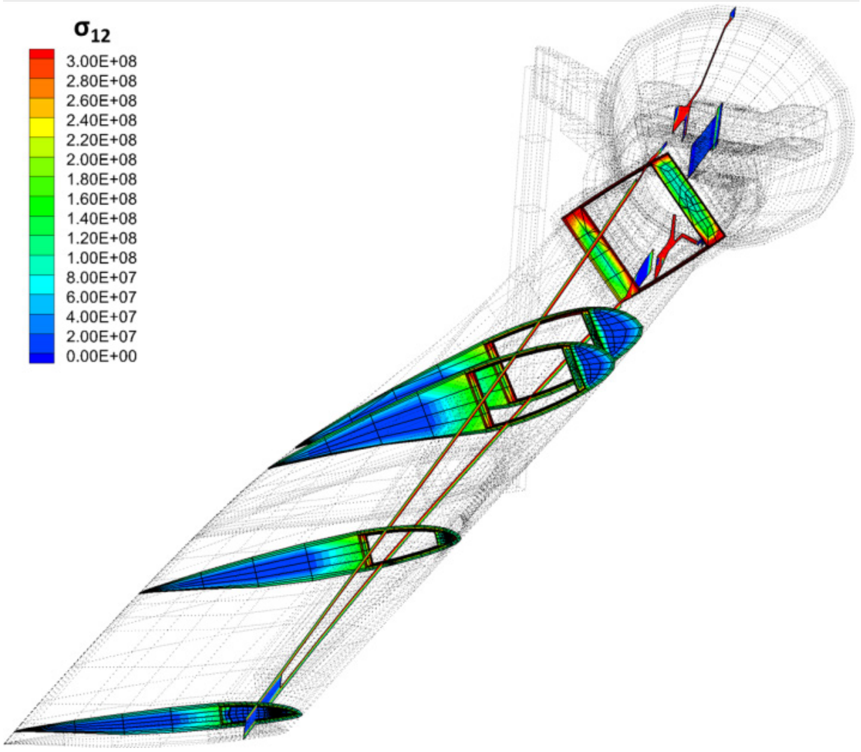


Figure 2.58 Sample 3-D stress distribution prediction for a tiltrotor blade in edgewise flight generated by X3D

One method is to break a mesh at the location where the loads are desired, and place a stiff joint between the segments which can act as a load sensor. This is an approximate method as it constrains the deformations of the section artificially and thus can impact the dynamics of the system. Instead, a method is developed to use the 3-D strain field to extract sectional loads. Analogous to experimental calibration of strain gauges embedded on the blade surface, this method calibrates calculated strains against applied tip loads as a pre-processing step. Unlike experimental measurements, strain can be calculated at any node and is not limited to the surface.

For illustration, consider four nodes, one each on the top, the bottom, the leading edge, and the trailing edge of the cross-section of an ideal 90° twisted beam at a location where sectional loads are desired (Figure 2.59).

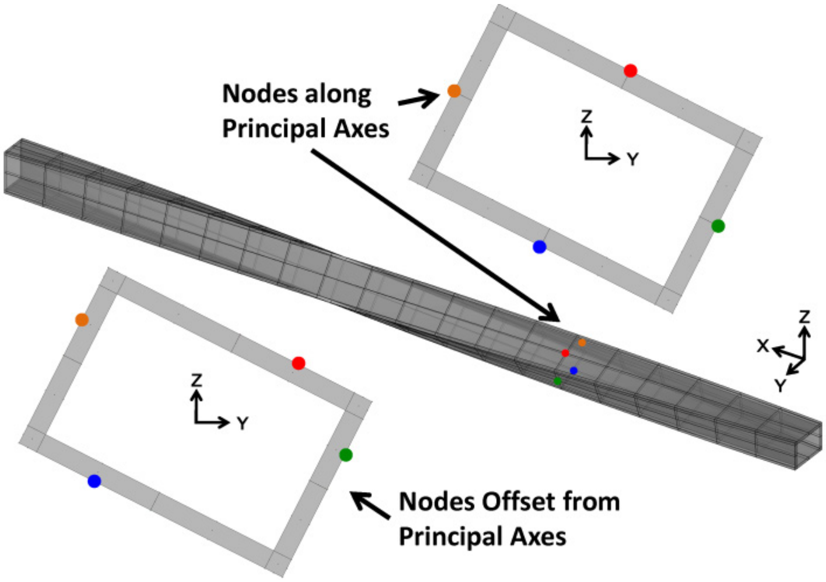


Figure 2.59 A 90° twisted beam with nodes on the top (red), bottom (blue), leading edge (green), and trailing edge (orange) surfaces identified for calculating sectional loads at a single radial station located at 30% radius; nodes may be selected on the principal axes (top right insert) but this is not necessary (bottom right insert)

Static tip loads in extension (F_x), torsion (M_x), flap bending (M_y), and lag bending (M_z) are applied at varying magnitudes one at a time. Unlike experiments, pure moments can be applied in X3D. The strains generated by each loading at the four cross-sectional nodes are identified and grouped into four pairs:

$$\begin{Bmatrix} EX \\ ET \\ EF \\ EL \end{Bmatrix} = \begin{bmatrix} \epsilon_{11_{Top}} + \epsilon_{11_{Bottom}} \\ \epsilon_{12_{Top}} - \epsilon_{12_{Bottom}} \\ \epsilon_{11_{Top}} - \epsilon_{11_{Bottom}} \\ \epsilon_{11_{LE}} - \epsilon_{11_{TE}} \end{bmatrix} \qquad 2.11$$

where EX, ET, EF, and EL represent the extension, torsion, flap, and lag strain pairs, respectively. By taking the sum and differences in this manner, the effects of any one load on the other is minimized (for example, in an ideal beam, pure bending will create equal and opposite stresses on top and bottom, therefore EX will be zero). However, there will still be coupling between loads and uncorrelated strain pairs due to structural coupling or the nodes not being on the principal axes (bottom left inset of Figure 2.59). For these reasons, the calibration takes the form of a Jacobian matrix. Each entry in the Jacobian is the partial derivative of one of the four strain pairs (EX, ET, EF, and EL) with respect to one of the four sectional tip loadings (F_x, M_x, M_y, M_z):

$$J_{ij} = \partial E_i / \partial F_j \qquad 2.12$$

This matrix correlates the known sectional loads to the measured strain; the system can then be solved (inverting the Jacobian) to calculate unknown sectional blade loads from an arbitrary strain field.

For verification of the method, the twisted beam in Figure 2.59 is subjected to a combined tip loading with a steady axial force and a steady torsion moment plus harmonic flap and lag bending moments. Two models have been considered, one as depicted in the figure, and another where the mesh is split at the same radial location (30% R) with a stiff joint used as a load sensor. Figure 2.60 compares the loads sensed at the joint for the split mesh (black symbols) to extracted sectional loads using nodes located at the principal axes (red solid line) and offset nodes (blue dashed line). It shows that the dynamic sectional loads are recovered exactly using the strain extraction method, independent of the selection of cross-section nodes. If the full Jacobian is not used, errors will appear in the predictions due to structural coupling caused by the beam twist, as shown in Figure 2.61.

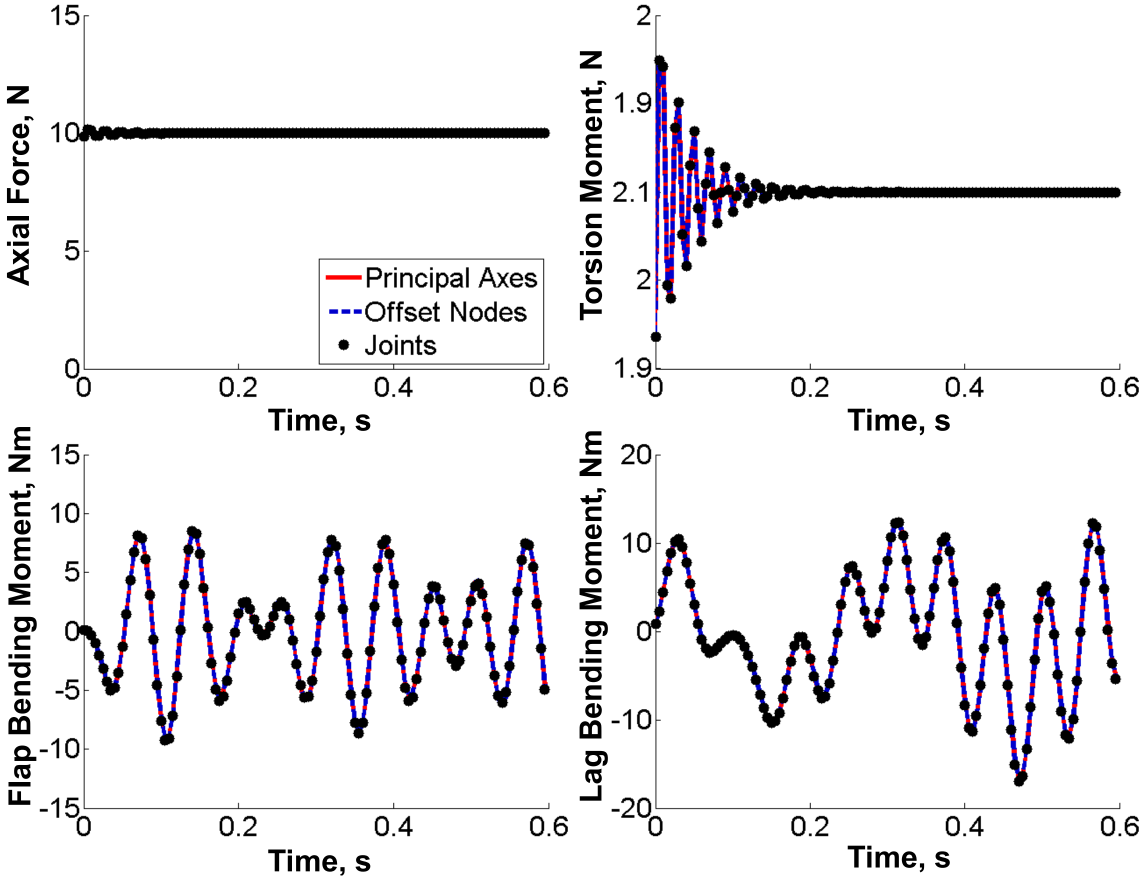


Figure 2.60 Dynamic sectional loads for a twisted beam undergoing combined tip loading with steady axial force and torsion moment and oscillatory flap and lag bending moments

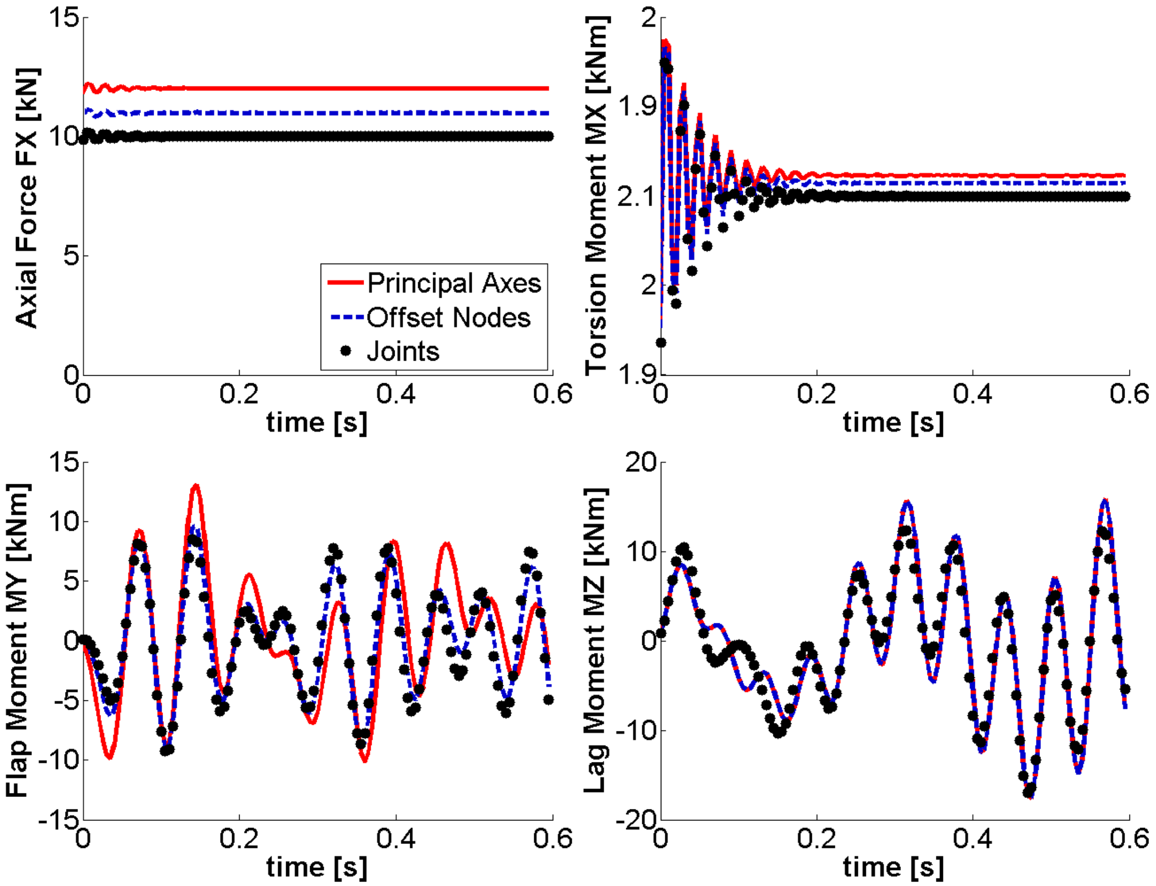


Figure 2.61 Dynamic sectional loads for a twisted beam undergoing combined tip loading with steady axial force and torsion moment and oscillatory flap and lag bending moments using only the diagonal elements of the Jacobian

Chapter 3: Component Verification and Validation

This chapter presents validation and verification of the X3D structural solver applied to composite beams and plates by comparing deformation and stress predictions to analytical or experimental results. The effect of mesh resolution is also examined for both beams and plates. This chapter also verifies the integration of multibody structural analysis models (SAM) developed using the methodology laid out in Chapter 2 with the solver using a notional bearingless rotor. Prior verification of the solver for predicting natural frequencies of static and rotating isotropic beams and static isotropic plates have been presented by Datta and Johnson [7, 74] and are omitted from this dissertation.

3.1 Composite Beams

3.1.1 Deformation, With and Without Homogenization

In this section, the composite homogenization method presented in Section 2.6.4.2 is validated by comparing deflection predictions with results from the Maryland composite beam tests [134]. Two composite box beams of identical dimensions are considered (Figure 3.1): one with a symmetric ply lay-up, resulting in a bending-torsion coupled beam; and another with an anti-symmetric lay-up, resulting in an extension-torsion coupled beam.

The beams have six layers of plies in each wall. Two models are constructed: in the first (Figure 3.2) every ply is resolved with a single layer of Hex27 elements with properties equivalent to the manufacturer listed ply properties (orthotropic), and in the second (Figure 3.3) a single layer of elements is used to model the entire laminate with (homogenized) effective material properties (anisotropic). The ply-resolved model has 192 cross-sectional elements – most of them occurring at the corners for nodal consistency – whereas the homogenized model has only 12. The effective properties are obtained in a pre-processing step from the ply properties, the layup angles, and layer thicknesses. In both models transverse anisotropy (X3D `ORTHTISO` material tag) is assumed as cross-ply properties were unavailable.

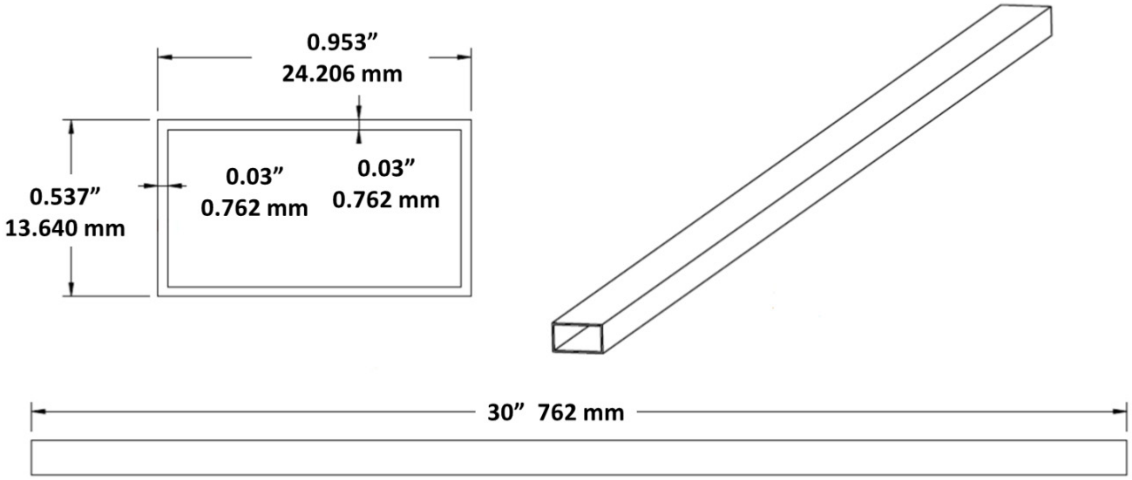


Figure 3.1 Schematic of the composite coupled box beams used for validation of the composite material modulus homogenization technique

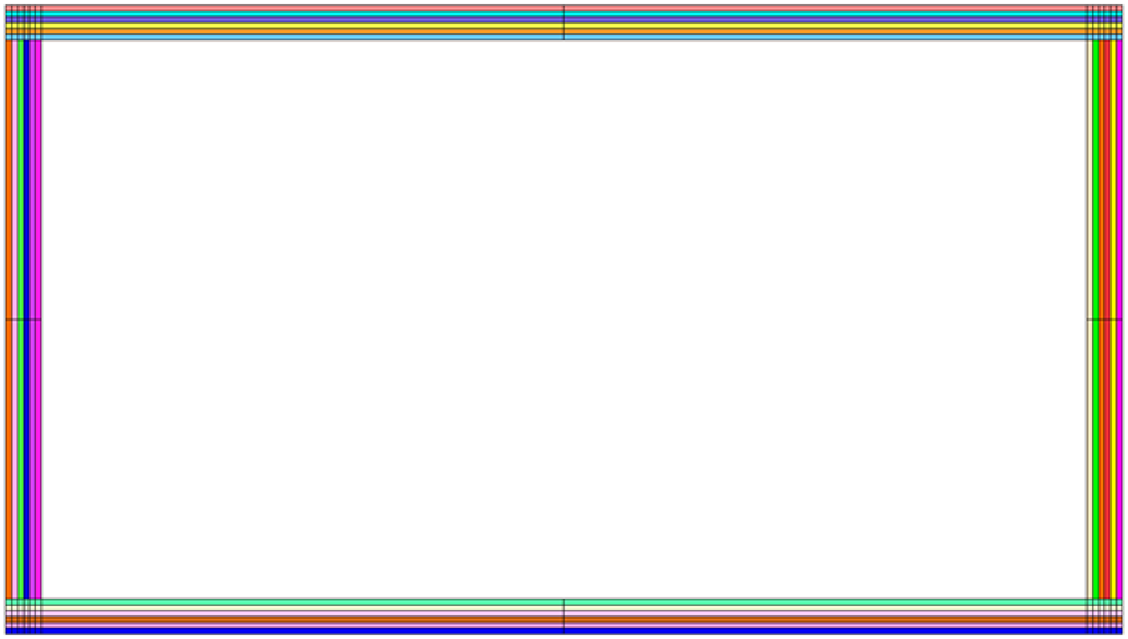


Figure 3.2 Ply-resolved composite beam mesh cross-section with one thickness-wise element per ply, resulting in 192 elements per cross-section

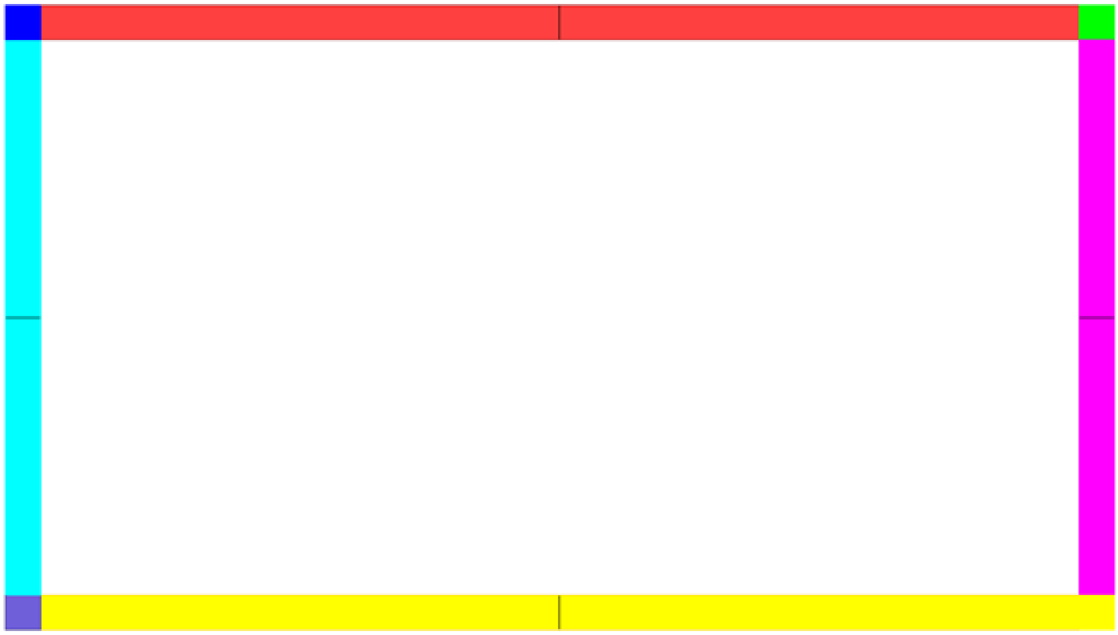


Figure 3.3 Homogenized composite beam mesh cross-section with a single element for all plies across thickness, resulting in 12 elements per cross-section

Homogenized and ply-resolved meshes with material properties for both types of coupled beams were subjected to static tip loading in X3D to produce a torsion response: the bending-torsion beam received a transverse bending force at the tip and the extension-torsion beam received an axial extension force. Predicted deformations for these cases are provided in Figure 3.4 and Figure 3.5 respectively for an increasing set of ply angles, with both homogenized and ply-resolved materials. The effective modulus model (dashed lines) shows close agreement with the ply-resolved model (solid lines) which indicates that the simplification of the mesh yields nearly identical results for deformations. Discrepancies between the two values could possibly be due to the assumptions of transverse anisotropy and other inherent assumptions of the method.

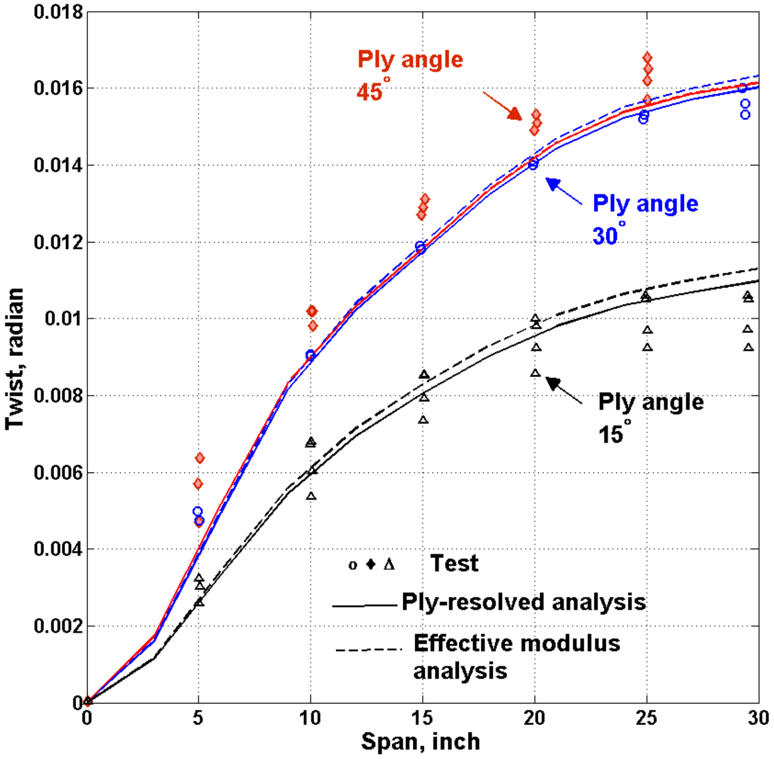


Figure 3.4 Composite bending-torsion coupled beam tip twist due to tip bending load

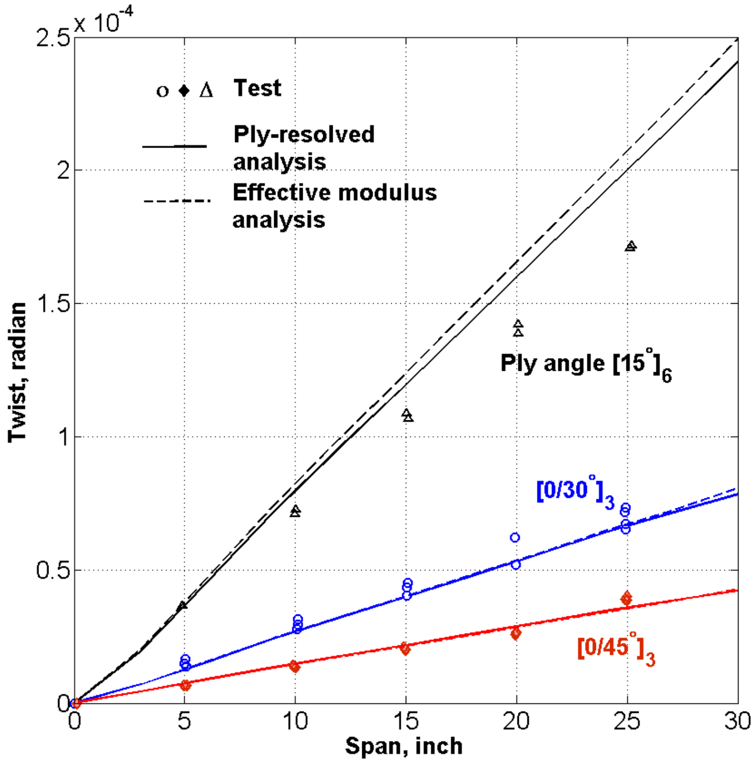


Figure 3.5 Composite extension-torsion coupled beam tip twist due to tip axial load

3.1.2 Composite Beam Stress

A beam of the same dimensions to that depicted in Figure 3.1 was also examined for stress through the thickness of the webs and caps. Unlike the deformation results, the beam used for stress validation did not have composite coupling; layups for both caps and both webs consisted of alternating 0° and 90° plies ($[0/90]_3$). Three meshes were considered, all of which are ply resolved, which is to say the boundaries between the six plies of the composite laminate are represented in the mesh and no homogenization is used. 3-D FEM predictions are presented for three plate meshes: one with 6 elements through the wall thickness (one element layer per ply), one with 12 elements (two per ply), and one with 18 elements (three per ply), thus the meshes have 13, 25, and 37 nodes through the walls, respectively, due to the use of Hex27 elements.

Because no experimental measurements of the stress in the beam are available, verification against a 1-D composite beam analysis is presented instead. Because the aspect ratio of the beam is greater than 30 and their being no spanwise discontinuities, beam analysis is expected to produce accurate results away from the ends of the beams, and hence provides the basis for verifying the 3-D analysis, at least for the dominant stresses. For this comparison, beam results were generated using DYMORE, a multibody rotor analysis package featuring state-of-the-art models of composite beams. DYMORE generates a 3-D stress field by applying the beam loads at a section to a stiffness matrix generated using FEM applied to the 2-D cross-section. Stress predictions using DYMORE and X3D are compared at a cross-section at the beam mid-span through the thickness of the top cap and through the thickness of the front web, as shown in Figure 3.6.

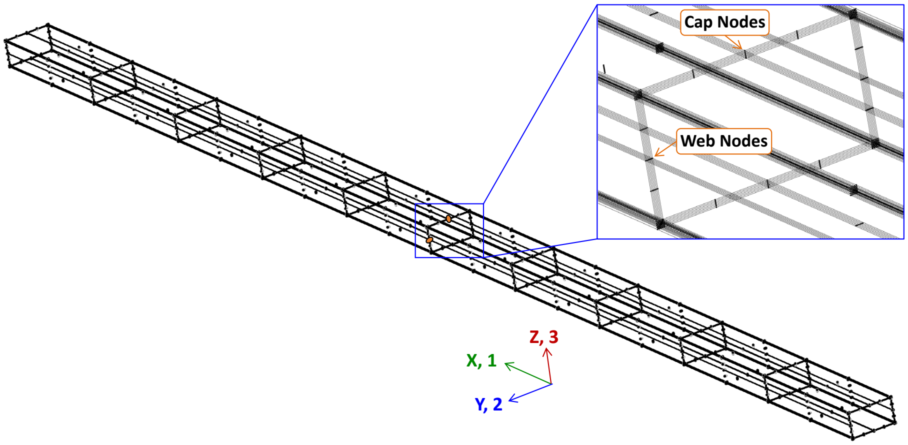


Figure 3.6 An X3D mesh used for composite beam stress analysis with six elements through the thickness, highlighting the nodes selected for stress plotting

Figure 3.7 shows the stress through the top cap of the beam at its center caused by an extensional tip load (F_x). In the extensional loading case, the dominant stress is the axial component, σ_{11}. As expected, the pattern shows stress that is high (about 150 kPa) in the stiff 0° oriented plies, and lower (about 10 kPa) in the 90° cross plies. All plies of a given orientation have the exact same stress, because axial loading results in evenly distributed stress throughout the cross-section. The 3-D predictions exactly match those from the beam analysis, although the treatment of boundaries between the two plies is different. Depending on which side of the boundary is evaluated, the stress experiences a jump (strains must be continuous). DYMORE simply reports one side of the jump; X3D averages the two sides of the step for the node that falls on the interface. The result is a jagged zig-zag mesh for the X3D case with 6 elements through the cap (one element per ply) because every-other node is a ply interface. When the mesh is further resolved, the steps are properly captured. Note that Figure 3.7 plots straight lines between the X3D prediction points: this graph does not use the proper higher order shape functions associated with Hex27 elements. If it did, the lines connecting the X3D nodes would have a smooth quadratic curvature, and therefore approximate the step even better.

In-plane transverse stress due to Poisson's effect (σ_{22}) is the next strongest, but magnitudes are significantly lower, on the order of ± 5 kPa. Once again, the 3-D stress predictions for the finer meshes match with those from the beam analysis. Differences appear in the out-of-plane stress (σ_{33}) term. DYMORE assumes it is zero, but X3D predicts a small value (less than 0.1% of the max stress) as the interior of the wall is approached due to 3-D effects. Shear stresses are also exactly zero in DYMORE whereas X3D predicts negligibly small shear stress. The shear stress (σ_{13}) is the only stress that does not converge with mesh size, although the magnitudes are too low to be of practical significance.

For the same tip extension loading case, Figure 3.8 shows the stress through the thickness of the web. The loading patterns are identical to those in the cap, with strong axial stress (σ_{11}) and some transverse stress (σ_{33}), and minimal values for other stresses. As before, 3-D FEM predictions match beam predictions well, and the shear stress (σ_{13}) is the only stress that does not converge with mesh size, though its magnitude is too low to be of practical significance.

Changing to a bending case, Figure 3.9 shows stress through the top cap of the beam at its center caused by a vertical tip load (F_z). Here the axial stress in the top cap is under compression, indicated by negative stress. Unlike the axial case, the stress through the top cap is no longer uniform. Instead, it is highest at the surface ($z/t = 1$) and reduces towards the neutral axis at the center of the box. The bottom cap would show the reverse pattern, with positive tensile stress. As with the axial results, normal and in-plane transverse stress predictions from 3-D are well matched to beam analysis, though boundaries are treated differently. There is a large difference in prediction of shear stress σ_{13} between the two analyses. The beam analysis shows zero stress, whereas 3-D FEM predicts shear stress in the upper cap that varies from ply to ply, and a maximum magnitude at -85.5 Pa. Other components of stress also show differences between the two analyses, but those values are much less significant. As with other stress components that show differences between plies, the boundaries are more distinctly resolved in the finer meshes.

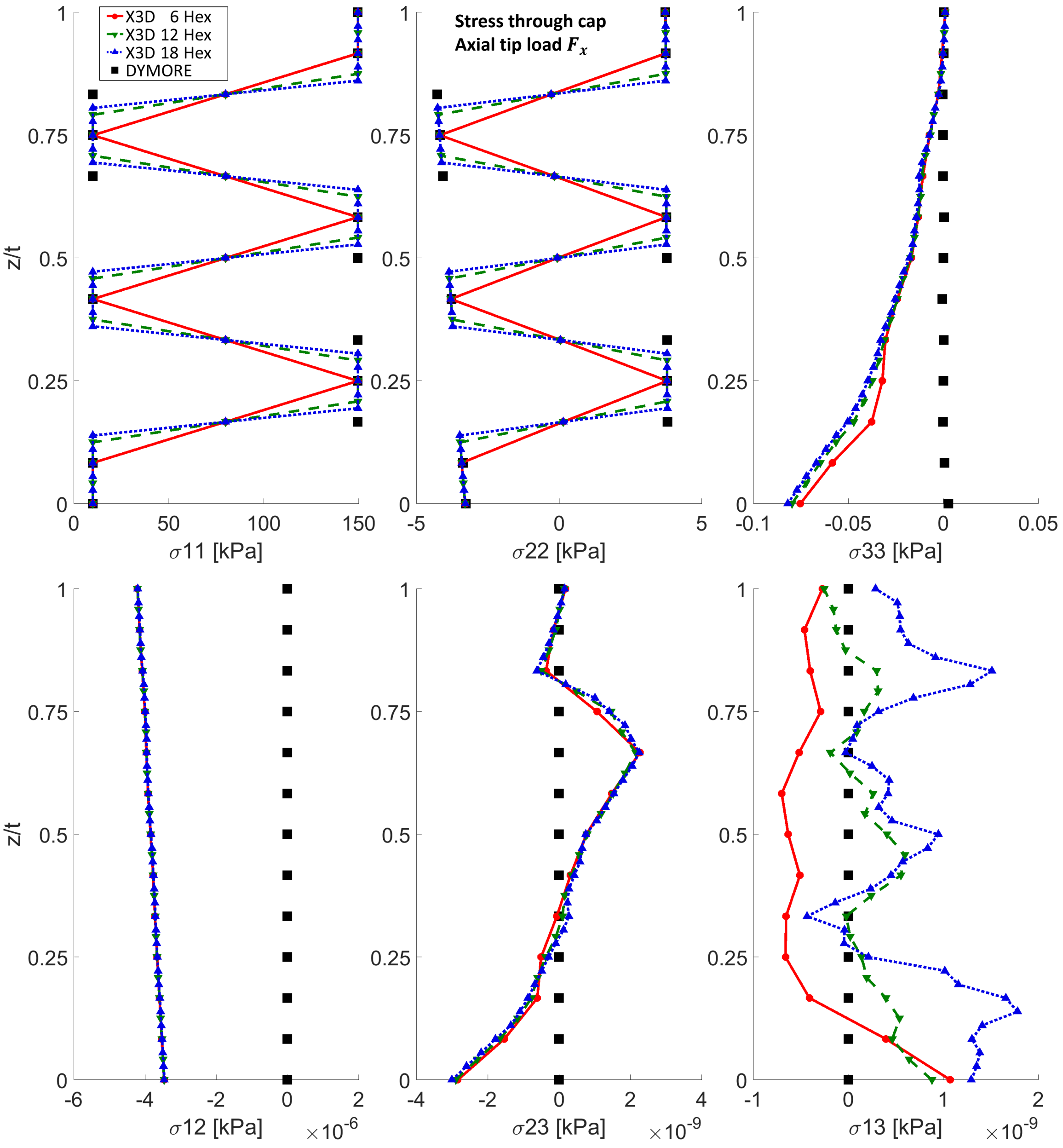


Figure 3.7 Stress in the top cap of a composite box beam (Figure 3.6) undergoing an extensional tip force (F_x) with three mesh resolutions through the thickness of the walls; the vertical axis represents position in the z-axis normalized by the thickness of the spar walls: $z/t = 1$ represents the outside surface of the cap, and $z/t = 0$ represents the inside surface

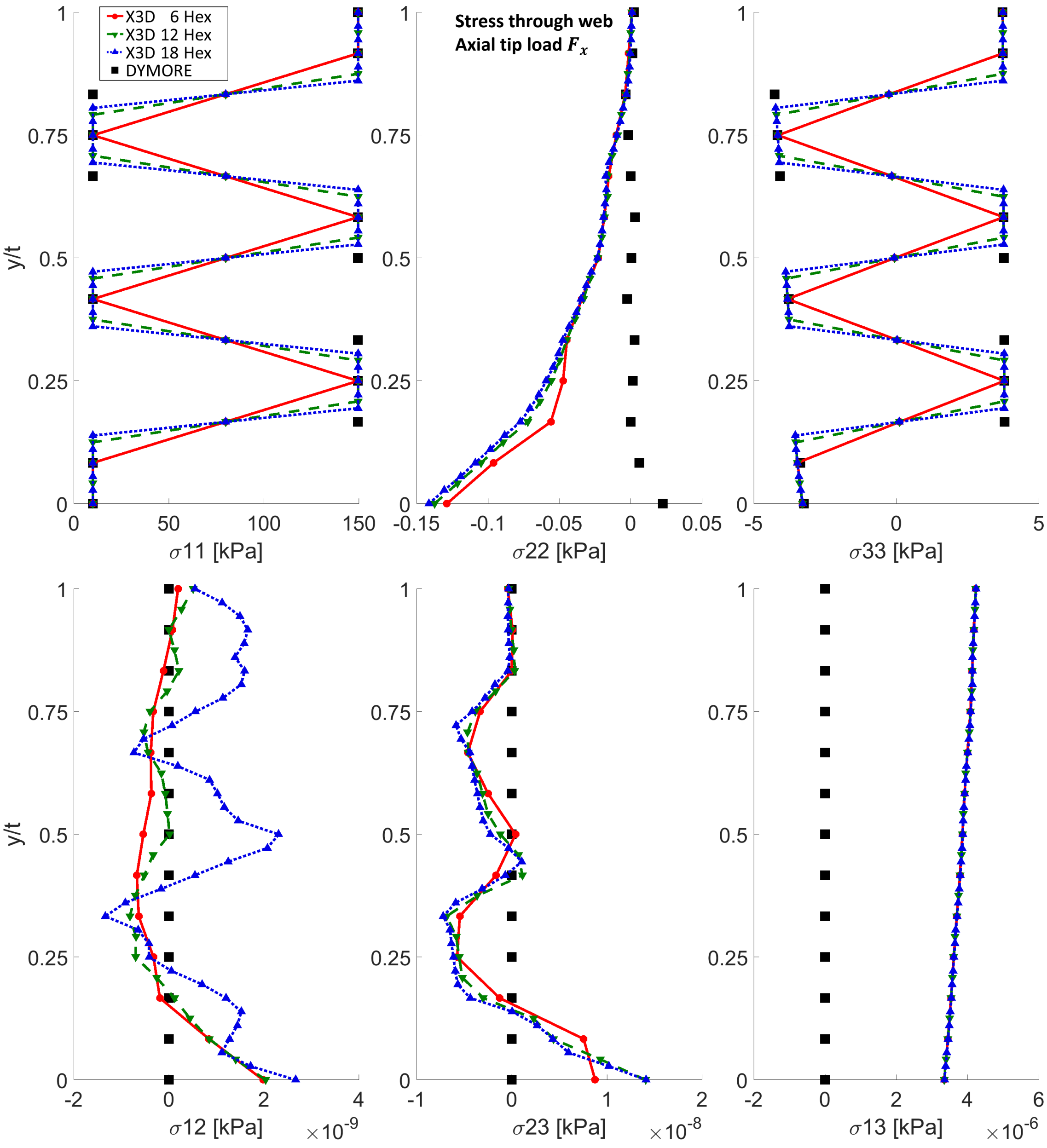


Figure 3.8 Stress in the front web of a composite box beam (Figure 3.6) undergoing an extensional tip force (F_x) with three mesh resolutions through the thickness of the walls; the vertical axis represents position in the z-axis normalized by the thickness of the spar walls: $y/t = 1$ represents the outside surface of the cap, and $y/t = 0$ represents the inside surface

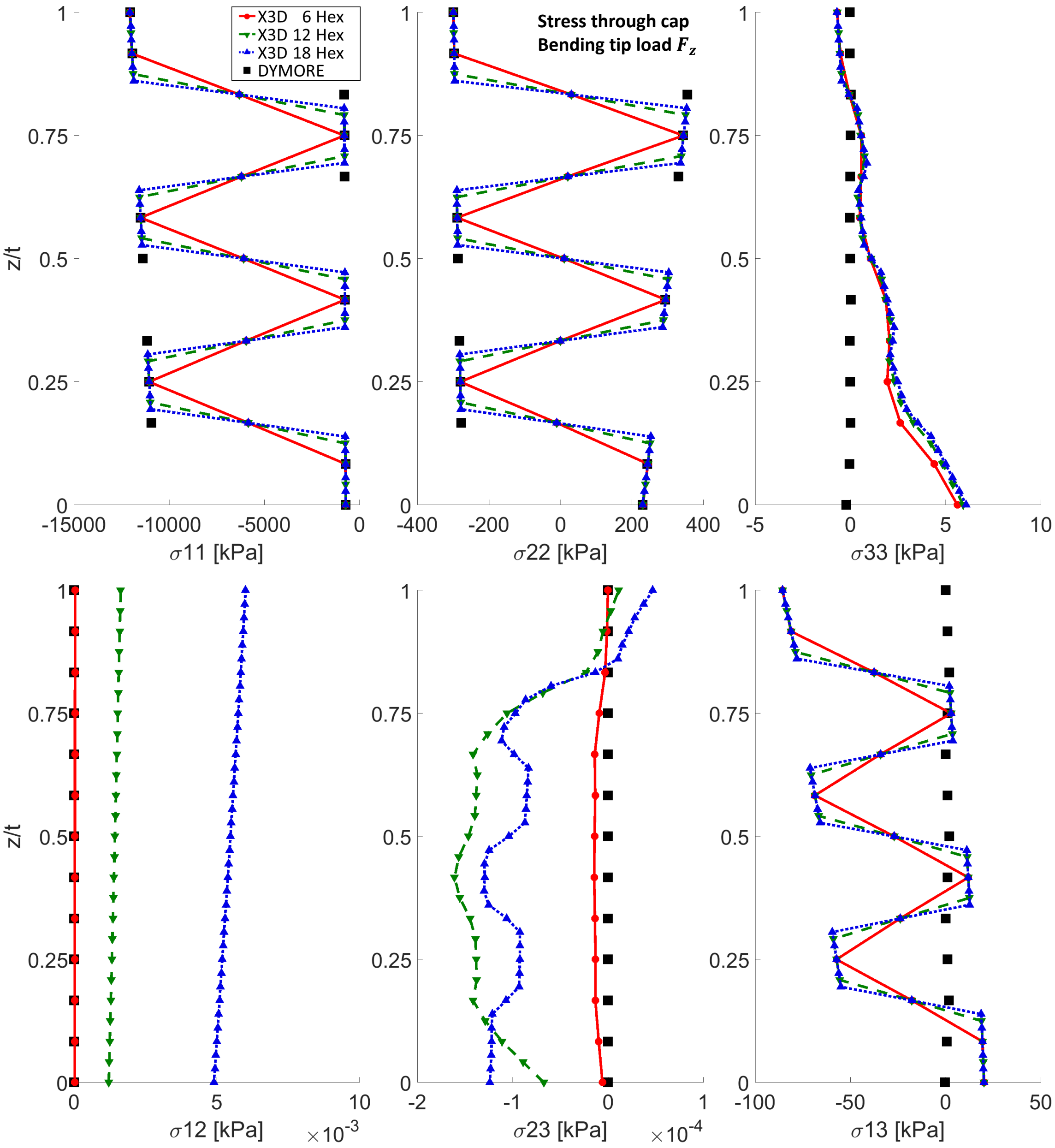


Figure 3.9 Stress in the top cap of a composite box beam (Figure 3.6) undergoing a vertical bending tip force (F_z) with three mesh resolutions through the thickness of the walls; the vertical axis represents position in the z-axis normalized by the thickness of the spar walls: $z/t = 1$ represents the outside surface of the cap, and $z/t = 0$ represents the inside surface

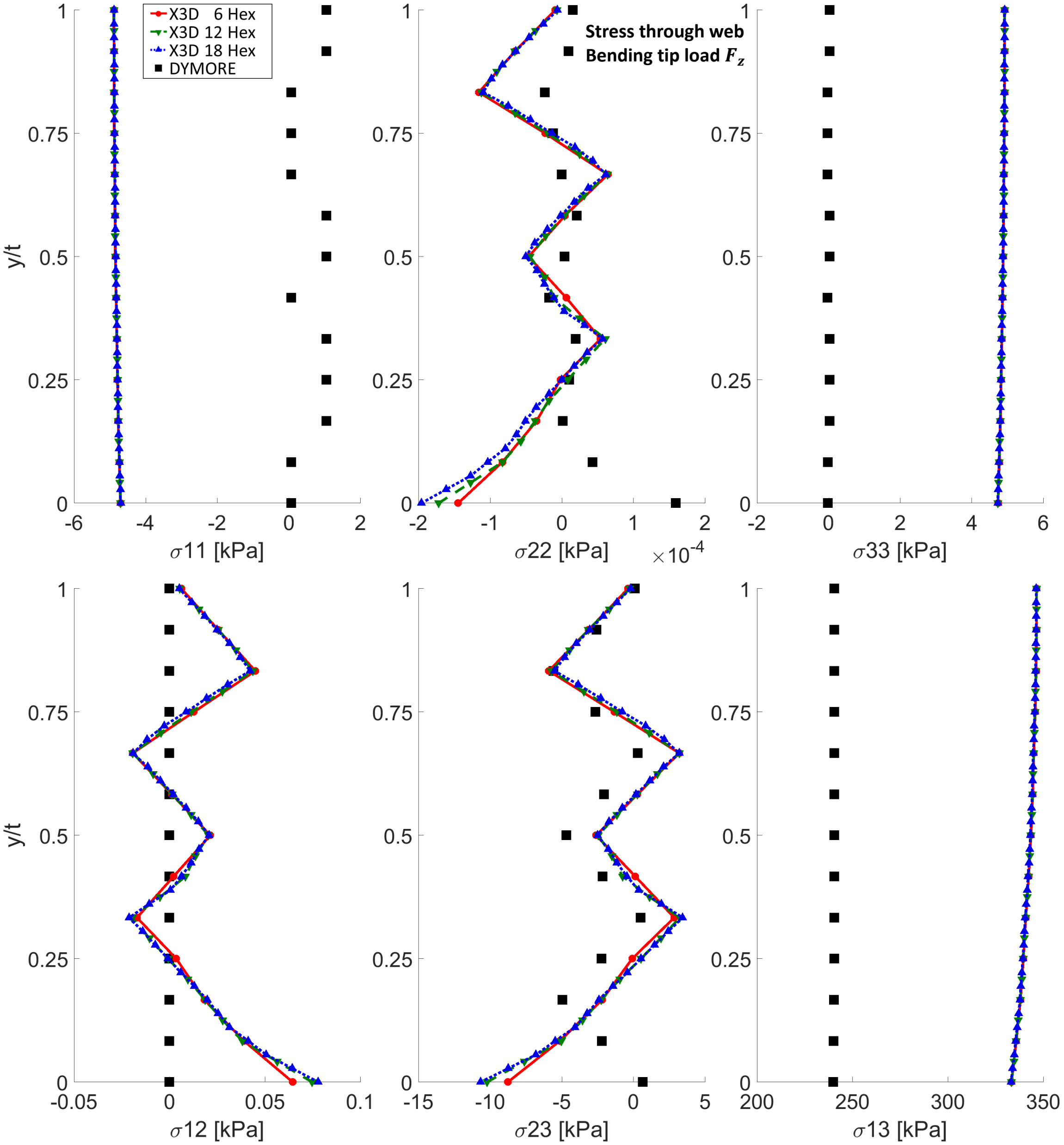
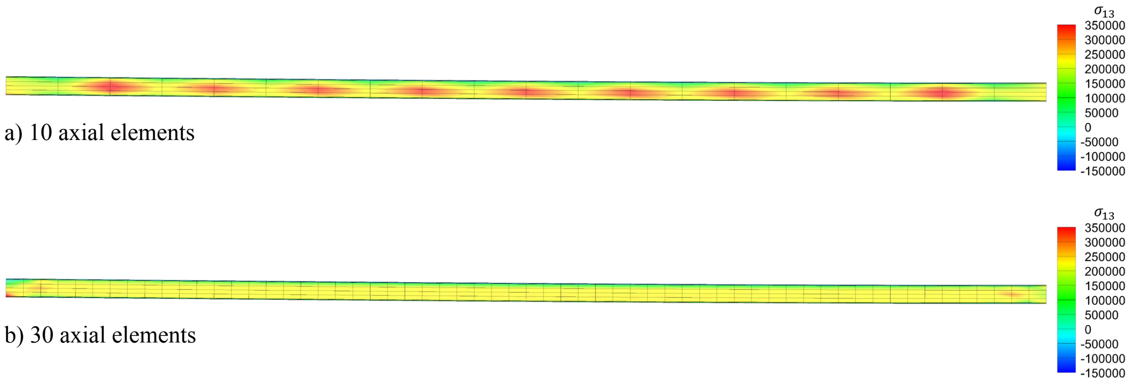


Figure 3.10 Stress in the front web of a composite box beam (Figure 3.6) undergoing a vertical bending tip force (F_z) with three mesh resolutions through the thickness of the walls; the vertical axis represents position in the z-axis normalized by the thickness of the spar walls: $y/t = 1$ represents the outside surface of the cap, and $y/t = 0$ represents the inside surface

Finally, Figure 3.10 shows stress through the center of the web caused by the same vertical tip load. Here, disagreements in predictions from 3-D and beam analyses appear in all stress components. Most of the stresses are very small in magnitude, so differences in the predictions are not critical. The shear stress σ_{23}, however, is not negligible and shows both a magnitude and sign difference at the inside surface of the beam web, despite both analyses agreeing on the outer surface. Shear stress σ_{13} shows a large difference between 3-D and beam predictions, with 3-D stresses about 40% higher than what is predicted by the beam analysis. The 3-D predictions of stress do not vary with mesh resolution through the cap.

To investigate the source of the large difference in shear stress σ_{13} in the web, the overall stress field predicted by 3-D FEM is shown in Figure 3.11. With 10 axial elements in the mesh, the stress pattern has axial oscillations along the center line, with magnitudes varying from highs of approximately 350 kPa (including at the center section where the predictions in Figure 3.10 were taken) to lows of approximately 200 kPa (lower than the beam prediction of 240.5 kPa). Increasing the radial number of elements in the mesh from 10 to 30 resolves these spatial oscillations.



a) 10 axial elements

b) 30 axial elements

Figure 3.11 X3D predictions of shear stress σ_{13} in the web of a composite beam undergoing vertical tip bending using a mesh with (a) 10 and (b) 30 axial elements

Figure 3.12 shows the stress in the cap of the beam under a static vertical tip bending load with three meshes. All meshes have only 6 elements through the thickness of the four beam walls, or one element per ply, but the axial mesh resolution increases from 10 to 20 to 30 elements. Changing the axial mesh resolution has very little effect on predictions for stress in the cap, though there is a slight change in the magnitude of σ_{13}. The stress in the web (Figure 3.13), on the other hand, shows significant improvements in the prediction of the shear stress σ_{13}, with values converging to those predicted by the beam analysis. This suggests radial mesh resolution can be important to accurate shear stress predictions in 3-D FEM analysis of slender structures.

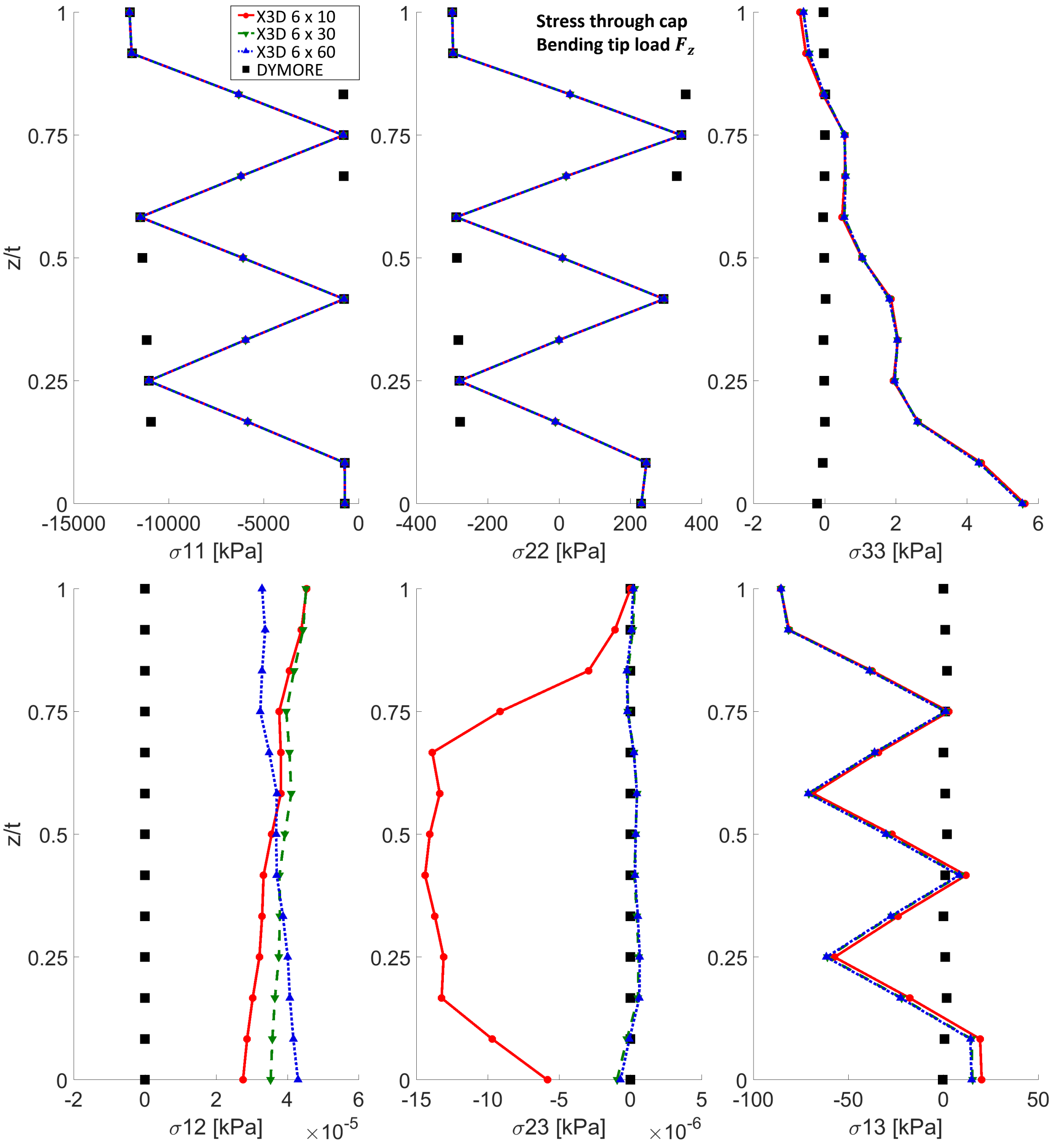


Figure 3.12 Stress in the top cap of a composite box beam (Figure 3.6) undergoing a vertical bending tip force (F_z) with three different axial mesh resolutions; the vertical axis represents position in the z-axis normalized by the thickness of the spar walls: $z/t = 1$ represents the outside surface of the cap, and $z/t = 0$ represents the inside surface

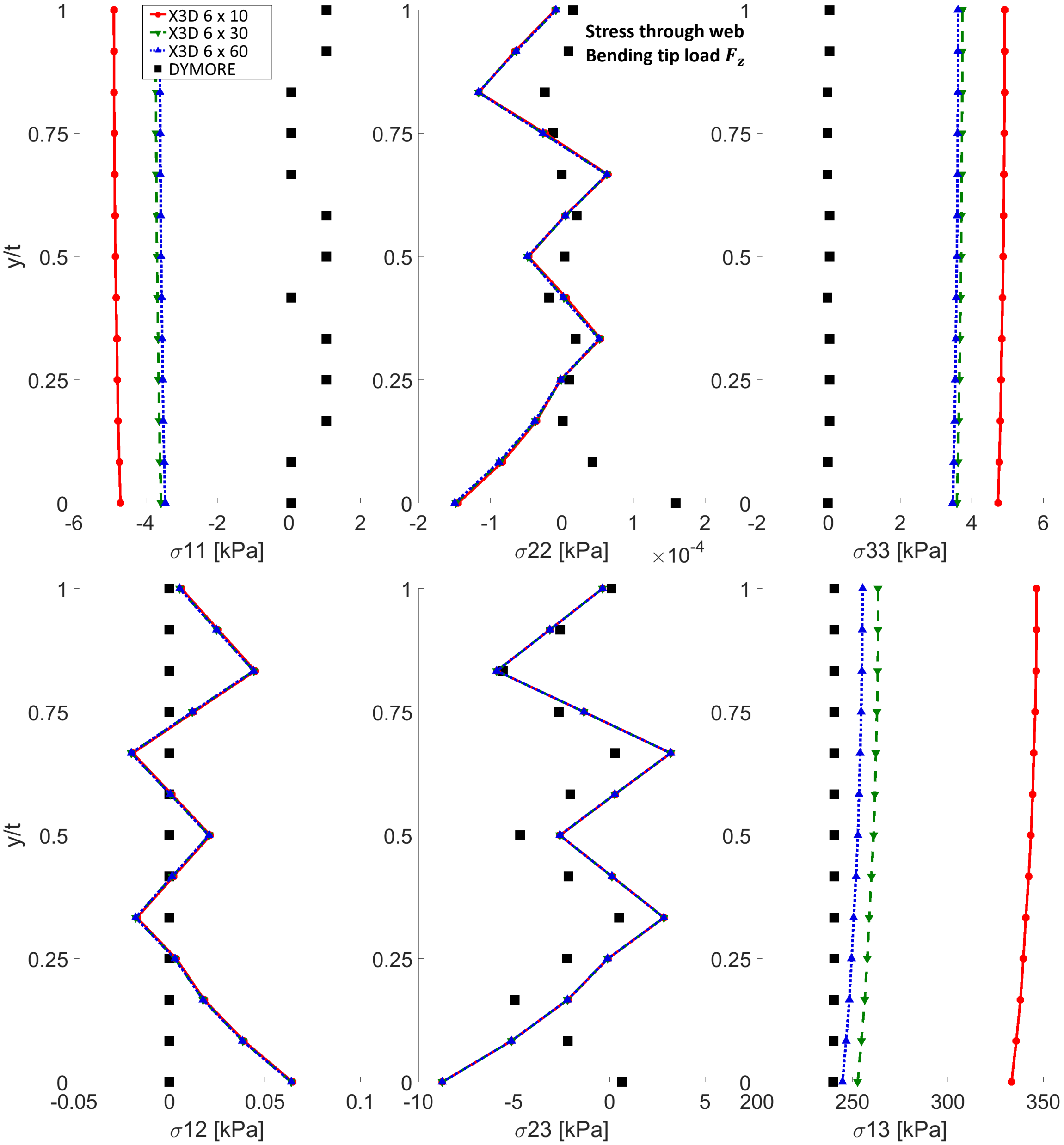


Figure 3.13 Stress in the front web of a composite box beam (Figure 3.6) undergoing a vertical bending tip force (F_z) with three different axial mesh resolutions; the vertical axis represents position in the z-axis normalized by the thickness of the spar walls: $y/t = 1$ represents the outside surface of the wall, and $y/t = 0$ represents the inside surface

3.2 Composite Plate Stress

This section validates the 3-D solver for stress predictions inside a thick composite plate. The sample problem and the laminated plate analysis used for validation are based on Fan and Ye [135]. Results show good agreement between 3-D predictions and the analytical solution. The publication provides sparse documentation of the problem and methodology, so they are re-derived as part of this work and documented in this section.

3.2.1 Sample Problem

The sample problem is a simply supported plate under normal surface pressure [135]. The example in the paper is non-dimensional, but dimensions are assigned here for clarity. The plate considered (Figure 3.14) has a square surface measuring 1 m by 1 m and a thickness of 0.1 m. The plate is composed of three isotropic plies: two identical plies on the top and bottom have a thickness of 0.01 m, and the central ply has a thickness of 0.08 m.

The plate is simply supported on its four sides. Specifically,

$$\begin{aligned} &\text{for } x = 0, a\text{: } \sigma_{11} = 0;\;\; w = v = 0 \\ &\text{for } y = 0, b\text{: } \sigma_{22} = 0;\;\; w = u = 0 \end{aligned} \qquad 3.1$$

where a and b represent with width and length of the plates in the x and y directions (here, $a = b = 1$ m); u, v, and w are the plate displacement in the x, y, and z directions; and σ_{11} and σ_{22} are the stresses in the x and y directions, respectively. These special conditions allow for analytical solution of the stresses in the plate, using the methodology described below in Section 3.2.2.

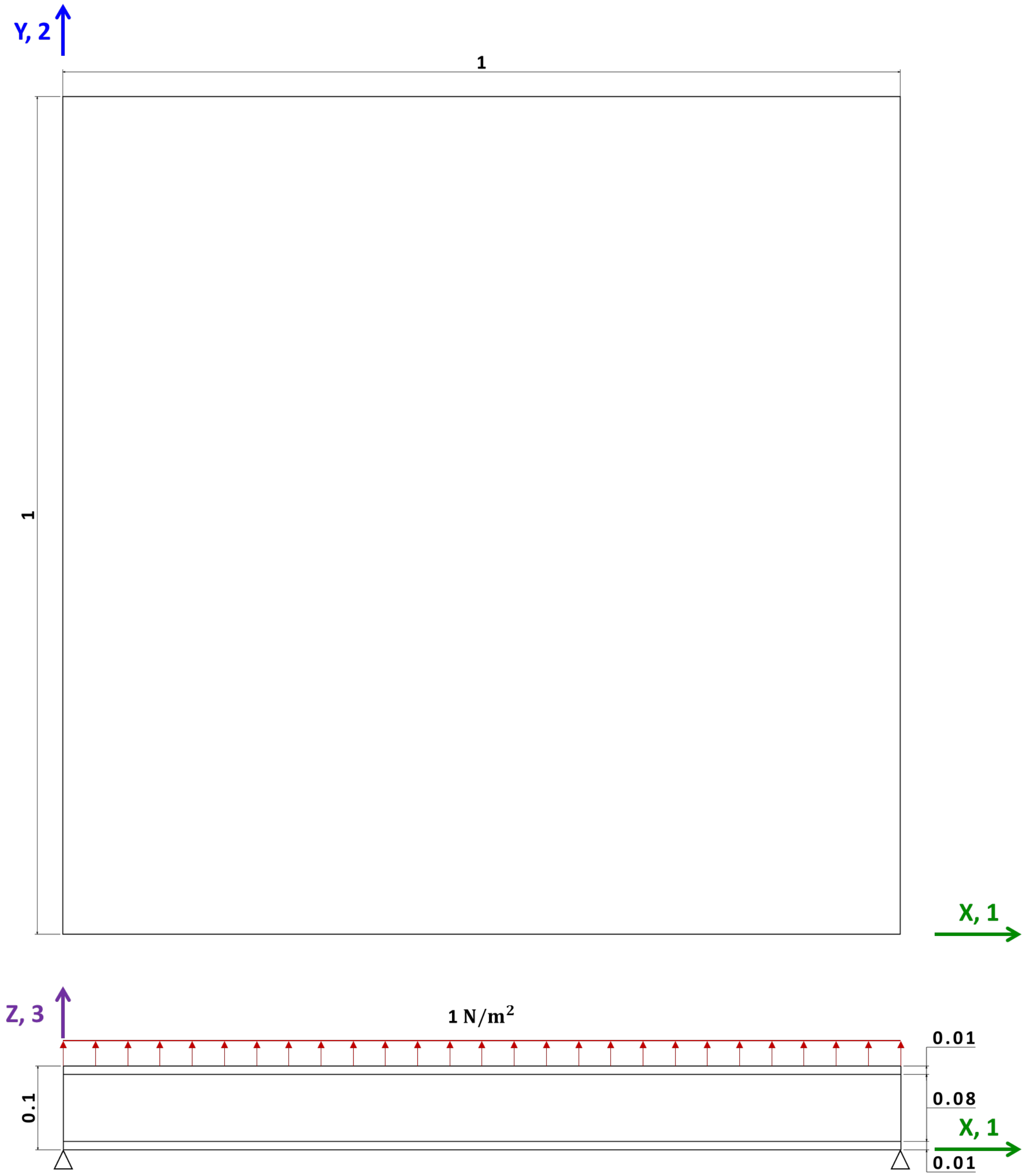


Figure 3.14 Schematic of the three ply simply supported plate with a unit normal surface pressure, dimensions in meters

The material properties of the each ply are provided in the form of a 6-by-6 stiffness matrix ($C^{(p)}$), where p is the ply number, normalized by the first entry C_{11}^{p}: for the 3-D analysis, a value of 1E4 N/m^2 is assigned to C_{11} for the top and bottom plates, $C_{11}^{(1)}$ and $C_{11}^{(3)}$. The stiffness properties of the middle ply are the same with respect to the first entry of the top and bottom plies, but inversely scaled by a constant γ:

$$\gamma = C_{11}^{(1)}/C_{11}^{(2)}. \tag{3.2}$$

Thus, for γ equal to one, all plies are identical and the plate is isotropic. If γ is greater than one, the middle ply is softer than the top and bottom plies by a factor of γ. The rest of the stiffness values for the three plies are summarized in Table 3.1. Results for four values of γ (1, 5, 10, 15) are presented in this section.

Table 3.1 Stiffness matrix values for each of the three plies of the thick plate used for stress validation. The first entry in the stiffness matrix is given for each ply and the remaining values are defined by the ratios that follow; any stiffness values not provided are zero

$C_{11}^{(1)}$	1E4 N/m^2
$C_{11}^{(2)}$	1E4 N/m^2
$C_{11}^{(3)}$	$C_{11}^{(1)}/\gamma$
$C_{22}^{(p)}/C_{11}$	0.543103
$C_{12}^{(p)}/C_{11}$	0.233190
$C_{23}^{(p)}/C_{11}$	0.098276
$C_{33}^{(p)}/C_{11}$	0.530172
$C_{13}^{(p)}/C_{11}$	0.010770
$C_{44}^{(p)}/C_{11}$	0.266810
$C_{55}^{(p)}/C_{11}$	0.159914
$C_{66}^{(p)}/C_{11}$	0.262930

3.2.2 Analysis Method

The analysis method presented by Fan and Ye is based on the method of initial functions [136, 137] and begins by examining the stress-strain relationship for an individual ply, j:

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{Bmatrix}_j = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{13} & C_{23} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix}_j \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{Bmatrix}_j . \qquad 3.3$$

Here, σ and ϵ represent stress and strain components in the axis directions given in Figure 3.15. The in-plane stress and strain terms (σ_{11}, σ_{22}, σ_{12} and corresponding ϵ) can be eliminated using strain-displacement and equilibrium equations [137], giving the following differential equations in the spatial variables for a given ply:

$$\begin{gathered} \frac{\partial u}{\partial z} = \frac{\sigma_{13}}{C_{55}} - \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial z} = \frac{\sigma_{23}}{C_{44}} - \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial z} = -\frac{C_{13}}{C_{33}}\frac{\partial u}{\partial x} - \frac{C_{23}}{C_{33}}\frac{\partial v}{\partial y} + \frac{1}{C_{33}}\sigma_{33} \\ \frac{\partial \sigma_{13}}{\partial z} = \rho\frac{\partial^2 u}{\partial t^2} - \left(C_{11} - \frac{C_{13}^2}{C_{33}}\right)\frac{\partial^2 u}{\partial x^2} - C_{66}\frac{\partial^2 u}{\partial y^2} - \left(C_{12} - \frac{C_{13}C_{12}}{C_{33}} + C_{66}\right)\frac{\partial^2 v}{\partial x \partial y} - \frac{C_{13}}{C_{33}}\frac{\partial \sigma_{33}}{\partial x} \\ \frac{\partial \sigma_{23}}{\partial z} = \rho\frac{\partial^2 v}{\partial t^2} - \left(C_{11} - \frac{C_{13}^2}{C_{33}}\right)\frac{\partial^2 v}{\partial y^2} - C_{66}\frac{\partial^2 v}{\partial x^2} - \left(C_{12} - \frac{C_{13}C_{12}}{C_{33}} + C_{66}\right)\frac{\partial^2 u}{\partial x \partial y} - \frac{C_{23}}{C_{33}}\frac{\partial \sigma_{33}}{\partial y} \\ \frac{\partial \sigma_{33}}{\partial z} = -\frac{\partial \sigma_{13}}{\partial x} - \frac{\partial \sigma_{23}}{\partial y} + \frac{\rho\partial^2 w}{\partial t^2} \end{gathered} \qquad 3.4$$

These can be combined with equation 3.3 to yield the state equation:

$$\frac{\partial}{\partial z}\left\{\begin{matrix} u \\ v \\ \sigma_{33} \\ \sigma_{13} \\ \sigma_{23} \\ w \end{matrix}\right\}_j = \begin{bmatrix} 0 & 0 & 0 & a_{11} & 0 & -\alpha \\ 0 & 0 & 0 & 0 & a_{22} & -\beta \\ 0 & 0 & 0 & -\alpha & -\beta & \xi^2 \\ \xi^2 - C_2\alpha^2 - C_6\beta^2 & -(C_3 + C_6)\alpha\beta & C_1\alpha & 0 & 0 & 0 \\ -(C_3 + C_6)\alpha\beta & \xi^2 - C_6\alpha^2 - C_4\beta^2 & C_5\beta & 0 & 0 & 0 \\ C_1\alpha & C_5\beta & C_{10} & 0 & 0 & 0 \end{bmatrix}_j \left\{\begin{matrix} u \\ v \\ \sigma_{33} \\ \sigma_{13} \\ \sigma_{23} \\ w \end{matrix}\right\}_j \qquad 3.5$$

In this equation, several terms are combined to simplify the formulation:

$$C_1 = -\frac{C_{13}}{C_{33}}, \quad C_2 = C_{11} - \frac{C_{13}^2}{C_{33}}, \quad C_3 = C_{12} - \frac{C_{13}C_{23}}{C_{33}}, \quad C_4 = C_{22} - \frac{C_{23}^2}{C_{33}}$$

$$C_5 = -\frac{C_{23}}{C_{33}}, \quad C_6 = C_{66}, \quad C_{10} = \frac{1}{C_{33}}, \quad a_{11} = \frac{1}{C_{55}}, \quad a_{22} = \frac{1}{C_{44}} \qquad 3.6$$

$$\alpha = \frac{\partial}{\partial x}, \quad \beta = \frac{\partial}{\partial y}, \quad \xi^2 = \rho\frac{\partial^2}{\partial t^2}$$

Next, the distribution of displacement within the plate can be assumed to be represented by an infinite Fourier series which satisfies the boundary conditions given in equation 3.1:

$$\left\{\begin{matrix} u \\ v \\ w \end{matrix}\right\}_j = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\left\{\begin{matrix} u_{mn}(z)\,cos\left(\frac{m\pi x}{a}\right) sin\left(\frac{n\pi y}{b}\right) \\ v_{mn}(z)\,sin\left(\frac{m\pi x}{a}\right) cos\left(\frac{n\pi y}{b}\right) \\ w_{mn}(z)\,sin\left(\frac{m\pi x}{a}\right) sin\left(\frac{n\pi y}{b}\right) \end{matrix}\right\}_j e^{i\omega_{mn}t} \qquad 3.7$$

In this equation z is the thickness-wise position within ply j, a and b represent the width and length of the plate (both 1 m), t and ω_{mn} are the time variable and temporal frequency for dynamic loading, and the coefficients u_{mn}, v_{mn}, and w_{mn} are unknowns to be solved. Substituting equation 3.7 into equation 3.5 and solving for stress yields a similar expression for the stress components:

$$\begin{Bmatrix}\sigma_{13}\\ \sigma_{23}\\ \sigma_{33}\end{Bmatrix}_j = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\begin{Bmatrix}\sigma_{13_{mn}}(z)\,cos\left(\frac{m\pi x}{a}\right)sin\left(\frac{n\pi y}{b}\right)\\ \sigma_{23_{mn}}(z)\,sin\left(\frac{m\pi x}{a}\right)cos\left(\frac{n\pi y}{b}\right)\\ \sigma_{33_{mn}}(z)\,sin\left(\frac{m\pi x}{a}\right)sin\left(\frac{n\pi y}{b}\right)\end{Bmatrix}_j e^{i\omega_{mn}t} \tag{3.8}$$

Substituting equations 3.7 and 3.8 into equation 3.5, for any given pair of summation indices m and n in ply j:

$$\frac{\partial}{\partial z}\begin{Bmatrix}u_{mn}(z)\\ v_{mn}(z)\\ \sigma_{33_{mn}}(z)\\ \sigma_{13_{mn}}(z)\\ \sigma_{23_{mn}}(z)\\ w_{mn}(z)\end{Bmatrix}_j = \begin{bmatrix}0 & A_{mn}\\ B_{mn} & 0\end{bmatrix}_j \begin{Bmatrix}u_{mn}(z)\\ v_{mn}(z)\\ \sigma_{33_{mn}}(z)\\ \sigma_{13_{mn}}(z)\\ \sigma_{23_{mn}}(z)\\ w_{mn}(z)\end{Bmatrix}_j \tag{3.9}$$

$$A_{mn} = \begin{bmatrix}a_{11} & 0 & -\zeta\\ 0 & a_{22} & -\eta\\ \zeta & \eta & -\rho\omega^2\end{bmatrix} \quad B_{mn} = \begin{bmatrix}-\rho\omega^2 - C_2\zeta^2 - C_6\eta^2 & -(C_3 + C_6)\alpha\beta & C_1\zeta\\ (C_3 + C_6)\zeta\eta & -\rho\omega^2 + C_6\zeta^2 + C_4\eta^2 & C_5\eta\\ -C_1\zeta & -C_5\eta & C_{10}\end{bmatrix}$$

$$\zeta = \frac{m\pi}{a} \quad \eta = \frac{n\pi}{b} \quad \omega = \omega_{mn}$$

For static loading cases, the matrices A_{mn} and B_{mn} depend only on the geometry and material properties of the ply. Equation 3.9 can be solved for the deformation and stress coefficients (u_{mn}, $\sigma_{13_{mn}}$, etc.), which are functions only of position through the ply thickness, z. The solution to this differential equation is:

$$\begin{Bmatrix}u_{mn}(z)\\ v_{mn}(z)\\ \sigma_{33_{mn}}(z)\\ \sigma_{13_{mn}}(z)\\ \sigma_{23_{mn}}(z)\\ w_{mn}(z)\end{Bmatrix}_j = \exp\left(z\begin{bmatrix}0 & A_{mn}\\ B_{mn} & 0\end{bmatrix}_j\right)\begin{Bmatrix}u_{mn}(0)\\ v_{mn}(0)\\ \sigma_{33_{mn}}(0)\\ \sigma_{13_{mn}}(0)\\ \sigma_{23_{mn}}(0)\\ w_{mn}(0)\end{Bmatrix}_j \tag{3.10}$$

Define: $$R_j(z) = \begin{Bmatrix} u_{mn}(z) \\ v_{mn}(z) \\ \sigma_{33_{mn}}(z) \\ \sigma_{13_{mn}}(z) \\ \sigma_{23_{mn}}(z) \\ w_{mn}(z) \end{Bmatrix} \qquad D_j(z) = \exp\left(z \begin{bmatrix} 0 & A_{mn} \\ B_{mn} & 0 \end{bmatrix}_j \right)$$

So, for a given ply j:

$$R_j(z) = D_j(z) R_j(0) \qquad 3.11$$

This equation relates the stress and displacement coefficients R_j at a thickness position z within ply j to their values at the face of the ply, $z = 0$. These coefficients must be continuous at the ply interfaces:

$$R_j(h_j) = R_{j+1}(0) \qquad 3.12$$

where h_j is the thickness of the ply. Thus equation 3.11 can be repeated for a stack of plies:

$$R_1(h_1) = D_1(h_1)R_1(0) = R_2(0)$$

$$R_2(h_2) = D_2(h_2)R_2(0) = D_2(h_2)D_1(h_1)R_1(0)$$

$$\dots$$

$$R_p(h_p) = \left(\prod_{j=p}^{1} D_j\left(h_j\right) \right) R_1(0) \qquad 3.13$$

Thus the coefficients for displacement and stress can be solved at a given position z in ply p, if the coefficients at the bottom of the laminate, $R_1(0)$, are known. Those coefficients can be solved for by applying equation 3.13, through the entire laminated plate to relate the top surface of the plate, where the load is applied, to the bottom surface:

$$\begin{Bmatrix} u_{mn}(h) \\ v_{mn}(h) \\ \sigma_{33mn}(h) \\ \sigma_{13mn}(h) \\ \sigma_{23mn}(h) \\ w_{mn}(h) \end{Bmatrix} = \prod_{j=p}^{1} D_j(h_j) \begin{Bmatrix} u_{mn}(0) \\ v_{mn}(0) \\ \sigma_{33mn}(0) \\ \sigma_{13mn}(0) \\ \sigma_{23mn}(0) \\ w_{mn}(0) \end{Bmatrix} \qquad 3.14$$

where h is the thickness of the entire laminated plate. Using the third, fourth, and fifth equations from this system, the stress coefficients at the top of the plate can be solved in terms of the coefficients at the bottom of the plate:

$$\begin{Bmatrix} u_{mn}(h) \\ v_{mn}(h) \\ \sigma_{33mn}(h) \\ \sigma_{13mn}(h) \\ \sigma_{23mn}(h) \\ w_{mn}(h) \end{Bmatrix} = \begin{bmatrix} \overline{D}_{11} & \overline{D}_{12} & \overline{D}_{13} & \overline{D}_{14} & \overline{D}_{15} & \overline{D}_{16} \\ \overline{D}_{21} & \overline{D}_{22} & \overline{D}_{23} & \overline{D}_{24} & \overline{D}_{25} & \overline{D}_{26} \\ \overline{D}_{31} & \overline{D}_{32} & \overline{D}_{33} & \overline{D}_{34} & \overline{D}_{35} & \overline{D}_{36} \\ \overline{D}_{41} & \overline{D}_{42} & \overline{D}_{43} & \overline{D}_{44} & \overline{D}_{45} & \overline{D}_{46} \\ \overline{D}_{51} & \overline{D}_{52} & \overline{D}_{53} & \overline{D}_{54} & \overline{D}_{55} & \overline{D}_{56} \\ \overline{D}_{61} & \overline{D}_{62} & \overline{D}_{63} & \overline{D}_{64} & \overline{D}_{65} & \overline{D}_{66} \end{bmatrix} \begin{Bmatrix} u_{mn}(0) \\ v_{mn}(0) \\ \sigma_{33mn}(0) \\ \sigma_{13mn}(0) \\ \sigma_{23mn}(0) \\ w_{mn}(0) \end{Bmatrix}$$

$$\begin{Bmatrix} \sigma_{33mn}(h) \\ \sigma_{13mn}(h) \\ \sigma_{23mn}(h) \end{Bmatrix} = \begin{bmatrix} \overline{D}_{31} & \overline{D}_{32} & \overline{D}_{36} \\ \overline{D}_{41} & \overline{D}_{42} & \overline{D}_{46} \\ \overline{D}_{51} & \overline{D}_{52} & \overline{D}_{56} \end{bmatrix} \begin{Bmatrix} u_{mn}(0) \\ v_{mn}(0) \\ w_{mn}(0) \end{Bmatrix} + \begin{bmatrix} \overline{D}_{33} & \overline{D}_{34} & \overline{D}_{35} \\ \overline{D}_{43} & \overline{D}_{44} & \overline{D}_{45} \\ \overline{D}_{53} & \overline{D}_{54} & \overline{D}_{55} \end{bmatrix} \begin{Bmatrix} \sigma_{33mn}(0) \\ \sigma_{13mn}(0) \\ \sigma_{23mn}(0) \end{Bmatrix} \qquad 3.15$$

where: $\overline{D} = \prod_{j=p}^{1} D_j(h_j)$

Solving equation 3.15 for the coefficients of displacement at the bottom of the plate yields

$$\begin{Bmatrix} u_{mn}(0) \\ v_{mn}(0) \\ w_{mn}(0) \end{Bmatrix} = [H]^{-1}[Q] \qquad 3.16$$

where: $$H = \begin{bmatrix} \overline{D}_{31} & \overline{D}_{32} & \overline{D}_{36} \\ \overline{D}_{41} & \overline{D}_{42} & \overline{D}_{46} \\ \overline{D}_{51} & \overline{D}_{52} & \overline{D}_{56} \end{bmatrix}$$

and: $$Q = \begin{Bmatrix} \sigma_{33mn}(h) \\ \sigma_{13mn}(h) \\ \sigma_{23mn}(h) \end{Bmatrix} - \begin{bmatrix} \overline{D}_{33} & \overline{D}_{34} & \overline{D}_{35} \\ \overline{D}_{43} & \overline{D}_{44} & \overline{D}_{45} \\ \overline{D}_{53} & \overline{D}_{54} & \overline{D}_{55} \end{bmatrix} \begin{Bmatrix} \sigma_{33mn}(0) \\ \sigma_{13mn}(0) \\ \sigma_{23mn}(0) \end{Bmatrix}$$

The values of the stress coefficients at the top and bottom of the plate can be recovered from the natural boundary conditions. For a uniform surface pressure at the top surface, the coefficient of the Fourier series in equation 3.8 is

$$\sigma_{33_{mn}}(h) = \frac{16q}{mn\pi^2} \qquad 3.17$$

where q is the magnitude of the loading (here, unit pressure). The other components at the top of the plate, and all three components at the bottom of the plate where there is no loading, are equal to zero. Thus, equation 3.13 becomes

$$\begin{Bmatrix} u_{mn}(z) \\ v_{mn}(z) \\ 0 \\ 0 \\ 0 \\ w_{mn}(z) \end{Bmatrix}_p = \prod_{j=p}^{1} D_j \begin{Bmatrix} u_{mn}(0) \\ v_{mn}(0) \\ \dfrac{16q}{mn\pi^2} \\ 0 \\ 0 \\ w_{mn}(0) \end{Bmatrix} \qquad 3.18$$

where the displacement coefficients at the bottom of the plate on the right hand side are solved for using equation 3.16. Thus the coefficients at all ply boundaries, and within any ply, can be solved for using the natural boundary conditions. To do this, the matrix $D_j(z)$ must be calculated. The matrix exponential can be calculated numerically, or analytically:

$$D_j(z) = \alpha_1(z) + \alpha_2(z)G + \alpha_3(z)G^2 + \alpha_4(z)G^3 + \alpha_5(z)G^4 + \alpha_6(z)G^5 \qquad 3.19$$

where: $$G = \begin{bmatrix} 0 & A_{mn} \\ B_{mn} & 0 \end{bmatrix}_j$$

With the coefficients calculated, the displacement and out-of-plane stress components in the plate as a function of position x, y, and z using equations 3.7 and 3.8. Finally, the in-plane stress can be recovered from the relationships found in equation 3.3:

$$\begin{Bmatrix}\sigma_{11}\\\sigma_{22}\\\sigma_{12}\end{Bmatrix}_j = \begin{Bmatrix}C_2\frac{\partial u}{\partial x}+C_3\frac{\partial v}{\partial y}-C_1\sigma_{33}\\C_3\frac{\partial u}{\partial x}+C_4\frac{\partial v}{\partial y}-C_5\sigma_{33}\\C_6\frac{\partial u}{\partial y}+C_6\frac{\partial v}{\partial x}\end{Bmatrix}_j \qquad 3.20$$

which requires the partial derivatives of equation 3.7:

$$\partial\frac{\begin{Bmatrix}u\\v\end{Bmatrix}}{\partial x}_j = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\frac{m\pi}{a}\begin{Bmatrix}-u_{mn}(z)\,sin\left(\frac{m\pi x}{a}\right)sin\left(\frac{n\pi y}{b}\right)\\v_{mn}(z)\,cos\left(\frac{m\pi x}{a}\right)cos\left(\frac{n\pi y}{b}\right)\end{Bmatrix}_j \qquad 3.21$$

$$\partial\frac{\begin{Bmatrix}u\\v\end{Bmatrix}}{\partial y}_j = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\frac{n\pi}{b}\begin{Bmatrix}u_{mn}(z)\,cos\left(\frac{m\pi x}{a}\right)cos\left(\frac{n\pi y}{b}\right)\\-v_{mn}(z)\,sin\left(\frac{m\pi x}{a}\right)sin\left(\frac{n\pi y}{b}\right)\end{Bmatrix}_j$$

3.2.3 Validation of the 3-D Solver

Figure 3.15 shows one 3-D plate mesh with the three different plies colored based on their material properties (plate dimensions are given in Figure 3.14). Several plate meshes were considered with varying resolution across the surface and through the thickness. The example mesh in Figure 3.15 is 12x12x10, indicating it has twelve elements along each side of the surface and 10 elements through its thickness. Using ten elements through the thickness results in one element for the top and bottom plies, and eight for the thicker middle ply. Other plies considered had 20 and 30 elements through the thickness, resulting in two or three elements plies for the top and bottom plies, respectively.

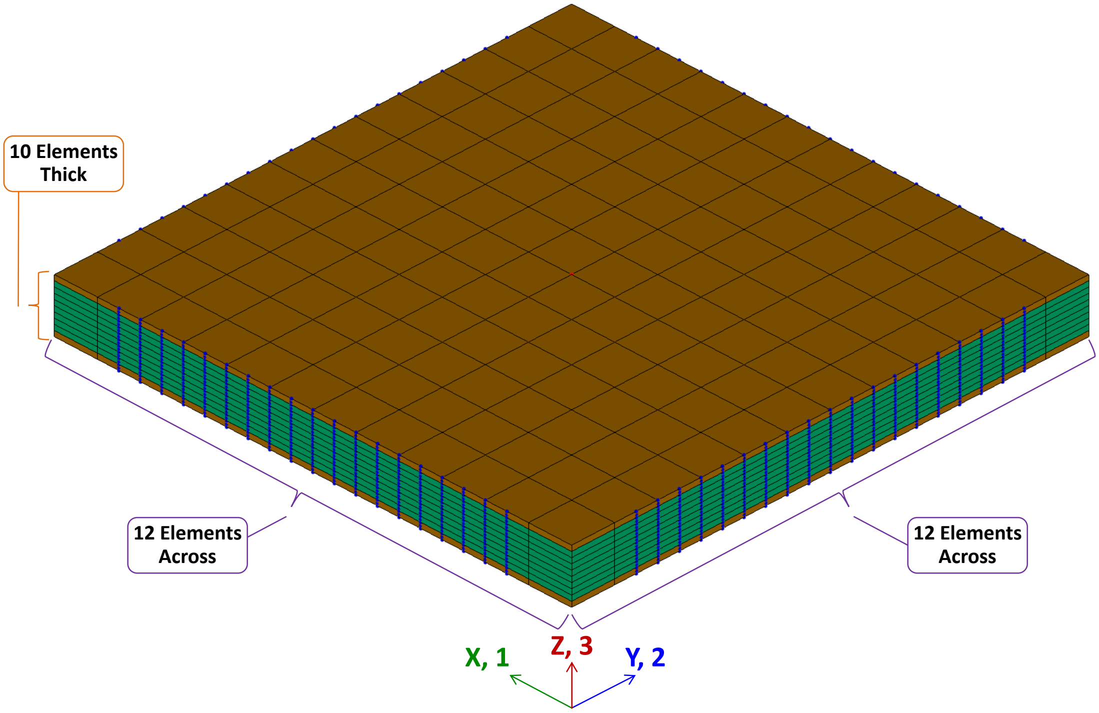


Figure 3.15 The 12x12x10 element mesh of the three ply composite plate used for X3D stress validation. Several meshes are considered, all of which have an equal number (4, 8, 12, or 16) of elements along the edges of the square, and 10, 20, or 30 elements through the thickness of the plate

Figure 3.16 shows the same mesh with plotting nodes and boundary nodes highlighted. The plotting nodes are those at which the stress is plotted for comparison to analysis. The boundary nodes are those at which the boundary conditions provided in Equation 3.1 are applied. The boundary nodes are applied through joint connections, as described in Section 2.4.1.2. A single element cannot be connected to multiple joints in X3D, so the nodes along the edges which belong to corner elements are left out. For fine meshes, such as the 12x12x10 shown in the figure, this is not a problem. When the mesh becomes very coarse, however, such as the 4x4x10 mesh in Figure 3.17, the boundaries are not well represented.

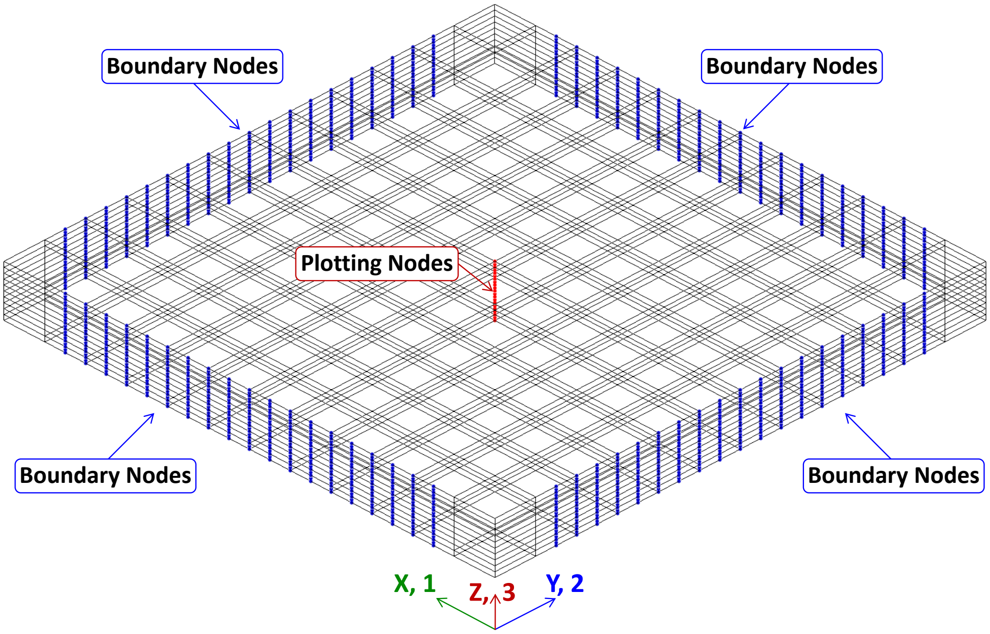


Figure 3.16 The 12x12x10 element mesh of the three ply composite plate showing boundary nodes and center nodes where the stress is plotted for validation

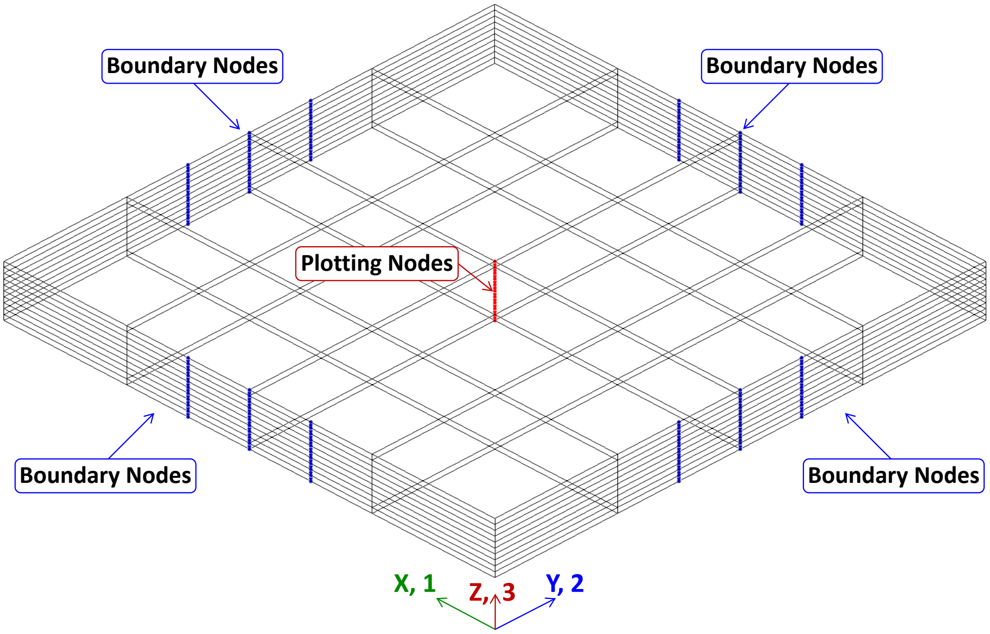


Figure 3.17 The 4x4x10 element mesh of the three ply composite plate used for X3D stress validation; note that because the boundaries do not extend to the corner elements, the boundary is not well represented by the very coarse mesh

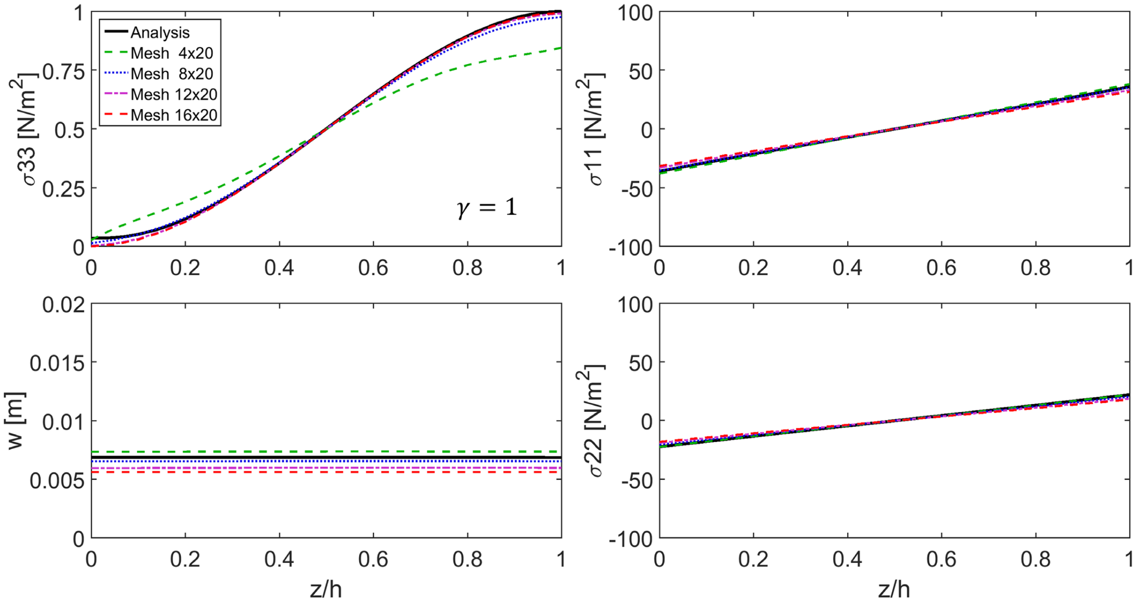


Figure 3.18 Effect of surface mesh size on stress and deformation through the thickness at the center of a simply supported three ply composite plate undergoing a uniform pressure; $\gamma = 1$ indicates the middle ply is identical to the upper and lower plies

Figure 3.19 shows results for $\gamma = 5$. Here the plate is no longer isotropic, creating a step in the in-plane stress components at the interfaces between the plies. As with the uniform case, the 3-D predictions converge to those from the analytical solution for the three stress components, but deflection is under-predicted. The ability to capture the step in stress is limited by the mesh resolution through the thickness of the plate, as was seen in the composite beam in Section 3.1.2. Figure 3.20 shows the same case using 12x12 meshes with 10, 20, and 30 elements through the thickness. The larger meshes offer no greater accuracy, but improve resolution of the step. Note, however, that these nodal stresses are being interpolated linearly in the figure, whereas the higher order Hex27 elements have quadratic shape functions that could be used for more accurate representation, thus allowing improved definition of the stress interface.

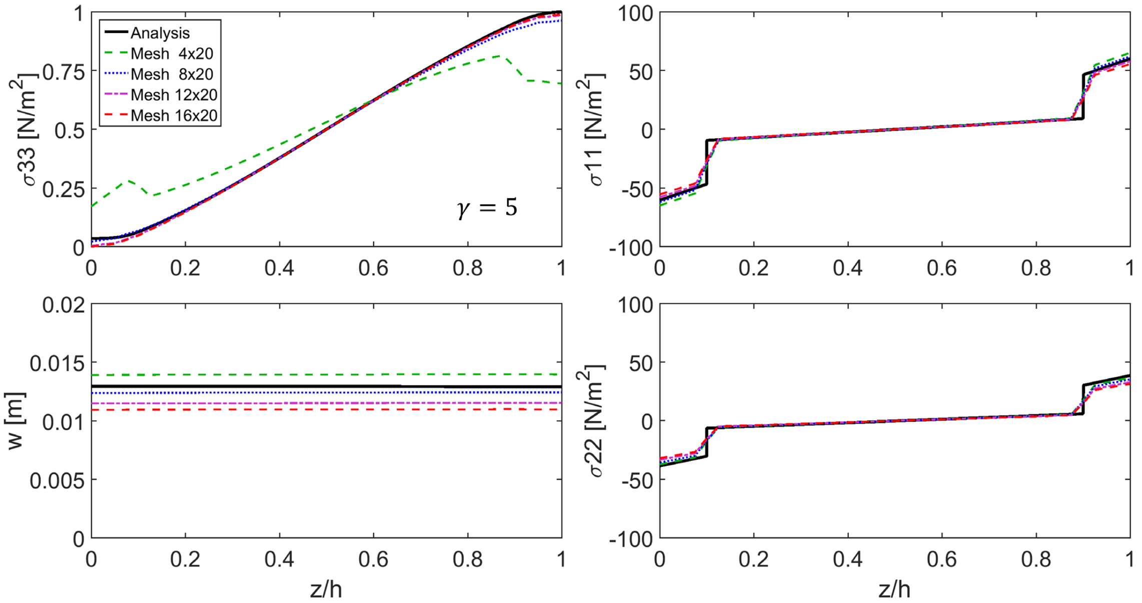


Figure 3.19 Effect of surface mesh size on stress and deformation through the thickness at the center of a simply supported three ply composite plate undergoing a uniform pressure; $\boldsymbol{\gamma = 5}$ indicates the middle ply $\boldsymbol{C}$ matrix stiffness terms are divided by a factor of 5

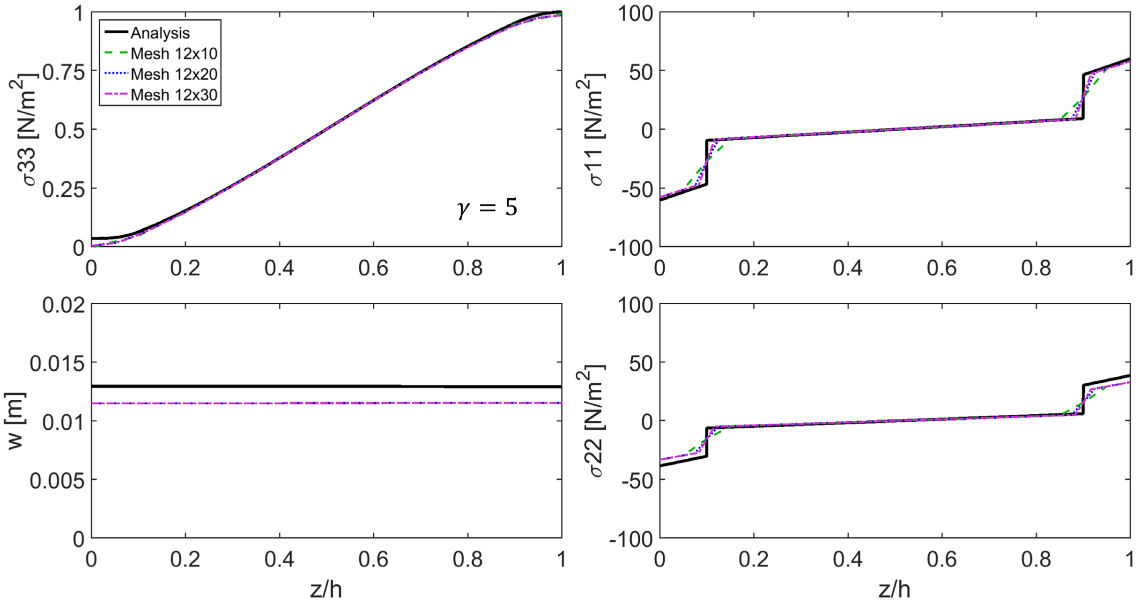


Figure 3.20 Effect of mesh resolution through the thickness on stress and deformation at the center of a simply supported three ply composite plate undergoing a uniform pressure; $\boldsymbol{\gamma = 5}$ indicates the middle ply $\boldsymbol{C}$ matrix stiffness terms are divided by a factor of 5

Similar results for plates with softer middle plies are shown in Figure 3.21 ($\gamma = 10$) and Figure 3.22 ($\gamma = 15$). Once again, stress is well predicted using the finer surface meshes, but deflection is under-predicted.

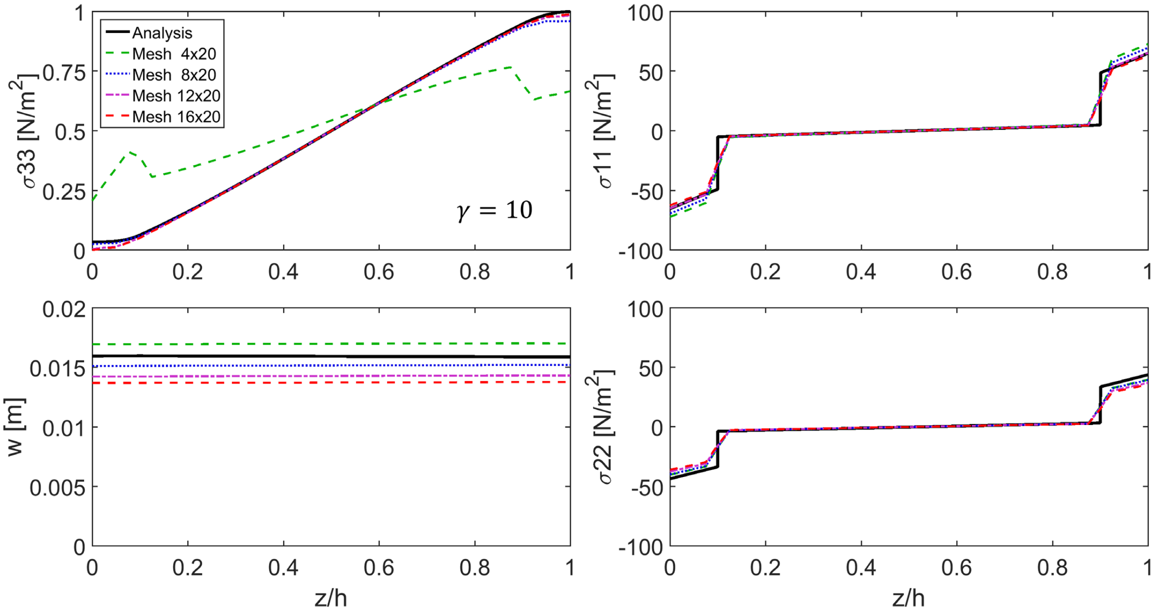


Figure 3.21 Effect of surface mesh size on stress and deformation through the thickness at the center of a simply supported three ply composite plate undergoing a uniform pressure; $\gamma = 10$ indicates the middle ply C matrix stiffness terms are divided by a factor of 10

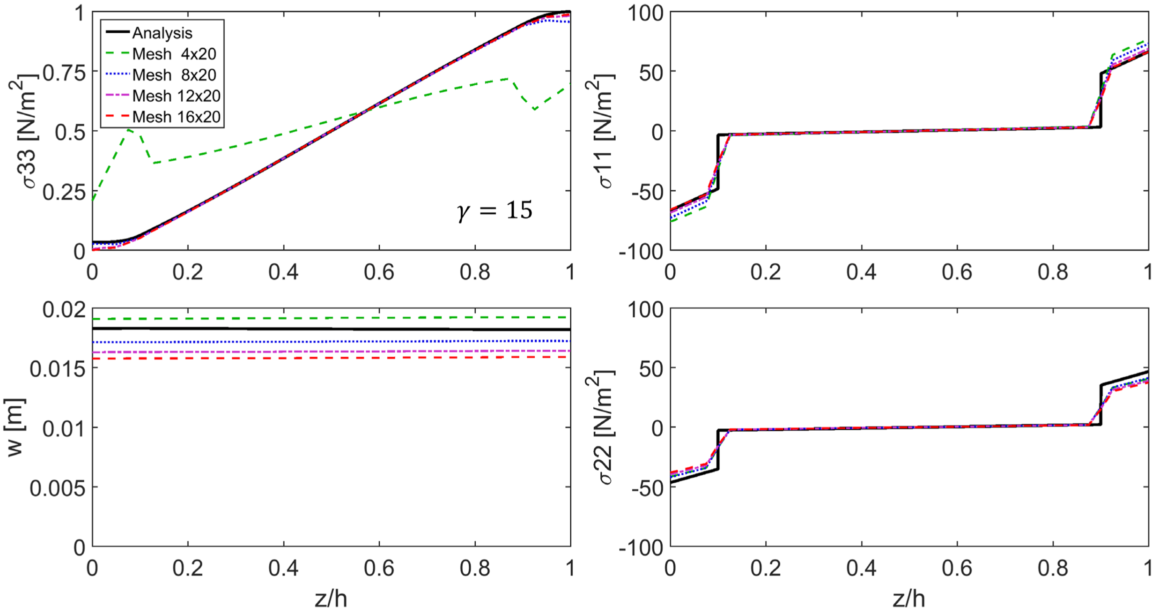


Figure 3.22 Effect of surface mesh size on stress and deformation through the thickness at the center of a simply supported three ply composite plate undergoing a uniform pressure; $\gamma = 15$ indicates the middle ply C matrix stiffness terms are divided by a factor of 15

3.3 Rotor Model Integration with 3-D Solver

A full-up rotor model consists of many individual parts that must be connected through joints and assembled into a rotor structural analysis model (SAM). This section verifies that SAMs built using the methodology laid out in Chapter 2 can indeed be interfaced with the 3-D solver. The test model is a notional bearingless rotor consisting of four flex parts and six joints, as described in section 2.4.2.2.

3.3.1 Static Analysis

Figure 3.23 shows the full structural analysis model (SAM) of the bearingless rotor after being loaded into the solver. To verify the integration of the model, the response to a chordwise tip load of 500 N (112.4 lb) was examined (Figure 3.24). Note that for this case the snubber joint between the flexure and pitchcase (containing the damper) was deliberately set to zero stiffness and a large deformation was induced in order to illustrate the exactness of the blade-flexure-torque tube connection. As seen in the figure, the blade leads forward via elastic deflection of the flexure as well as its own small amount of in-plane chordwise deflection. The torque tube remains straight, not taking any load, and penetrating through the flexure because of the disabled lag damper (there is no contact model in the solver). This verifies that the blade bolts (rigid joints) are being correctly implemented and that the finite element analysis is yielding qualitatively appropriate deflections for flex parts in a multibody formulation.

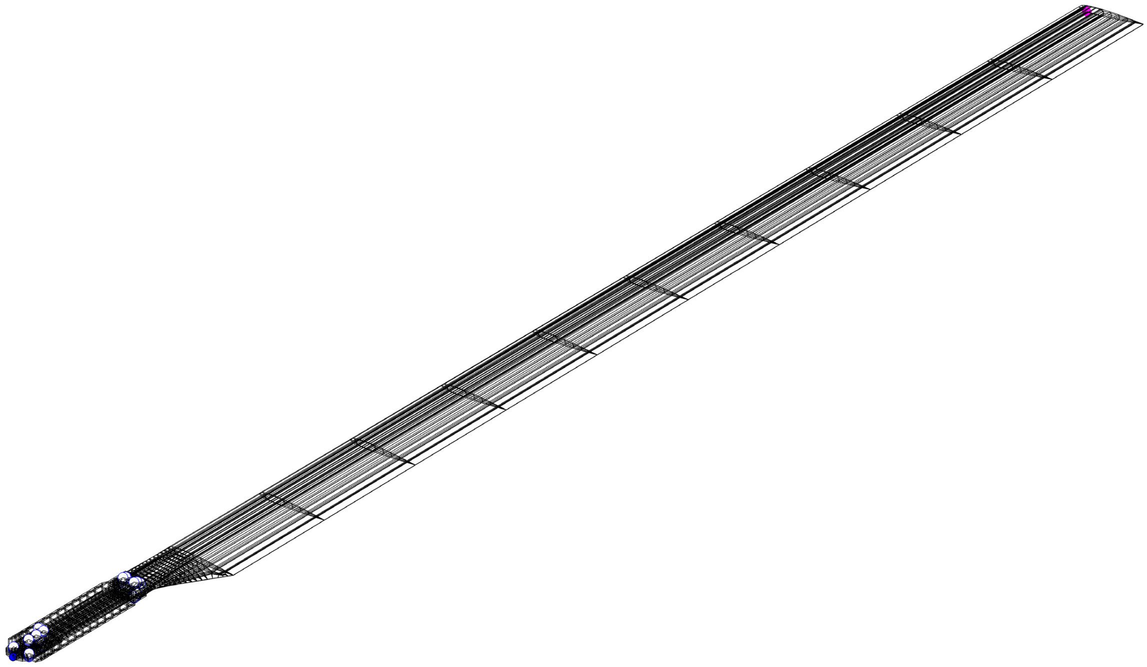


Figure 3.23 Notional bearingless rotor SAM, visualization output by X3D

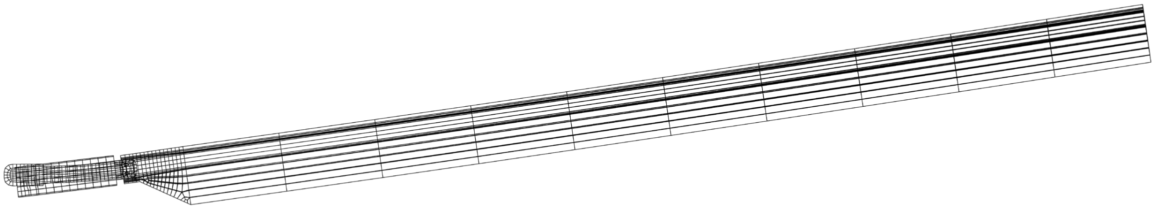


Figure 3.24 Bearingless rotor SAM, deflecting under a 500N chordwise tip loading

Next, flapping response to a 10 N (2.25 lb) vertical tip load was tested, with a stiff damper joint, shown in Figure 3.25. Looking closely at the root of the blade (Figure 3.26) reveals that most of the deflection occurs at the flexure, which bends within the stiffer torque tube, as anticipated. Again, this suggests that the structural model is responding properly with multiple joints (root bolts, blade bolts, and stiff lag damper) as intended.

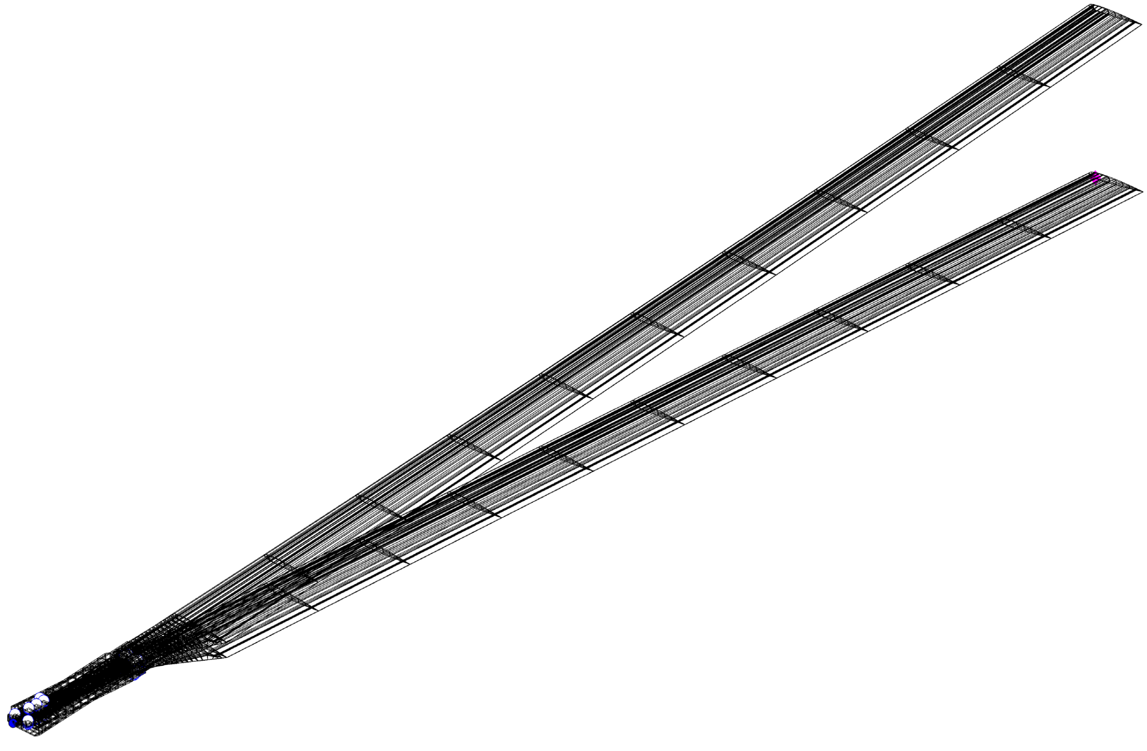


Figure 3.25 Bearingless rotor SAM, deflecting under a 10N vertical tip loading

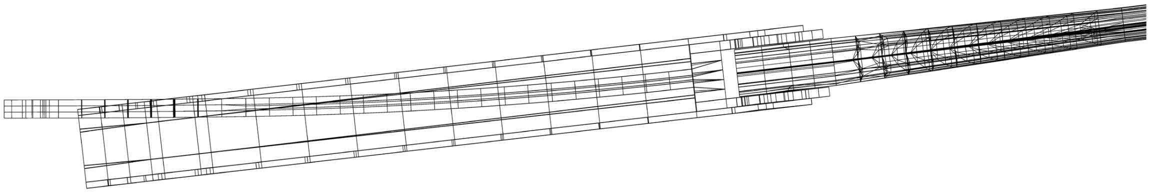


Figure 3.26 Root section of bearingless rotor, deflecting under a 10N vertical tip loading

3.3.2 Dynamic Analysis

After a qualitative examination of static deflections of the bearingless rotor, the integrity of the model is evaluated for a dynamic case by examining the action of the lag dumper. The snubber damper is a complex device, but for rotor dynamics only its primary mechanisms of interaction with the blade are needed. These are conveniently modeled using a five degree of freedom joint at the damper location (J3 in Figure 2.8), with vertical translation restricted, locking the separation between the flexure and the torque tube.

The pitch/flap/lag bearing degrees of freedom at the damper joint require some stiffness; a stiffness of 10,000 N/m (685.2 lb/ft) is assigned to both axial and chordwise translation, 100 Nm/rad (73.8 ft·lb/rad) assigned to flap and lead-lag rotations, and 10 Nm/ rad (7.38 ft·lb/rad) to pitch rotation. The primary action of this joint is to provide viscous damping to the relative lead-lag rotation, so a linear damping is assigned to the lead-lag rotational degree of freedom. The action of the damper is verified with a simple model in Figure 3.27. An in-plane translation is impulsively applied at the blade tip by means of a tip joint with a prescribed motion, and the rotation position of the damper lead-lag degree of freedom is studied for four different values of damping (as fractions of damping coefficient found in the UH-60A rotor). Typical damping values for a bearingless rotor of the type studied here are not available in the public domain, and the values used are more typical of an articulated rotor. The intent is not to validate the model, but to verify the assembled model exhibits the qualitative features of dynamic motion properly.

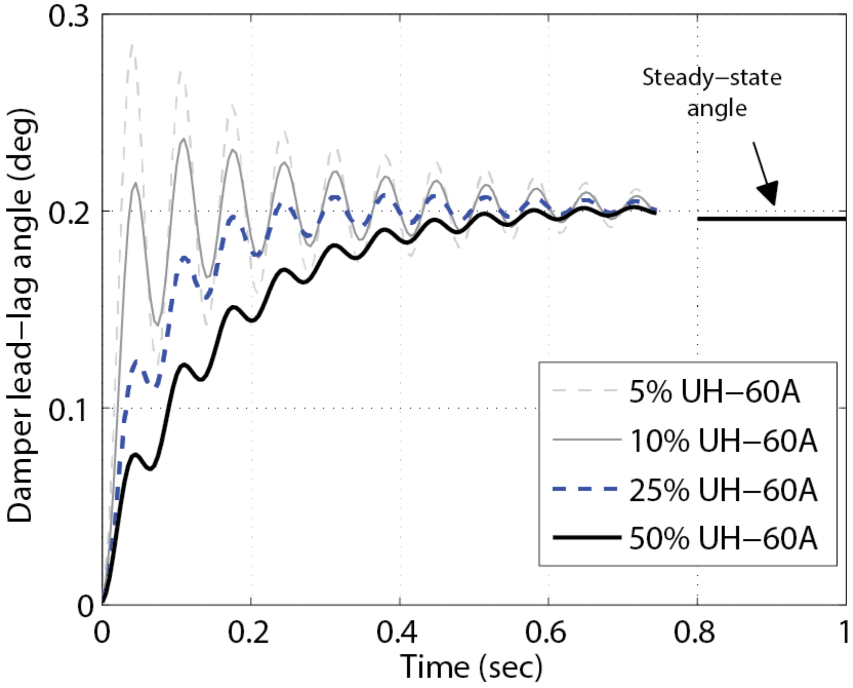


Figure 3.27 Bearingless rotor response to a leading tip force perturbation under various damping ratios

With 5% of the measured equivalent lag damping found in the UH-60A, the lag rotation shows a significant overshoot before it damps down to the expected steady state angle. At high damping (50% UH-60A), there is no overshoot, and oscillations damp out more quickly. This suggests the joint is applying damping to the lag mode as intended.

Chapter 4: Integrated 3-D Model of the TRAM Proprotor

This chapter introduces the NASA Tilt Rotor Aeroacoustic Model (TRAM) proprotor and presents the model developed for integrated 3-D analysis. Whereas earlier models discussed were notional concepts, the TRAM model is accurate to the actual experimental test article. The structural analysis model (SAM) is described, following the modeling workflow laid out in Chapter 2, and details of the aerodynamic models used for aeromechanics analysis are provided. This is the first 3-D finite element analysis/multibody model of an actual rotor.

4.1 The Tilt Rotor Aeroacoustic Model (TRAM)

The Tilt Rotor Aeroacoustic Model is a Mach-scale test article developed by NASA [118, 138, 139]. It is a 1/4-scale model of the V-22 tiltrotor aircraft dynamically scaled to match the V-22 frequencies in the first flap, lag, and torsion at similar tip Mach numbers to the V-22 ($M_{Tip} = 0.59$ in cruise, $M_{Tip} = 0.71$ in hover). The TRAM program consisted of both an isolated rotor (Figure 4.1, from [138]), tested at the DNW acoustic wind tunnel in the Netherlands [140], and a Full-Span model (FS TRAM), tested at the 40- by 80-foot National Full-scale Aerodynamics Complex (NFAC) wind tunnel at NASA Ames Research Center (Figure 4.2, from [141]). Only the isolated rotor model is considered in this work.

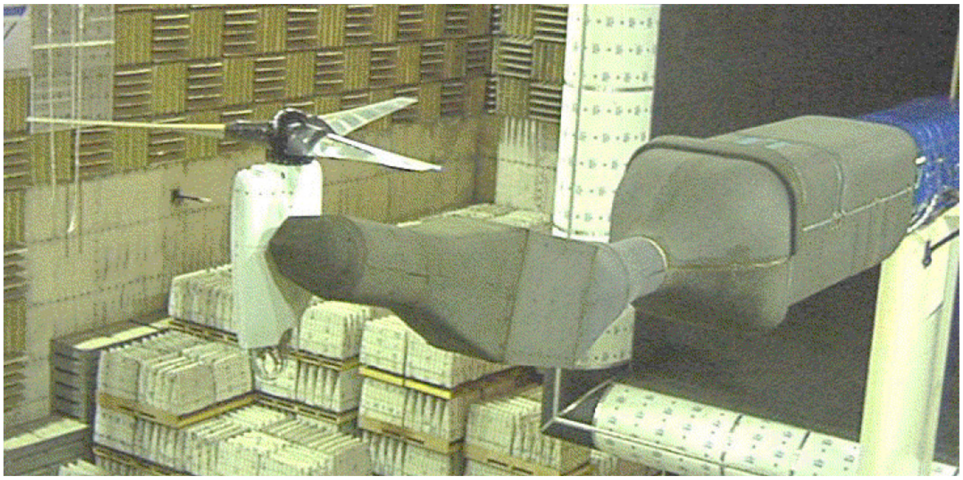


Figure 4.1 Isolated TRAM rotor in the DNW Tunnel

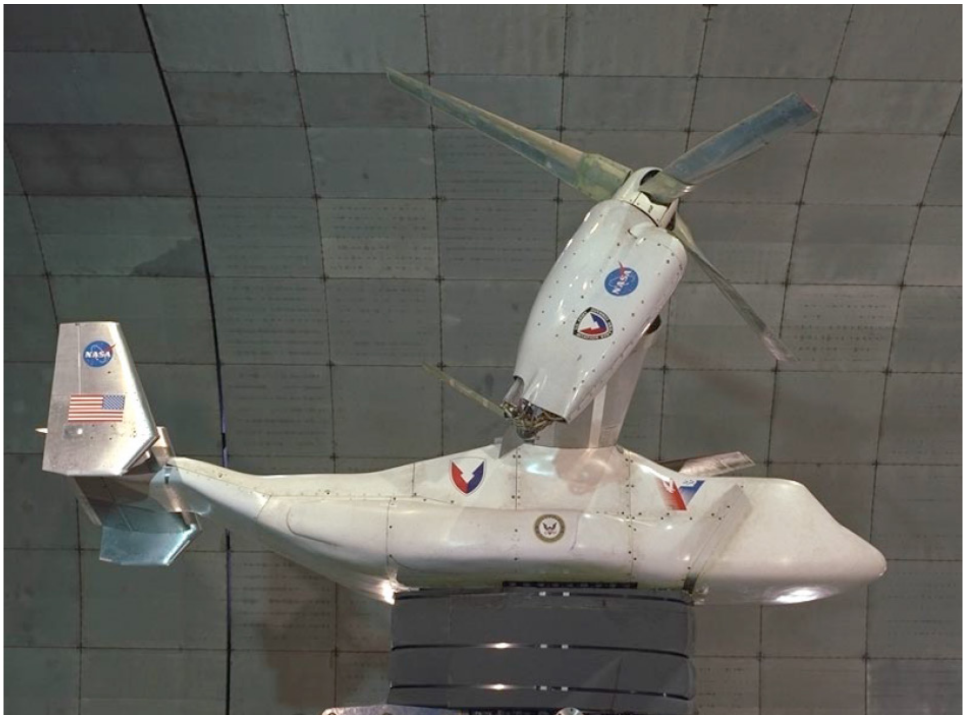


Figure 4.2 Full-Span TRAM in the NFAC 40- by 80-Foot Wind Tunnel

The TRAM proprotor was chosen for this research because most of the manufacturing drawings were made available by NASA Ames Research Center. TRAM is a good candidate for demonstrating and evaluating integrated 3-D analysis of helicopter rotors because it is a modern, composite design with a complex internal blade and hub structure with multiple load paths, involving multiple flexible components and bearings near the root. Furthermore, being a NASA project, many of the details of rotor design and construction are available, including reduced order beam properties for verification. It has also been somewhat analyzed previously, with test data available in open literature. Most of the analyses so far, however, have focused on aerodynamics. Only one study used comprehensive analysis (CAMRAD II) to examine structural loads using a beam-based model with an equivalent single load path root element [95, 109, 110, 32], although this was followed up by a similar study in 2017 using RCAS [111]. CFD analysis performed so far has been limited to hover with no structural coupling [91, 94]. These factors combine to create an opportunity to demonstrate the new modeling methodology proposed here, and verify the capabilities of the new 3-D analysis, while advancing the state-of-the-art in tiltrotor loads prediction far beyond the current generation.

The TRAM proprotor, as seen in Figure 4.3 (from [138]), Figure 4.4 (from [141]), and Figure 4.5, (from [118]) has three blades, each held in place by a grip that also serves as a pitchcase for transferring torque from the pitch link. The pitchcase is connected to a flexbeam, which it surrounds, by two sets of pitch bearings: one outboard of the flexbeam and a second inboard. The flexbeam serves as the primary flapping hinge of the stiff in-plane rotor. At its end, the flexbeam is rigidly connected to the gimbaled hub. Because of the two pitch bearings passing forces between the pitchcase and flexbeam, the blade has a dual load path at its root: all of the blade loads are transferred via the blade grips to the end of the pitchcase; the outer bearing between the pitchcase and flexbeam allows only the forces – centrifugal force, and most (but not all) of the vertical and side shear forces – but no moments to be transferred to the flexbeam; the inner pitch bearing passes the pitchcase shears back into the flexbeam, but no moments; finally, the forces in the flexbeam are transferred to the gimbaled hub by the bolt attachments at its root. The blade torsion loads are carried via the pitchcase to the pitch horn and pitch link.

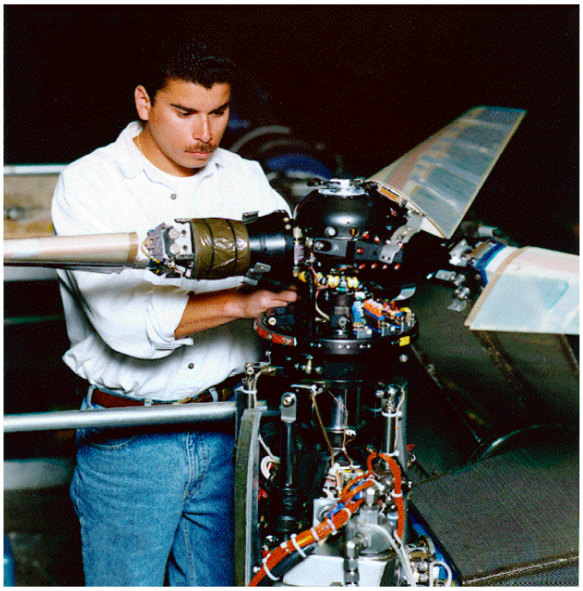


Figure 4.3 Photograph of the TRAM hub

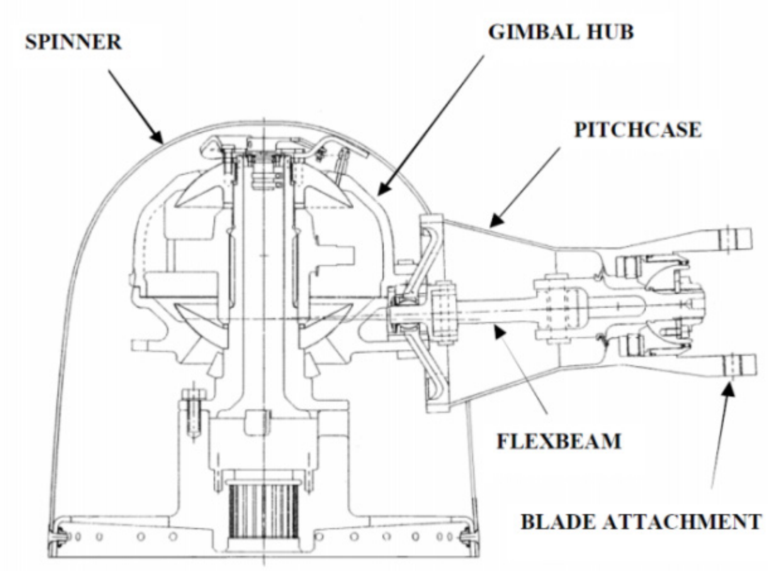


Figure 4.4 TRAM hub, cross-section side view

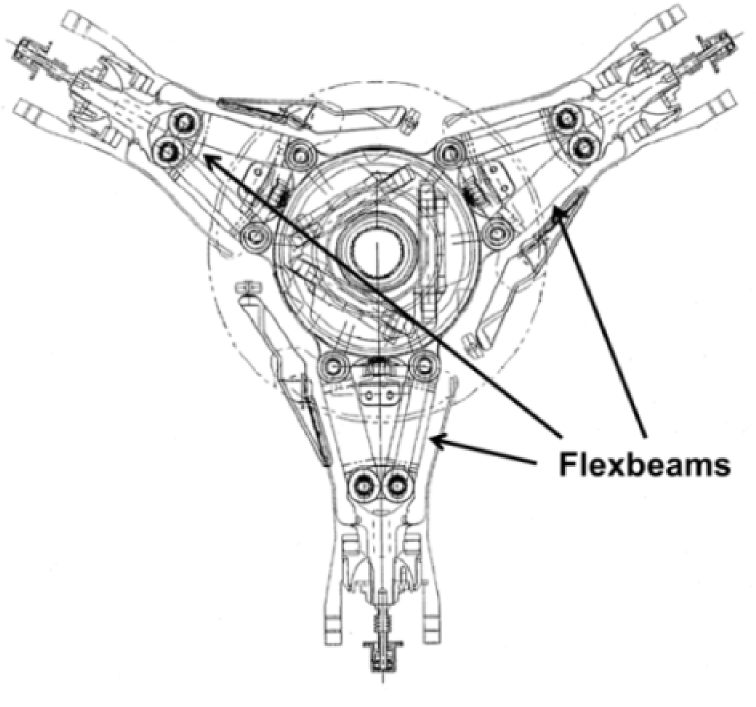


Figure 4.5 TRAM hub, cross-section top view

4.2 The TRAM Structural Analysis Model

A structural analysis model for the TRAM rotor was developed for integrated 3-D analysis. The TRAM rotor presents the first real test of CAD-based structural modeling, and the first attempt to validate a CAD-based rotor model against experimental data. This section follows the methodology and workflow laid out in Chapter 2.

4.2.1 TRAM CAD

The first step in modeling the TRAM proprotor was the creation of a CAD model. The CAD models were created using CATIA V5R20 and exported for meshing as STP files. Although no original CAD of the TRAM was available, detailed manufacturer part drawings of the rotor hub and blades were provided by NASA for this research. Using these drawings, it was possible to model the rotor hub in detail, matching the TRAM rotor very closely (Figure 4.6 and Figure 4.7).

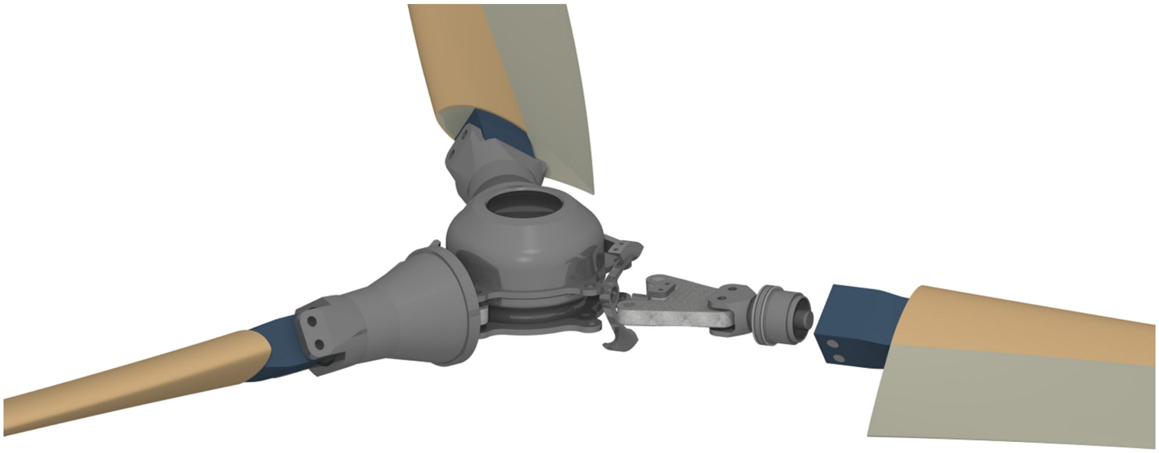


Figure 4.6 CAD model of the TRAM proprotor hub

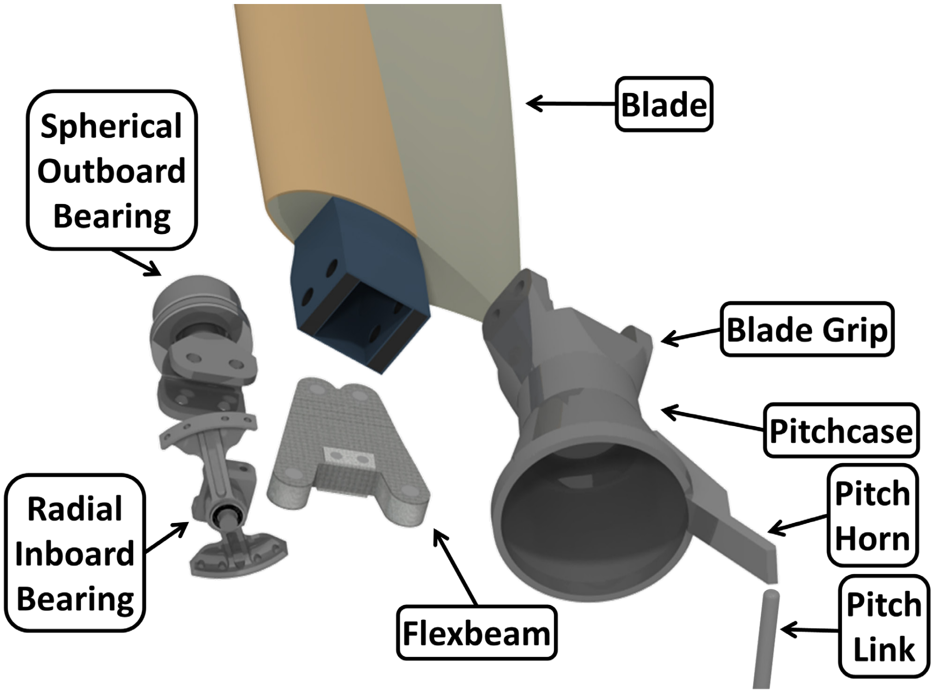


Figure 4.7 CAD model of the TRAM proprotor blade root

Modeling the blade proved more difficult due to its complicated internal structure. Blade cross-sections at several stations were obtained from 2-D manufacturer's drawings, ensuring that the model matches the TRAM rotor at these sections precisely. Between the provided sections, however, interpolation of the blade geometry had to be performed. In addition to complications caused by the blade structure, some simplifications were made to the blade to ease analysis. For example, the actual TRAM blade has an instrumentation package that covers the entire forward portion of the blade, and extends aft toward the trailing edge of the blade at several radial locations to allow chordwise pressure measurements (Figure 4.8). The model developed for this work, however, only considers the forward portion of the instrumentation package, which is the more substantial piece, neglecting the small portions that reach toward the trailing edge (Figure 4.9). Although these approximations could lead to flaws in the geometry, a comparison of mass properties (Section 5.1.1) shows that the model geometry was of acceptable accuracy, with less than 3% error in total blade mass.

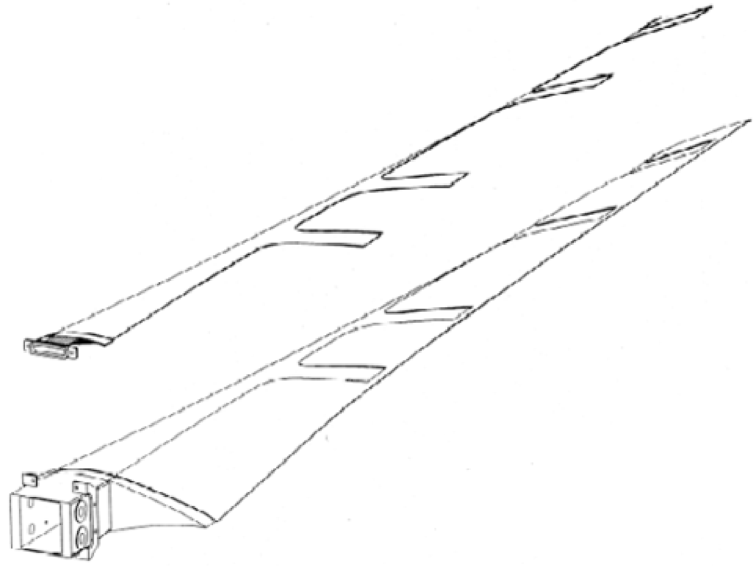


Figure 4.8 Schematic of the TRAM blade with instrumentation package removed

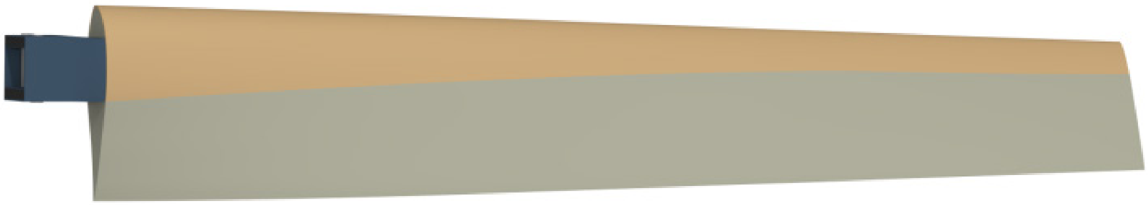


Figure 4.9 CAD model of the TRAM blade, with instrumentation package only over the forward portion

4.2.2 TRAM Structural Analysis Representation (SAR)

The structural analysis representation is a description of the topology of the structural model; at this stage the modeling method (flexible part treated with FEA or kinematic joint) for each rotor component is assigned. Two models of the TRAM rotor were developed, one with a single blade and one with all three blades attached to a central gimbal joint.

The single-bladed TRAM model is represented in the SAR by six flexible parts and eight joints with no devices, as represented by the schematic in Figure 4.10. The parts are numbered from outboard to inboard, followed by the controls, as detailed in Table 4.1. The blade, F1, and flexbeam, F4, are each treated as individual flex parts. Due to meshing concerns, the pitchcase is broken into three separate flex parts: the grips, F2; the pitchcase body, F3; and the pitch horn, F5. The pitch link, F6, is the final flex part. The first joint, J1, represents the bolts connecting the blade spar to the blade grips. Joint J2 is the attachment of the blade grips to the pitchcase body. The outboard and inboard centering bearing are modeled by joints J3 and J4 respectively, both connecting the flexbeam to the pitchcase. Joint J5 represents the gimbal, and is attached to the root of the flexbeam. The gimbal joint is a boundary and is thus linked to the reference frame, which is indicated by an entry of `-1` in its connections in Table 4.1. Joint J6 attaches the pitch horn to the pitchcase, while joint J7 connects it to the top end of the pitch link. The lower end of the pitch link is attached to joint J8, a slider with a prescribed vertical degree of freedom used to represent the motion of a swashplate. As with the gimbal, the bottom of the pitch link forms a boundary, and thus has an entry of `-1` in its list of connections.

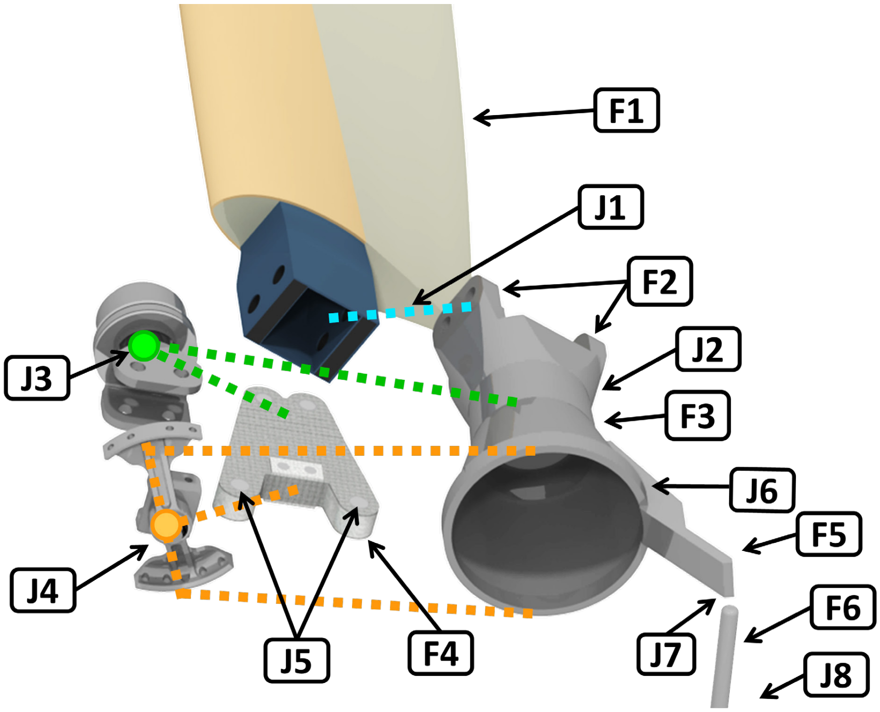


Figure 4.10 Schematic of the single-bladed TRAM assembly structural analysis representation

Table 4.1 Structural analysis representation of the single-bladed TRAM model

Part ID	Type ID	Name	Connections
1	F1	Blade	2
2	J1	Blade – Grip	1, 3
3	F2	Grip	2, 4
4	J2	Grip – Pitch Case	3, 5
5	F3	Pitch Case	4, 6, 8, 10
6	J3	Pitch Bearing Outer	5, 7
7	F4	Flexbeam	6, 8, 9
8	J4	Pitch Bearing Inner	5, 7
9	J5	Gimbal	-1, 7
10	J6	Pitch Case – Pitch Horn	5, 11
11	F5	Pitch Horn	10, 12
12	J7	Pitch Horn – Pitch Link	11, 13
13	F6	Pitch Link	12, 14
14	J8	Pitch Link – Swash Plate	-1, 13

The three-bladed model is similar to the single bladed model, but repeated three times around a single gimbaled joint. This leads to a model with 18 flex parts and 22 joints. The central gimbal joint (formerly J5 for the single-bladed model) is modified to be the last part in the model and is connected to all three flexbeams. Table 4.2 identifies all of the parts in the SAR and their attachments for the three-bladed gimbaled rotor model. For example, part 7 is the first flexbeam (flex part F4), and it connects to parts 6 (the first outboard pitch bearing), 8 (the first inboard pitch bearing), and 40 (the gimbal hub joint).

Table 4.2 SAR for the 3-bladed gimbaled rotor model, "outside-in" part ordering, with horizontal dividers indicating change in blade

Part ID	Type ID	Name	Connections
1	F1	Blade 1	2
2	J1	Blade-Grip 1	1, 3
3	F2	Grip 1	2, 4
4	J2	Grip-Pitch Case 1	3, 5
5	F3	Pitch Case 1	4, 6, 8, 9
6	J3	Pitch Bearing Outer 1	5, 7
7	F4	Flexbeam 1	6, 8, 40
8	J4	Pitch Bearing Inner 1	5, 7
9	J5	Pitch Case – Pitch Horn 1	5, 10
10	F5	Pitch Horn 1	9, 11
11	J6	Pitch Horn – Pitch Link 1	10, 12
12	F6	Pitch Link 1	11, 13
13	J7	Pitch Link – Swashplate 1	12
14	F7	Blade 2	15
15	J8	Blade-Grip 2	14, 16
16	F8	Grip 2	15, 17
17	J9	Grip-Pitch Case 2	16, 18
18	F9	Pitch Case 2	17, 19, 21, 22
19	J10	Pitch Bearing Outer 2	18, 20
20	F10	Flexbeam 2	19, 21, 40
21	J11	Pitch Bearing Inner 2	18, 20
22	J12	Pitch Case – Pitch Horn 2	18, 23

23	F11	Pitch Horn 2	22, 24
24	J13	Pitch Horn – Pitch Link 2	23, 25
25	F12	Pitch Link 2	24, 26
26	J14	Pitch Link – Swashplate 2	25
27	F13	Blade 3	28
28	J15	Blade-Grip 3	27, 29
29	F14	Grip 3	28, 30
30	J16	Grip-Pitch Case 3	29, 31
31	F15	Pitch Case 3	30, 32, 34, 35
32	J17	Pitch Bearing Outer 3	31, 33
33	F16	Flexbeam 3	32, 34, 40
34	J18	Pitch Bearing Inner 3	31, 33
35	J19	Pitch Case – Pitch Horn 3	31, 36
36	F17	Pitch Horn 3	35, 37
37	J20	Pitch Horn – Pitch Link 3	36, 38
38	F18	Pitch Link 3	37, 39
39	J21	Pitch Link – Swashplate 3	38
40	J22	Gimbal	-1, 7, 20, 33

Part ordering is important when performing 3-D analysis because of the skyline storage scheme used when the solver is run on a single processor, as in this work (see Section 2.6.1.3). The skyline solver is most efficient when the bandwidth of each element in the solution is as low as possible. The bandwidth for an element is defined by the gap in nodes included in an element, or the difference between the highest and lowest node identification (ID) number. Node IDs are assigned part-by-part, so a joint between two parts separated by a large gap will result in very high bandwidth. To reduce bandwidth, and thereby improve computational efficiency, parts which connect to one another should be ordered close together.

The single bladed model, as shown in Figure 4.10, is ordered "outside-in" (starting with the blade as F1) and has an average bandwidth of 1,673. This is illustrated in Figure 4.11, which shows the non-zero stiffness matrix entries above the diagonal (which appear as a "skyline") for the one-bladed structural analysis model: the relatively low bandwidth is indicated by having most entries fall close to the diagonal. The skyline shows the total memory required by the solver, and the bandwidth indicates the solution time.

Despite the good average bandwidth, there is a large spike around the 2900th degree of freedom: this is joint J1, the connection between the grips and the blade. The blade is the largest mesh and represents all degrees of freedom before that spike. The mesh of the blade has its nodes ordered starting from the root and ending at the tip; if the node IDs had been renumbered from tip to root it would better match the outside-in ordering scheme and further reduce bandwidth. At the end of the skyline is a "plateau" like region; this represents where the relatively fine flexbeam mesh attaches to the root joint.

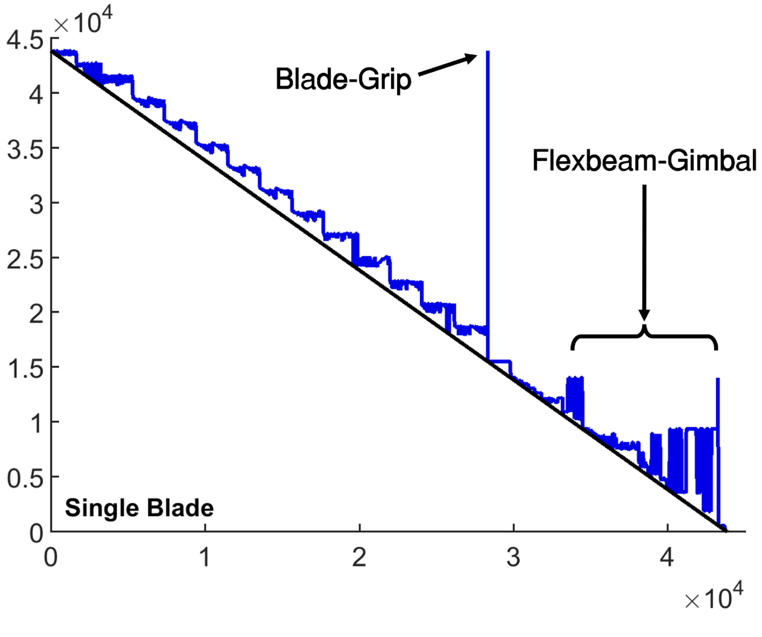


Figure 4.11 Depiction of the skyline for the single-bladed rotor, outside-in part ordering

For the three bladed rotor, the part ordering scheme shown in Table 4.2 was used in the present analysis. Like the one-bladed structural analysis model, it is ordered outside-in, starting from the blades (blade 1 is part 1, blade 2 is part 14, blade 3 is part 27) then continuing inward toward the center, with the final part (40) being the gimbal joint. Because it is based on the one-bladed model, it has a similar average bandwidth of 1,677. Its skyline (Figure 4.12) looks similar to that for the one-bladed model, but repeated three times, with one final large spike at the end where the gimbal connects to the three flexbeams. Note that the height of this last peak reaches up the height of the skyline at the flexbeam "plateau" at the end of the first blade; the connection between the gimbal and the flexbeam node with the highest ID drives the maximum bandwidth peak in this skyline.

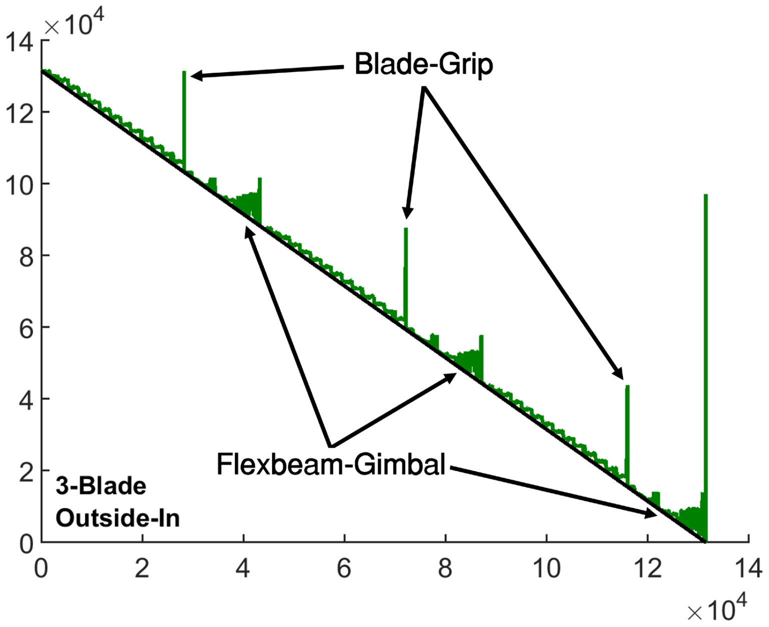


Figure 4.12 Depiction of the skyline for the three-bladed rotor, outside-in part ordering

A second model was also considered, with “inside-out” ordering, starting with the gimbal as part 1 and then working its way out to the three blades (parts 14, 27, and 40), as listed in Table 4.3. Note that this is not just a simple reversal of the earlier models, but also has its inboard components ordered differently. This leads to a skyline with many large spikes, as seen in Figure 4.13. Whereas the first three-bladed model is characterized by tall peaks (blade roots) and shorter plateaus (flexbeam roots), this “outside-in” ordering has three plateaus where the flexbeam connects to the gimbal. The first plateau, at the very beginning of the skyline, is very small because the first flexbeam is close to the gimbal, which is part 1 and therefore embodies the first degrees of freedom. The second and third flexbeam plateaus, however, are very tall because they must connect all the way back to the gimbal. This time, though, there are no blade root peaks because the part ordering matches the mesh node ordering.

Even though the meshes are well suited for this “inside-out” scheme, the fact that many nodes must connect to the gimbal leads to a much worse bandwidth, with an average value of 2,948, 76% higher than the “outside-in” model. In practical terms, this leads to a 71% increase in memory usage for the second “inside-out” rotor SAR versus the “outside-in,” demonstrating the importance of reducing bandwidth.

Table 4.3 SAR for the 3-bladed gimbaled rotor model, “inside-out” part ordering, with horizontal dividers indicating change in blade

Part ID	Type ID	Name	Connections
1	J1	Gimbal	-1, 2, 15, 28
2	F1	Flexbeam	1, 3, 4
3	J2	Pitch Bearing Inner	2, 5
4	J3	Pitch Bearing Outer	2, 5
5	F2	Pitch Case	3, 4, 6, 11

6	J4	Pitch Case–Pitch Horn	5, 7
7	F3	Pitch Horn	6, 8
8	J5	Pitch Horn-Link	7, 9
9	F4	Pitch Link	8, 10
10	J6	Pitch Link-Swash	9
11	J7	Grip-Pitch Case	5, 12
12	F5	Grip	11, 13
13	J8	Blade-Grip	12, 14
14	F6	Blade	13
15	F7	Flexbeam	1, 16, 17
16	J9	Pitch Bearing Inner	15, 18
17	J10	Pitch Bearing Outer	15, 18
18	F8	Pitch Case	16, 17, 19, 24
19	J11	Pitch Case–Pitch Horn	18, 20
20	F9	Pitch Horn	19, 21
21	J12	Pitch Horn-Link	20, 22
22	F10	Pitch Link	21, 23
23	J13	Pitch Link-Swash	22
24	J14	Grip-Pitch Case	18, 25
25	F11	Grip	24, 26
26	J15	Blade-Grip	25, 27
27	F12	Blade	26
28	F13	Flexbeam	1, 29, 30
29	J16	Pitch Bearing Inner	28, 31
30	J17	Pitch Bearing Outer	28, 31
31	F14	Pitch Case	29, 30, 32, 37
32	J18	Pitch Case–Pitch Horn	31, 33
33	F15	Pitch Horn	32, 34
34	J19	Pitch Horn-Link	33, 35
35	F16	Pitch Link	34, 36
36	J20	Pitch Link-Swash	35
37	J21	Grip-Pitch Case	31, 38
38	F17	Grip	37, 39
39	J22	Blade-Grip	38, 40
40	F18	Blade	39

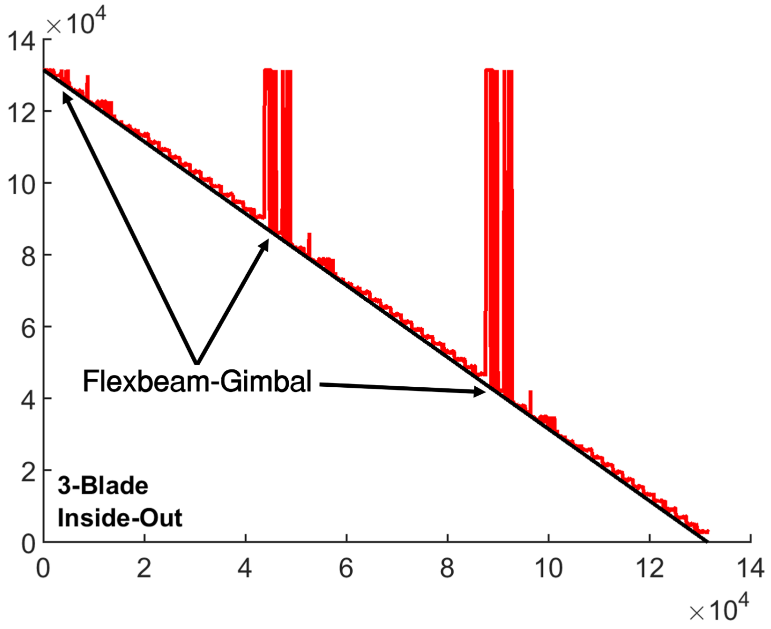


Figure 4.13 Depiction of the skyline for the three-bladed rotor, inside-out part ordering

4.2.3 TRAM Meshing

The six flexible parts of the TRAM assembly were meshed with Hex27 elements using Cubit, as described in Section 2.5. The flexbeam mesh, shown in Figure 4.14, has its bolt holes plugged by elements whose nodes are used to define joints. Nodesets 401 and 402 are used to connect to the outboard and inboard pitch bearings, joints J3 and J4, respectively. Nodeset 403 occupies the root bolts and is connected to the hub bolt boundary joint, J8. The total mesh is composed of 1,304 elements with 12,463 nodes (12,463×3 = 37,389 degrees of freedom).

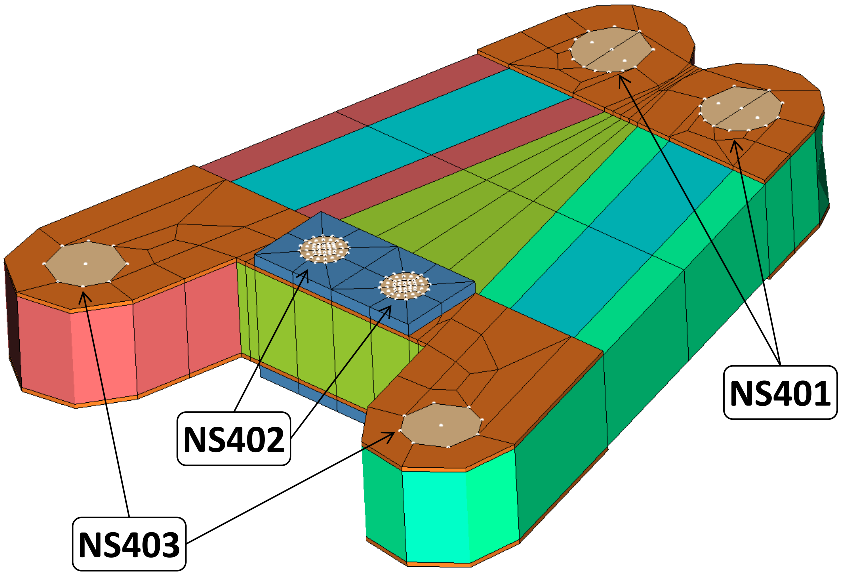


Figure 4.14 TRAM flexbeam mesh, with nodesets for joint connection identified

The blade was discretized by meshing an internal cross-section surface and then sweeping inboard and outboard from the middle. Due to constraints imposed by the internal geometry of the blade, the cross-sectional mesh is relatively fine. Multiple radial mesh sizes were considered (Figure 4.15). The number of elements and nodes in each mesh can be found in 4.4. Finer meshes were not considered due to limitations in computational power available. Each blade mesh has a single nodeset (NS401) comprising the root of the spar which connects to the blade grips via joint J1 (Figure 4.16). Notice that the two finer meshes have a bias towards a very small element which is present even in the coarse mesh (called out by an arrow); this is due to an internal feature of the CAD model which needed to be resolved with a smaller mesh step. The linear bias was not strictly required, but was determined to be the easiest way to resolve the feature and generate a consistent mesh.

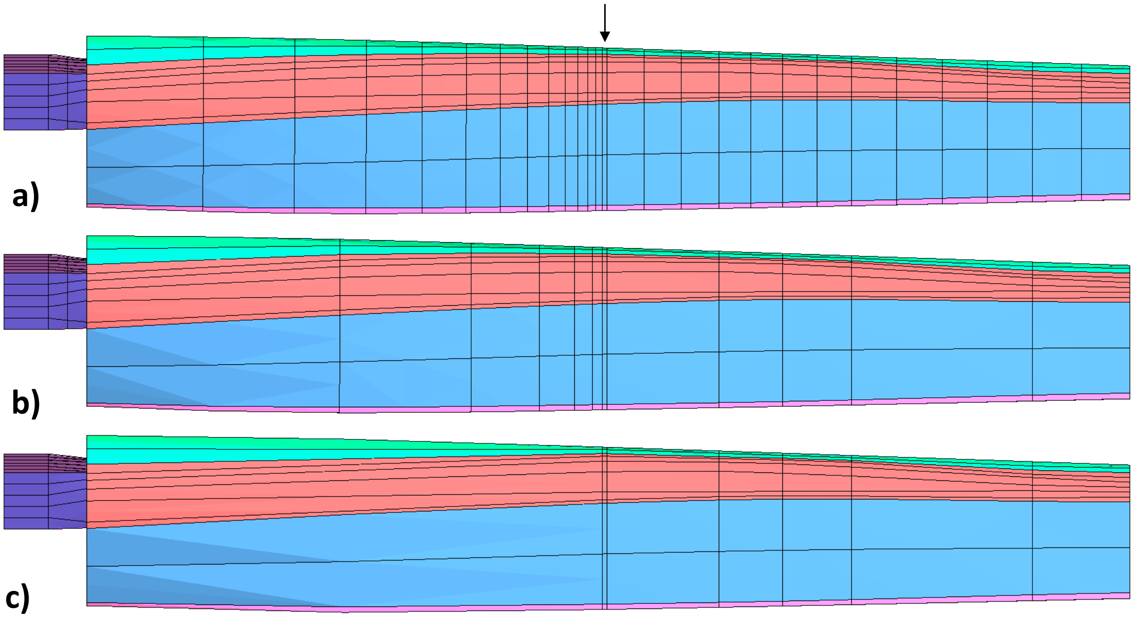


Figure 4.15 Blade meshes of three radial sizes: (a) fine, (b) medium, (c) coarse; mesh sizes in Table 4.4, colors represent material blocks, call out arrow marks mesh-driving CAD feature

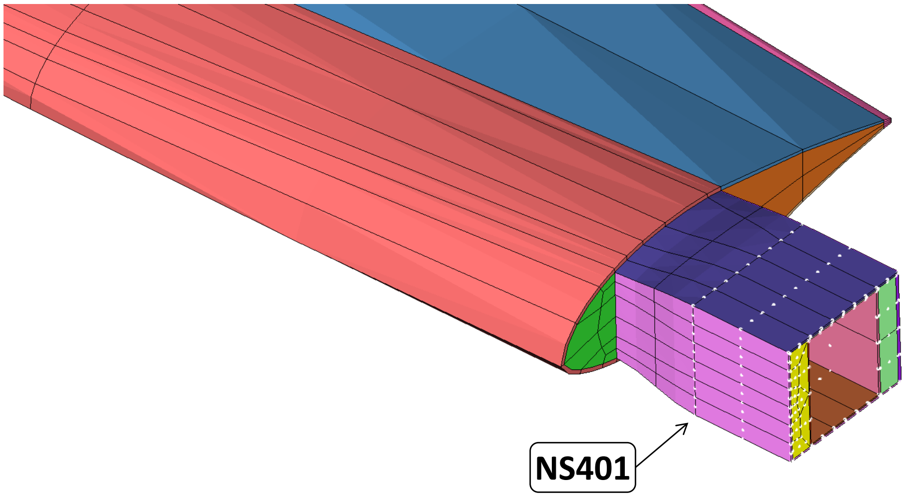


Figure 4.16 Blade root mesh, highlighting nodeset NS401, which is connected to joint J1

Table 4.4 Element and node counts for the three blade meshes

Radial Mesh Resolution	Number of Elements	Number of Nodes
(a) Fine	2,208	20,247
(b) Medium	1,047	9,831
(c) Coarse	623	6,023

As mentioned in the structural analysis description, the pitchcase was meshed in three separate parts, which are shown assembled in Figure 4.17. The pitchcase body mesh has five nodesets (Figure 4.18): NS401 and NS402 are the outboard surfaces connecting to the grips via joint J2; NS403 represents the inner surface, which is attached to the outer pitch bearing (joint J3); NS404 are the nodes on the inboard rim to which the inner pitch bearing carrier, joint J4, is bolted; and NS405 connects the pitchcase body to the pitch horn. The two blade grips share a single mesh, depicted in Figure 4.19 (left). They have two nodesets on their root surfaces, NS401 and NS402, which connect to the pitchcase via joint J2. The grips also have NS403 along their inside surfaces, which connects the blade via joint J1. The pitch horn (Figure 4.19, right) has two nodesets. Nodeset NS401 attaches to the pitchcase body by joint J5 and nodeset NS402 connects to the top of the pitch link by joint J6. In total, there are 229 elements and 538 nodes between the three flex parts that form the pitchcase assembly.

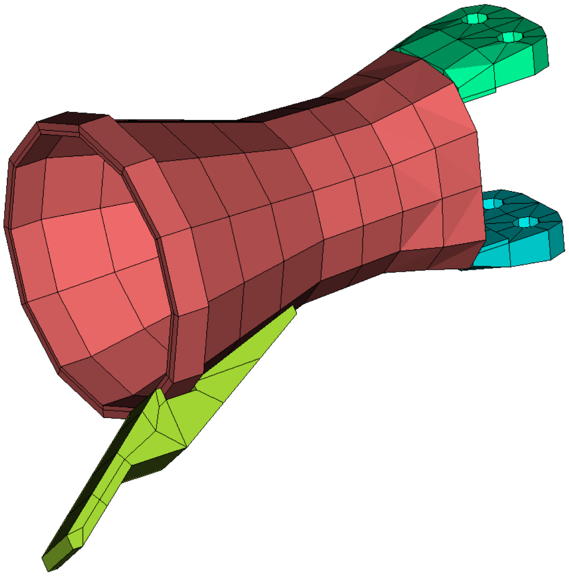


Figure 4.17 Assembled pitchcase mesh with body, blade grips, and pitch horn

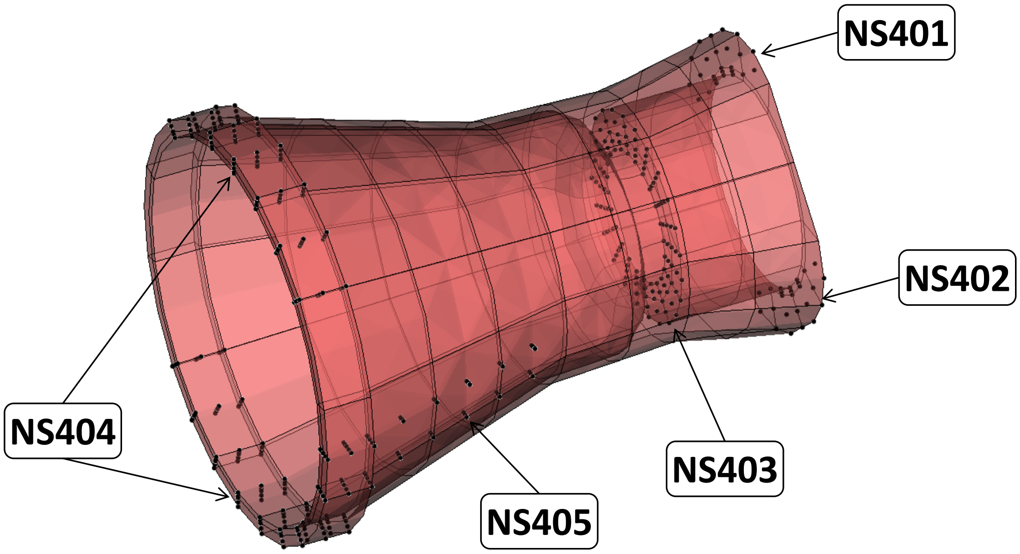


Figure 4.18 Pitchcase body mesh, with nodeset labels

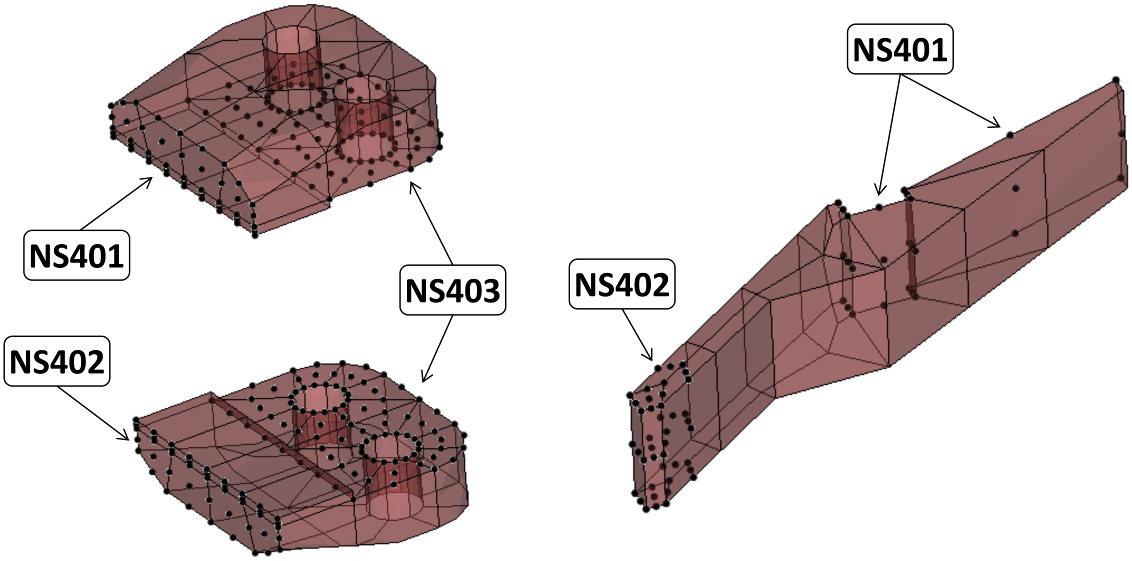


Figure 4.19 Meshes of the blade grips (left) and pitch horn (right), with nodeset labels

The pitch link (Figure 4.20) is simply modeled as a rod of three hexahedral elements with 63 nodes. It has nodeset NS401 at one end, which connects to the pitch horn via joint J6. The other end, nodeset NS402, connects to joint J7, which is used to represent a swashplate.

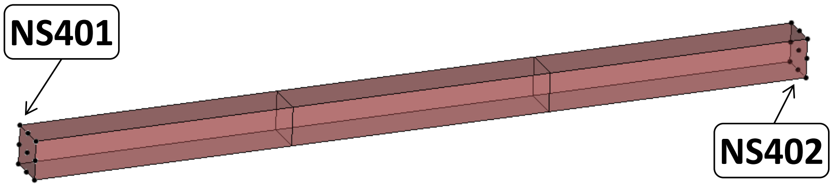


Figure 4.20 Simple pitch link mesh, with nodeset labels

4.2.4 TRAM Structural Analysis Model

The final structural analysis model (SAM) combines the flexible meshes with joint and material definitions to create a complete, multibody rotor model for the 3-D solver. The same meshes were used for both the single-bladed (Figure 4.21) and three-bladed (Figure 4.22) TRAM models. Although the TRAM rotor has a precone of 2°, this was omitted from both the single-bladed and three-bladed rotor models as an oversight.

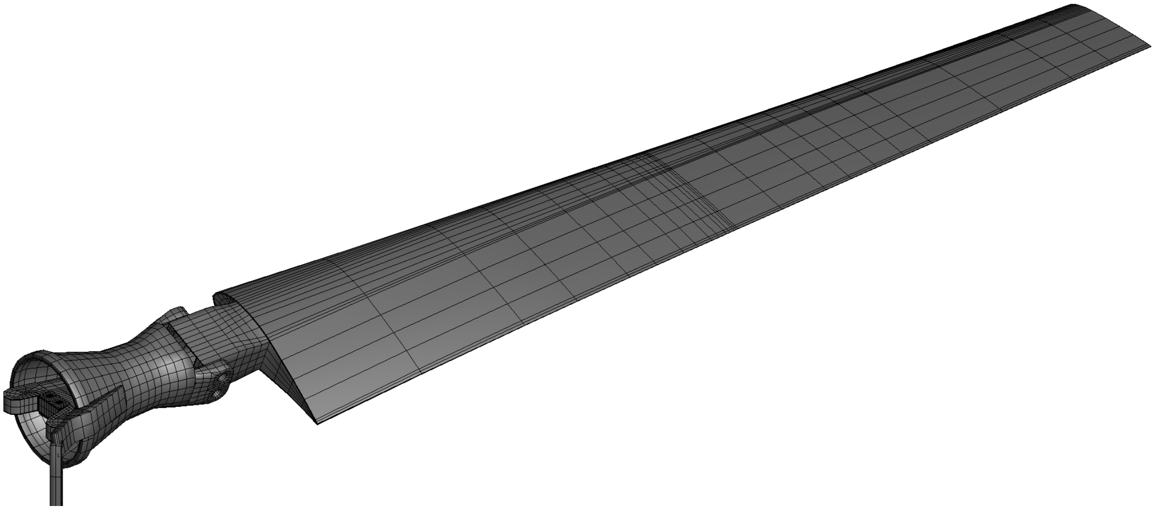


Figure 4.21 TRAM single bladed rotor structural analysis model

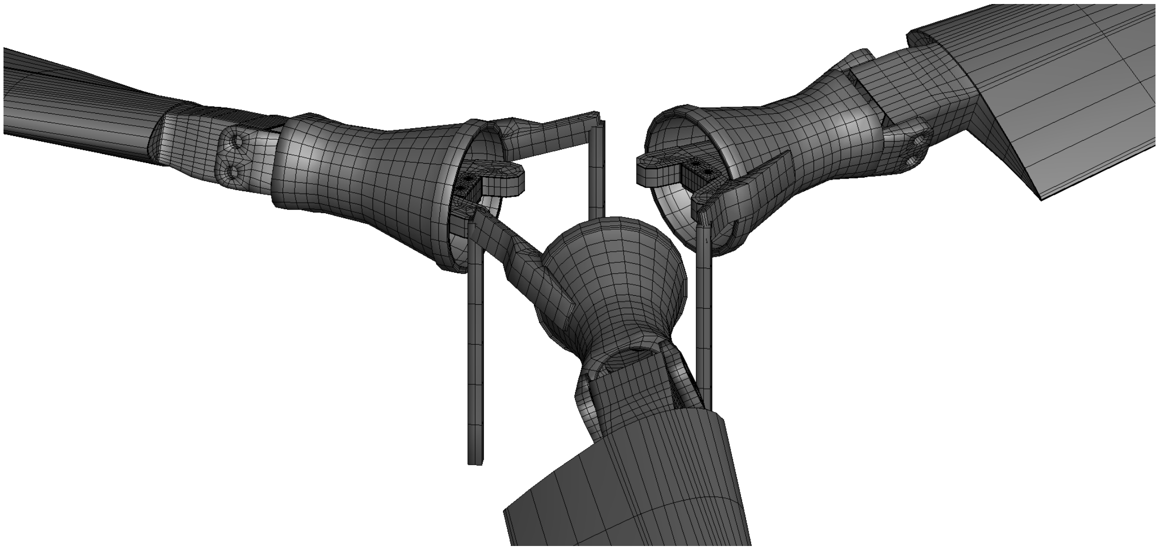


Figure 4.22 TRAM three-bladed rotor structural analysis model

In addition to containing the flexible part meshes, the SAM defines the properties of all the joints, as shown in Table 4.5 for the single-bladed model. These properties include the joint degrees of freedom, and whether any degrees of freedom are prescribed or forced. For this model, joints J1 (blade to grip), J2 (grip to pitchcase), and J6 (pitchcase to pitch horn) are completely fixed, thus six degrees of freedom are given a value of zero for each of these joints. The outboard pitch bearing (J3), in contrast, is modeled as a spherical bearing, so the last three of its degrees of freedom are given a value of one. The inboard pitch bearing, J4, is modeled similarly but with axial and vertical degrees of freedom released and therefore given values of 1. (Later adjustments led to this joint being given an in-plane degree of freedom, as described in Section 6.1.2.) Joint J7, connecting the pitch link to the pitch horn, is also a spherical bearing. Joint J8 is more interesting; it is used to represent a swashplate that is not explicitly modeled. As such, it is given a spherical bearing connection with yaw locked, but has freedom to move along its z-axis, creating a slider. This slider is not entirely free to move in response to loads on the rotor. Instead, it has its position along the z-axis commanded as a control input, indicated by its entry in the fourth column of Table 4.5. Finally, joint J5 at the hub represents a true constant velocity gimbal (rotation speed does not change with gimbal tilt), allowing pitch and roll motion but locked in yaw to allow torque transmission with prescribed stiffness representing the torque link. Greater discussion of the degrees of freedom is provided in Section 4.4: Gimbal Modeling.

Table 4.5 Structural analysis representation of the single-bladed TRAM model

ID	Name	TJDOF	PJDOF	FJDOF	Connections	
J1/P2	J_BtoG	[0 0 0 0 0 0]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F1/P1: NS401	F2/P3: NS403
J2/P4	J_GtoC	[0 0 0 0 0 0]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F2/P3: NS401, NS402	F3/P5: NS401, NS402
J3/P6	J_PBOuter	[0 0 0 1 1 1]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F3/P5: NS403	F4/P7: NS401
J4/P8	J_PBInner	[1 0 1 1 1 1]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F3/P5: NS404	F4/P7: NS402
J5/P9	J_Gimbal	[0 0 0 1 1 0]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F4/P7: NS403	Reference frame
J6/P10	J_CtoH	[0 0 0 0 0 0]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F3/P5: NS405	F5/P11: NS402
J7/P12	J_PlinkTop	[0 0 0 1 1 1]	[0 0 0 0 0 0]	[0 0 0 0 0 0]	F5/P11: NS402	F6/P13: NS401
J8/P14	J_PlinkBot	[0 0 1 1 1 0]	[0 0 1 0 0 0]	[0 0 0 0 0 0]	F6/P13: NS402	Swashplate slider

The three-bladed model has similar joints to those found in the single bladed model, but each repeated three times (with the exception of the gimbal). In that case, the joint properties are identical to those of the corresponded single-bladed joints found in Table 4.5; the only differences between the joints are in their part IDs (and those of their connections) and in the joint positions and orientations. One interesting feature of the three-bladed gimbal rotor is that flapping of any one blade is reacted by the pitch links of the other blades (a phenomenon that only occurs in rotors with more than two blades).

Finally, the SAM defines the material properties. In order to represent composite materials without driving the number of degrees of freedom too high to solve quickly, the homogenization method presented in Section 2.6.4.2 was employed. It calculates an anisotropic effective modulus matrix for a composite laminate composed of multiple plies of orthotropic material. Material modeling for the TRAM rotor has proven to be a source of difficulty during this research effort due to the dearth of available material specifications. The manufacturer drawings specified material designations (which are now discontinued) without documenting moduli, Poisson's ratios, or densities. This is a problem specific to this rotor, not a limitation of the methodology in general. Because of this, material properties (such as shear modulus and Poisson ratio) typical of the composites used in the rotor were obtained from a variety of sources, the most prominent of which was the composite materials handbook CMH-17 [142].

The total problem size was 43,863 degrees of freedom for the single bladed rotor model with the medium radial blade mesh: 1,709 finite elements, 17,364 nodes, about 30 different material types, and two hub load sensors (one at the gimbal and one below the pitch link to measure hub loads). The three-bladed rotor, with the medium radial blade mesh, had 131,583 degrees of freedom: 5,127 finite elements, 52,092 nodes, and four hub load sensors (one gimbal, three pitch links). A Generalized-α scheme, executed in a simple Newmark-β mode with damping (first order) with an azimuth resolution of 7.5° was used for trim solutions. Rotor trim is controlled via collective and cyclic inputs at the bottom of the pitch link, which is connected to a vertical slider joint (J8, Figure 4.10).

4.3 Aerodynamic Model

The X3D solver, in addition to structural dynamics, has built-in aerodynamics modeling capabilities. The provided aerodynamics analysis toolkit is low fidelity in comparison with the structural analysis and is intended to provide the minimum capability required to couple with CFD. The wake models available include uniform inflow momentum theory and blade element momentum theory (BEMT) for hover. In edgewise flight, wake modeling options include linear inflow as well as Maryland Freewake [30]. The linear inflow models implemented are the Coleman, Feingold and Stempin model, the Drees model, and the White and Blake model [143]. Aeromechanics analysis was performed for the TRAM rotor using these low order aerodynamics models as well as unsteady 3-D Reynolds averaged Navier-Stokes (RANS)-based CFD.

4.3.1 Low Order Aerodynamics

In hover, blade element momentum theory (BEMT) and uniform inflow models were considered. For edgewise flight, uniform inflow, Drees linear inflow, and free-wake models were used, in addition to CFD. The induced tip loss factor κ_h was set to 1.15 for all uniform inflow cases. The free-wake model was initialized using linear inflow for its first iteration. A single rolled-up tip vortex was used, with strength equal to the maximum bound circulation occurring from 50% radius to the tip, with an initial core size of 0.2 times the tip chord. An inboard vortex at the blade root was also used for some analysis cases but followed a prescribed wake trajectory. An example of the wake geometry with the single tip vortex is shown in Figure 4.23, and an example with an inboard vortex is shown in Figure 4.24.

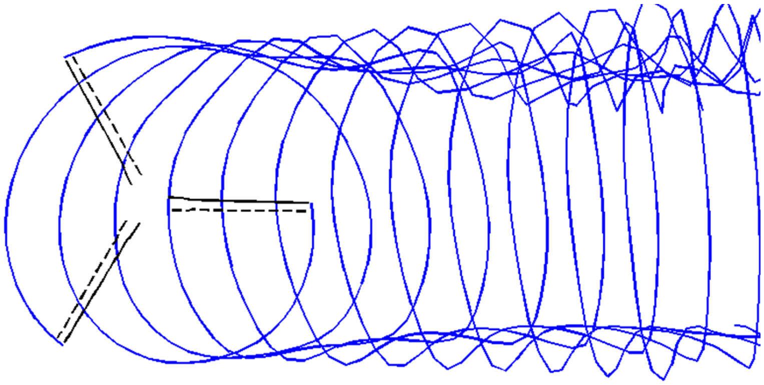


Figure 4.23 Sample free-wake trajectory with a single tip vortex trailer, black solid lines and doted lines show blade 1/4 chord (bound circulation points) and 3/4 chord (control points); $C_T/\sigma = 0.128, \mu = 0.15, \alpha_s = -2°$

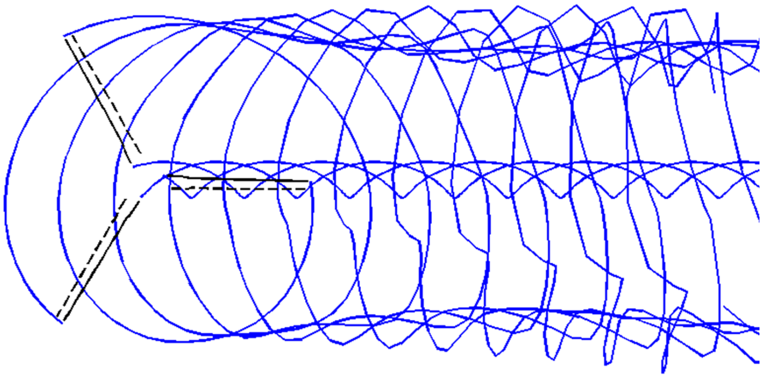


Figure 4.24 Sample free-wake trajectory with a tip vortex trailer and an inboard prescribed wake vortex, black solid lines and doted lines show blade 1/4 chord (bound circulation points) and 3/4 chord (control points); $C_T/\sigma = 0.128$, $\mu = 0.15$, $\alpha_s = -2°$

The low-order aerodynamics model uses a lifting-line model for analyzing the blade. The TRAM blade was discretized into 21 aerodynamic panels. Aerodynamic loading at each panel is passed by the fluid structure interface (FSI) to one of the eight aerodynamic segments defined on the 3-D structural analysis model (Figure 4.25). Blade section aerodynamic coefficients are determined using an airfoil table look-up. C81 airfoil tables for the large civil tiltrotor (LCTR) project [144, 145] were provided by NASA and used in place of the actual V-22 Bell XN airfoils [146], which are proprietary and unavailable. Each panel was assigned to one of four LRA (LCTR reference airfoils): 28%, 18%, 12%, and 9% thick.

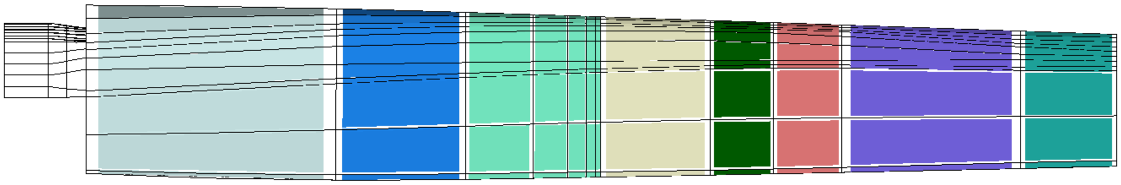
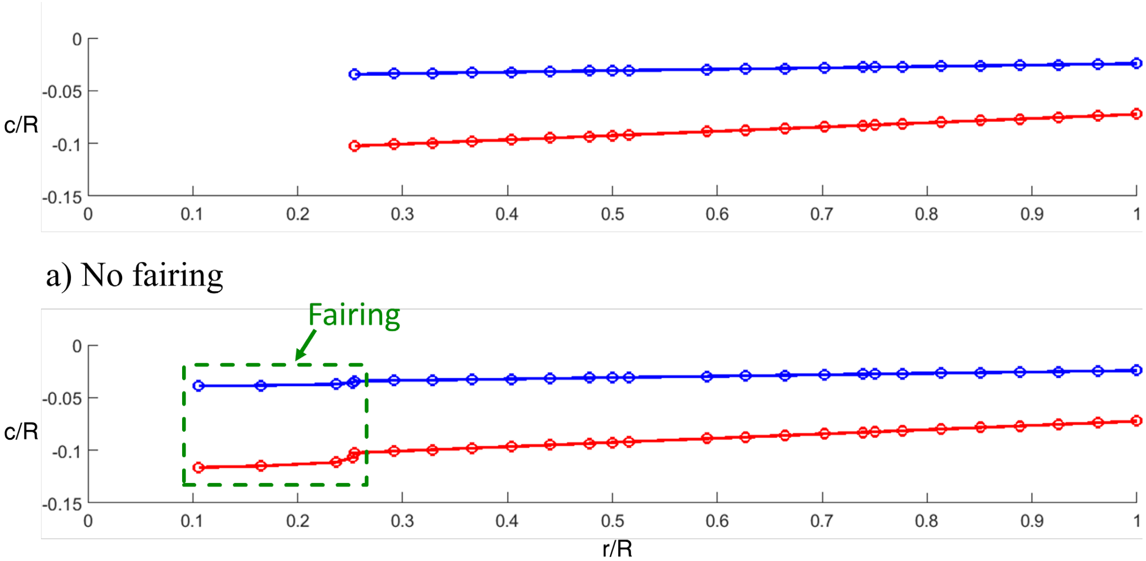


Figure 4.25 TRAM Blade mesh with eight aerodynamic segments identified

The root portion of the TRAM blade is covered by an aerodynamic fairing. The fairing was modeled by adding four additional panels inboard of the root of the blade, as shown in Figure 4.26, and disabling structural deflections in the region because no structural description of the fairing was available. The airloads from the inboard fairing are instead applied to the root of the blade structural model (the first aerodynamic segment in Figure 4.25). Although the chord and twist of the fairing were known, the airfoil properties of the fairing were also unavailable. For this reason, the root airfoil (LRA-28) was extended to the fairing.



a) No fairing

b) With fairing

Figure 4.26 TRAM blade lifting line stations (a) without and (b) with the aerodynamic fairing. Each point represents one aerodynamic node where properties such as chord and twist angle are provided; the blue represents the ¼-chord line and the red represents the ¾-chord line

4.3.2 CFD/CSD Coupling

Helios [147] was used for CFD predictions. It uses a dual mesh paradigm: here unstructured RANS (NSU3D) is used as the near-body solver and Cartesian Euler is used as the off-body solver. The CFD mesh used (Figure 4.27) is a fine mesh used for hover predictions by Wissink et al. [94] containing 9.27 million grid points for the near body and 26 million grid point for the off-body mesh. An azimuth resolution of 0.25° was used (implicit near-body, explicit off-body; see [148] for details). The simulations were performed on 128 cores.

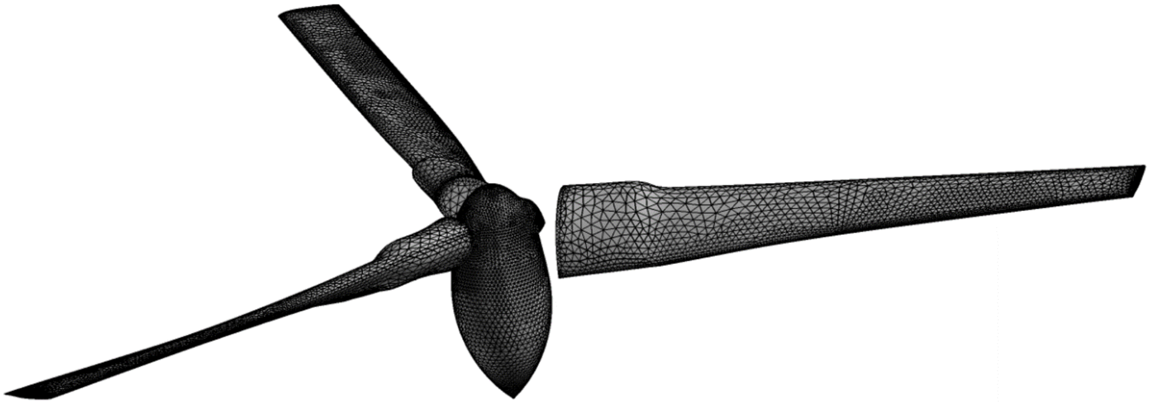


Figure 4.27 CFD mesh of the TRAM rotor

The CFD/CSD analysis was performed on the single-bladed TRAM structural analysis model with the medium radial resolution blade mesh. The fluid structure interface consisted of 55,600 on-surface nodes of the near-body CFD mesh and 391 surface elements of the structural mesh. The X3D solver provides an interface for coupling external CFD software, available both as a file-based input/output as well as a Python-based module. The Python-based module is part of the Helios framework. All current generation rotorcraft CFD software (including those in Helios) use a 1-D beam-like interface to exchange deformations and airloads. Therefore, even if inexact in the context of 3-D, such an interface (defined as level-I, see [8] for details) is used here; it is the only interface that allows CFD/CSD coupling with no changes in the CFD mesh-motion domain.

To pass structural deformations from CSD to CFD using this level-I interface, the 3-D deformation field must be reduced to a 1-D beam like deformation along the 1/4 chord line. This is accomplished by using the deformed positions of nodes at the leading edge, trailing edge, upper surface, and lower surface of airfoil sections to extract beam-like deformations, in terms of three translations (u, v, w) at ¼ chord and three Euler rotations $(\theta_x, \theta_y, \theta_z)$ of sequence Z-Y-X, for each blade section.

To return airloads from CFD to CSD, the predicted aerodynamic surface traction is collapsed into segmental normal and chordwise (dimensional) forces and pitching moment about 1/4-chord. The aerodynamic forces and moments are then distributed over the surface nodes of the structural mesh, which are identified for each segment during model development as described in Section 2.5.3.5. Sample aerodynamic segments for a notional blade mesh are shown in Figure 4.28 (from [8]).

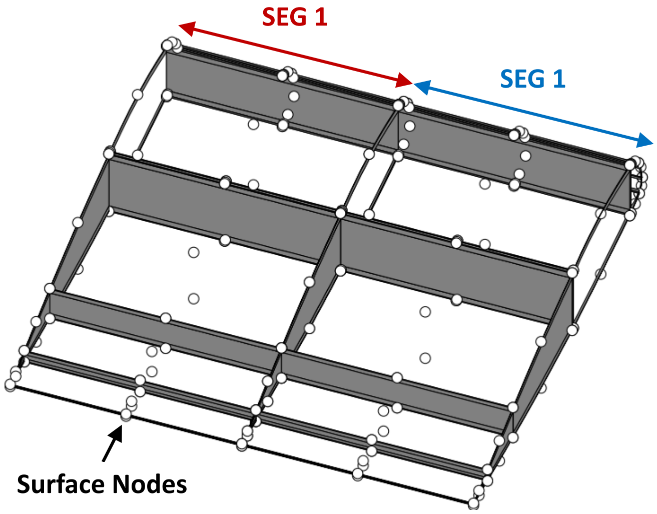


Figure 4.28 Structural mesh aerodynamic segments

Normal and chordwise aerodynamic force components are applied to each structural mesh surface node following an assumed distribution: the normal force on a given node is assumed to vary linearly with its chordwise position, and the chordwise force to vary quadratically with its thickness position. Thus, for a given node:

$$f_N = \alpha_0 + \alpha_1 \xi \quad \text{and} \quad f_C = \beta_0 + \alpha_1 \eta^2, \tag{4.1}$$

where f_N and f_C are the normal and chordwise forcing on the node, ξ and η are the chordwise and thickness-wise position of the node, and the α and β terms are three coefficients to be solved once for all nodes in the segment. Based on these nodal forces, the total segmental airloads (normal force F_N, chordwise force F_C, pitching moment $M_{1/4}$) are

$$F_N = \sum f_N, \;\; F_C = \sum f_C, \;\; \text{and} \;\; M_{1/4} = \sum(\xi f_N - \eta f_C), \tag{4.2}$$

where the sum is taken over all of the surface nodes in the segment. For n surface nodes:

$$\begin{bmatrix} F_N \\ F_C \\ M_{1/4} \end{bmatrix} = \begin{bmatrix} n & \sum\xi & 0 \\ 0 & \sum\eta^2 & n \\ \sum\xi & \sum(\xi^2 - \eta^3) & -\sum\eta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \beta_0 \end{bmatrix}. \qquad 4.3$$

The matrix depends only on the surface node coordinates and is evaluated once for the segment, and the constants are solved for at every time step. The internal stresses are only as good as this assumed force distribution, but because the correct normal force, chordwise force, and pitching moments are ensured, they are expected to follow the characteristics of the external loading.

For the particular case of the TRAM blade, the root of the spar and the pitch case is covered by an aerodynamic fairing from approximately 10% to 25% radius, which can be seen in Figure 4.27. This fairing is not included in the structural model because details of its construction were not available, but is a part of the aerodynamic model. The forces over the fairing must be applied to the structure, and this is accomplished by adding the normal force, chordwise force, and pitching moment over the fairing to the inboard-most aerodynamic segment. The CFD mesh also includes a center body (nacelle) geometry. The loads on the center body are not passed to the structural solver for trim (there is no structural model of the center body), but modeling the flow over the center body is important as its wake can have a significant impact on the flow experienced by the blades.

The CFD/CSD iterations then proceed as per the typical delta coupling procedure. A total of eight iterations were carried out with CFD airloads exchanged over 1, 1, and six sets of 1/3 revolution (for a 3-bladed rotor, 1/3 revolutions are adequate to construct a nominal periodic solution until convergence to the final solution). The CFD/CSD simulation time varied, taking around 3 to 4 days in total, of which structural solver's share was 10 hours: about two hours for the baseline solution and one hour on average for each subsequent trim solution. The subsequent solutions take less time because the previous iteration trim constants are used as a starting solution, and the Jacobian from the baseline solution is re-used.

4.4 Gimbal Modeling

Initial analysis of the TRAM rotor modeled the rotor as a single articulated blade with a free-flapping hinge at the center of the radius to simulate the gimbal. This approximation was made to simplify the analysis, but the magnitude of its effect on the predicted rotor aeromechanics was unknown. A more rigorous model of the gimbal motion was desired. The most straight forward means of considering the gimbal in the 3-D analysis is to use the three-bladed model presented above in Section 4.2. However, the X3D solver is currently run on a single processor for complex geometries, leading to large solution times for even the one-bladed rotor. A more elegant solution to modeling the gimbal in X3D, which would not dramatically increase solution times, was therefore desired.

Gimbaled rotors are similar to two-bladed teetering rotors, in that they combine modes of both articulated and hingeless boundary conditions with regards to flapping. For a teetering rotor, odd harmonics of the flap moment result in tilting of the hub, causing the blade to act like an articulated rotor with zero hinge offset. For even harmonics, the bending moments from one blade are reacted against those from the other blade at the hub, resulting in a net bending load at the root with no tilt, which is to say the blade acts like a hingeless rotor [149]. Similarly, a gimbaled rotor acts like a hingeless rotor for all harmonics of airloads that are integer multiples of the number of blades

(pN_b/rev, where p is an integer), and acts like an articulated rotor for all other harmonics ($pN_b \pm 1$/rev). For $N_b > 2$, an additional complication is the flap-pitch coupling of one blade due to the control system stiffness of the other blades. This is typically a 1/rev effect (because pitch input is at 1/rev) and its neglect can be assumed to affect the control loads more than blade loads, beyond a small increase in the first (articulated) flap frequency.

For the three-bladed gimbaled TRAM rotor, this means that there is zero gimbal tilt at 0/rev, 3/rev, 6/rev, and so on, with all flap motion at these harmonics resulting only from elastic deformation of the coning flexbeam. To simulate this behavior with a single-bladed model, the 3-D solver is modified to suppress the 0/rev, 3/rev sine, and 3/rev cosine components of flapping at the hub joint. Higher multiples of N_b/rev are left unaltered because their contribution to loads was observed to be minimal.

The suppression of these gimbal motions is accomplished by adding them as additional trim variables (g_0, g_{3c}, g_{3s}) in the solution procedure (Table 4.6). These trim variables are commanded into the gimbal motion; these, plus the response of the gimbal due to rotor aerodynamics, yield the net flapping angles β_0, β_{3c}, and β_{3s}. These net angles are then set as additional trim targets with values of zero. This method requires adding some stiffness to the gimbal joint so that command inputs can be prescribed (see Section 2.6.3). The effect of this stiffness on the rotor dynamics was found to be minimal, as discussed in Section 5.1.3.6, however, its addition is the penalty to be paid for the implementation of this procedure.

Table 4.6 TRIM variables and targets for gimbal rotor analysis, with additional terms for simulating gimbal motion through suppression of flapping at integer multiples of N_b/rev

	Trim Variables	Trim Targets
	θ_0	C_T/σ = prescribed
Standard	θ_{1c}	β_{1c} = prescribed
	θ_{1s}	β_{1s} = prescribed
	g_0	$\beta_0 = 0$
Gimbal	g_{3c}	$\beta_{3c} = 0$
	g_{3s}	$\beta_{3s} = 0$

Chapter 5: Results of the TRAM Analysis

This chapter presents results from integrated 3-D analysis of the TRAM proprotor, with all predictions compared to experimental measurements wherever available. It begins by examining pure structural dynamics of the proprotor, including deflections and modal frequencies. Next, it considers (ideal) hover performance and loads using simple aerodynamic models. Finally, predictions for airloads, blade loads, and internal stresses are carried out in conversional flight, systematically, comparing lower-order aerodynamics models to CFD/CSD coupling with RANS.

5.1 Structural Dynamics

This section describes the TRAM proprotor structural dynamics analysis. It includes mass and center of gravity analysis, static deflection predictions for the flexbeam compared to experimental measurements, nonrotating and rotating natural frequencies of the model, and also an extraction of the blade structural stiffness properties in order to compare with documented measurements (to assess the implications of precise material property data being unavailable).

5.1.1 Mass Properties

Accurate material properties for the TRAM rotor were unavailable, as discussed in Chapter 4. The density information for the blade materials was estimated from the part weights (measured) and part volumes (CAD). The resulting total weights and C.G. locations are within 3% of the measured values, suggesting a good match between the CAD model and the physical blade geometry (Table 5.1). The chordwise C.G. offset of the whole blade model is 2.14 inches from the root leading edge, 1.83 in. behind the local leading edge, and 0.16 in. ahead of the local quarter chord at the spanwise C.G. location.

Table 5.1 Error in TRAM blade mass properties, measured versus CAD-based X3D structural analysis model

Property	**Error**
Weight (grams)	-2.79 %
Spanwise C.G. (in.)	-1.47 %
Chordwise C.G. (in.)	-0.47 %

5.1.2 Flexbeam Static Analysis

The TRAM flexbeam model was validated and material properties fine-tuned by comparing the results of static analysis to experimental static deflection data [extracted from unpublished NASA Report 194822, 1996, with permission]. Fine-tuning is needed because the original material properties of the orthotropic plies are not available; only their manufacturer's designations are known. Current materials of similar designations were considered in their place. However, there is considerable variation in documented properties from various sources. The effect of this variation was studied and the baseline values were selected to reproduce the static data as closely as possible. The primary variations observed were in the in-plane shear modulus and the Poisson ratio.

The static test for flapwise bending of the flexbeam is shown in Figure 5.1 [from NASA Report 194822], and a similar setup was used for chordwise bending. The model (Figure 5.2) closely approximates the test. The nodes at the root end of the flexbeam, clamped by the mounting fixture in the experiment, are set as boundaries with zero deformation in the analysis.

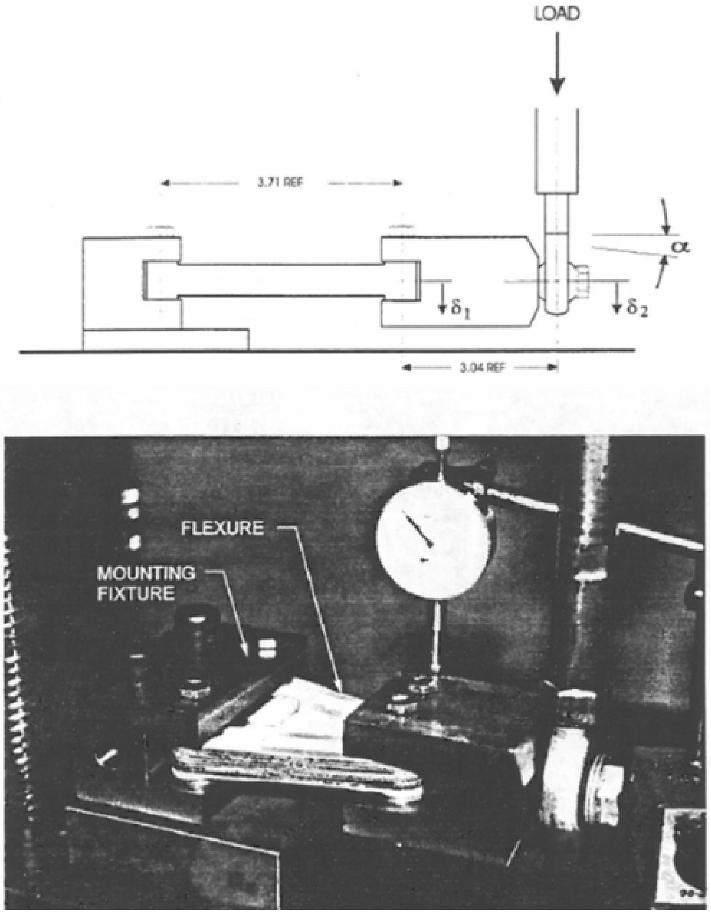


Figure 5.1 Schematic and photograph of the TRAM flexbeam flap bending test setup

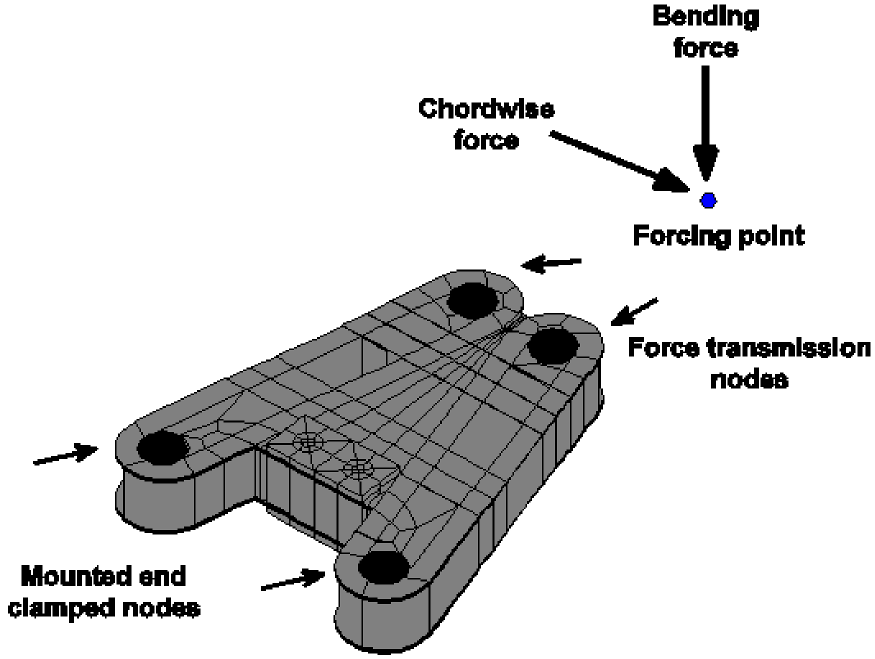


Figure 5.2 Flexbeam bending test model configuration for X3D static analysis

The actuator is modeled by a tip joint, to which the nodes at the loading end of the flexbeam are rigidly connected. The forcing is applied to the vertical (or horizontal, for chordwise deflection) degree of freedom of the joint. The deflection and rotation (δ_2, α) of the joint are then compared to measurements.

Figure 5.3 compares the analysis to experimental measurements for the deflection of the flexbeam under static flap and chordwise bending loads. The flapwise deformations are well captured, but the chordwise deformations are under-predicted by 50%. Material modeling for the flexbeam was difficult due to the dearth of available material specifications from manufacturers, so material properties (such as shear modulus and Poisson ratio) representative of similar materials were used. The baseline in-plane shear and Poisson ratio for the woven fibers used are 0.6 MPa and 0.2 respectively. To determine sensitivity to material properties, in-plane shear was varied between 0.3 MPa and 0.9 MPa and the Poisson's ratio between 0.2 and 0.5: the effect of these variations is shown in Figure 5.4. The chordwise deformations remain outside the range of these variations and hence unresolved. However, the effect of errors in chordwise stiffness on the model dynamics were found to be minimal, as discussed in Section 6.1.1.

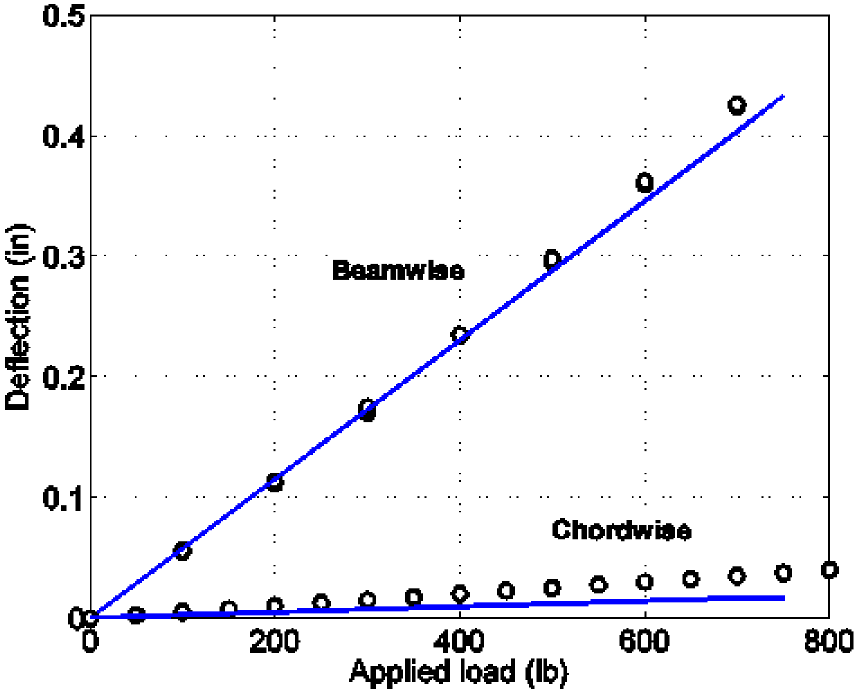


Figure 5.3 Comparison of analysis to experiment for flapwise and chordwise deflections of the flexbeam under static loading

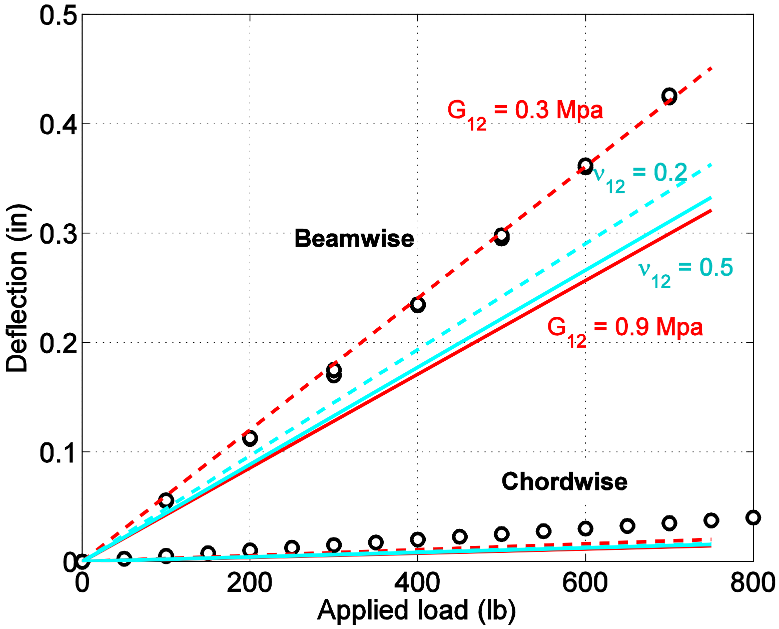


Figure 5.4 Static flexbeam deflections with varying material properties

5.1.3 Rotor Frequencies

The natural frequencies of the rotor are examined in this section. The analysis begins with a simplified single load path rotor, consisting of a single 3-D blade mesh attached to an equivalent root spring. It then proceeds to increase in fidelity, progressing to a dual load path model with a blade, a flexbeam, and a pitchcase (but no controls); followed by a dual load path model (all root components, including controls); and finally the full three-bladed model. At each step, frequencies obtained from eigenanalysis in a vacuum are compared to experimental measurements, which are available only for the nonrotating cases, as well predictions from a prior analysis using a beam-based rotor analysis code [95].

5.1.3.1 Isolated Blade, Nonrotating

The measured blade frequencies from a suspended free/free rap test are compared with predictions for three blade mesh sizes in Table 5.2. Significant errors can be seen in the isolated blade frequencies; even the first flap bending frequency shows errors of 30-50%. The frequencies begin to converge from the coarse mesh to the medium. As shown above (Table 5.2), mass properties seem to be satisfactory; therefore discrepancies are believed to be due to incorrect estimates of the unknown material properties. However, a free/free rap test is difficult, and large discrepancies are not unexpected.

Table 5.2 Error between X3D analysis and measured suspended free/free rap test blade frequencies for three meshes

Mode	**Frequency Error**		
	Coarse	Medium	Fine
1: Flap	43%	28%	26%
2: Flap	107%	70%	66%
3: Flap	101%	56%	34%
4: Lag	85%	55%	45%
5: Flap/Torsion	53%	28%	23%

5.1.3.2 Single Load Path Model, Nonrotating

Typical analyses today model a blade as an elastic beam with an equivalent root spring representing all inboard components including, in the case of TRAM, the coning flexbeam, pitch case, and bearings; this results in a single load path at the root. To begin the 3-D structural analysis, a similar technique is followed. The 3-D FEM model of the blade is attached to an equivalent root spring, forming a simplified single load path model of the TRAM blade.

Johnson [95] uses a single load path model with an equivalent root spring with stiffnesses obtained by matching deflection measurements from the TRAM rotor test. For the 3-D single load path model analysis, root spring is given the same properties used by Johnson. The frequencies for this model with different three blade mesh sizes are presented in Table 5.3.

Table 5.3 Blade frequencies (Hz): Nonrotating, zero collective, single load path model, using equivalent root spring based on measured deflections

Mode	Rap Test	CAMRAD Beam	X3D Coarse	X3D Medium	X3D Fine
Flap	13.9	13.8	14.3	13.9	13.9
Lag	24.1	34.1	36.8	36.0	35.8
Flap	46.3	53.4	70.8	61.5	61.0
Flap	124.4	145.0	195.8	168.3	164.3
Lag	195.0	177.0	289.7	256.3	249.8
Torsion	227.8	202.0	265.6	235.7	229.0
Flap	358.0	317.0	381.5	329.8	319.8

For each case, the root stiffnesses can be corrected to match the measured first flap and torsion frequencies (13.9 Hz and 227.8 Hz respectively). The measured lag frequency was noted as suspect by Johnson [95]: if the root stiffness in lag is corrected to match the rap test nonrotating frequency, it produces an unreasonably low rotating lag frequency (less than 1/rev, whereas the rotor was designed for greater than 1/rev). Therefore, the measured value is kept here without correction. Note that the previous study also used an additional correction (+4.7 %) in root lag stiffness in order to produce a lag frequency expected to more accurately represent the real rotor. This correction is not used here, as the present lag frequencies are already higher and therefore close to the expected frequency. The corrected root stiffnesses are shown in Table 5.4. The current blade is stiffer, therefore all corrections are of opposite in sign to those from the previous study. Note that the fine and medium meshes only need correction in torsion stiffness and only by a very small amount in the case of the fine mesh – the baseline frequencies are almost correct to begin with. The coarse mesh requires corrections in both flap and torsion. If these corrections appear confusing, it is because they are. This highlights the trial and error, empirical, and ad hoc nature of current generation structural modeling – significant drawbacks which this dissertation seeks to address.

Table 5.4 Root stiffnesses (Nm/rad) needed to match first flap and torsion frequencies, single load path model

	Flap	Lag	Torsion
Measured	7582	51811	6996
Match Fine	7582	51811	6696
Match Medium	7582	51811	5796
Match Coarse	7100	51811	4475
Johnson	7731	54252	10587

The corrected frequencies are re-compared with test and with Johnson's corrected frequencies in Table 5.5. As expected, the corrections only affect the first flap and torsion frequencies (and lag for Johnson) while leaving the higher flexible modes unaffected. The frequencies very greatly from the coarse blade mesh to the medium mesh, which begins to converge with the fine mesh. For the more detailed models, only the medium blade mesh is considered.

Table 5.5 Blade frequencies (Hz): Nonrotating, zero collective, single load path model, using equivalent spring based on corrected root stiffness

Mode	Rap Test	CAMRAD Beam	X3D Coarse	X3D Medium	X3D Fine
Flag	13.9	13.8	14.3	13.9	13.9
Lag	24.1	34.1	36.8	36.0	35.8
Flap	46.3	53.4	70.8	61.5	61.0
Flap	124.4	145.0	195.8	168.3	164.3
Lag	195.0	177.0	289.7	256.3	249.8
Torsion	227.8	202.0	265.6	235.7	229.0
Flap	358.0	317.0	381.5	329.8	319.8

5.1.3.3 Dual Load Path Model, Nonrotating

Next, as a step towards the full model, a dual load path model is considered. In this model, two 3-D hub components are introduced: the flexbeam and the pitch case. These are connected together via a pair of inboard and outboard centering bearings along two load paths. This configuration provides the flap and lag stiffnesses out of the flexbeam and its constraints, rather than relying on an equivalent root spring. It does not, however, include the controls, so stiffness must be added to the pitch bearings.

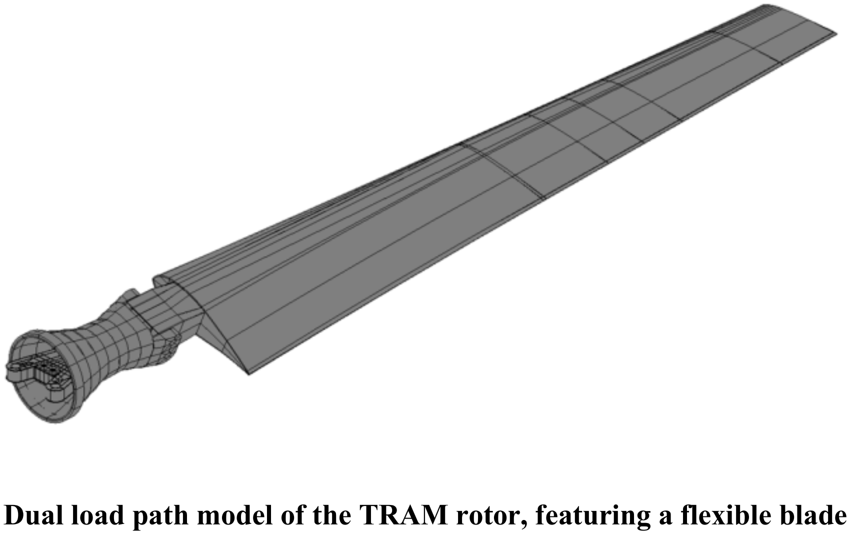


Figure 5.5 Dual load path model of the TRAM rotor, featuring a flexible blade, pitchcase, and flexbeam

Table 5.6 shows rotor frequency predictions for the dual load path model with multiple values of torsional stiffness applied at the inboard pitch bearing. Zero stiffness is applied to the flap and lag rotation degrees of freedom of either bearing, as well as the pitch degree of freedom for the outboard bearing. The second row of the table presents predicted frequencies with zero pitch stiffness (hence the rigid body mode in torsion, T). Here, the flap (F) and lag (L) frequencies are predicted from first principles. The flap frequency is close to exact, but the lag frequency shows a significant error, approximately a 42% deviation from the matched root stiffness value of 36 Hz. Further discussion of the sources of this error in first lag frequency is provided in Section 6.1.

Table 5.6 Blade frequencies (Hz): Nonrotating, zero collective, dual load path X3D model with varying inboard pitch bearing stiffness

	Pitch Bearing Stiffness (N/rad)			**Modal Frequencies (Hz)**						
	Flap	Lag	Torsion	F	L	F	F	L	T	F
Measured	-	-	-	13.9	24.1	46.3	124	195	227.8	358
Analysis	0	0	0	14.1	51.9	62.3	168	265	0.0	315
Analysis	0	0	6995	14.0	51.9	62.3	168	267	204.9	315
Analysis	0	0	8390	14.0	51.9	62.3	168	269	227.4	316

Introducing the measured pitch stiffness for the inboard bearing torsional stiffness places the torsion frequency near the measured value, but lower (third row). This is contrary to the single load path case where the measured stiffness placed the torsion frequency higher than measured. This is because of the flexibility of the pitch cone between the bearing and the main blade; the single load path model had assumed a rigid connection and was therefore stiffer. To accurately match the measured torsion frequency, the bearing stiffness must be corrected by an increase of approximately 20% (fourth row).

5.1.3.4 Full Single-Bladed Model, Nonrotating

Modeling the full TRAM single blade rotor involves adding pitch controls to the dual load path model discussed above (Figure 5.5). With the dual load path and pitch control, all blade root conditions can be reproduced from first principles. In this model (Figure 5.6), the pitch horn is attached to the pitch case, and is connected to the top of the pitch link by a spherical bearing. The bottom of the pitch link is attached to a vertical slider joint, representing the swashplate. Unlike the earlier dual load path model, the inboard spherical bearing is free in pitch; control system stiffness stems from the combination of the elastic stiffness of the pitch horn and pitch link, and from a spring placed on the vertical slider which represents swashplate and actuator stiffness.

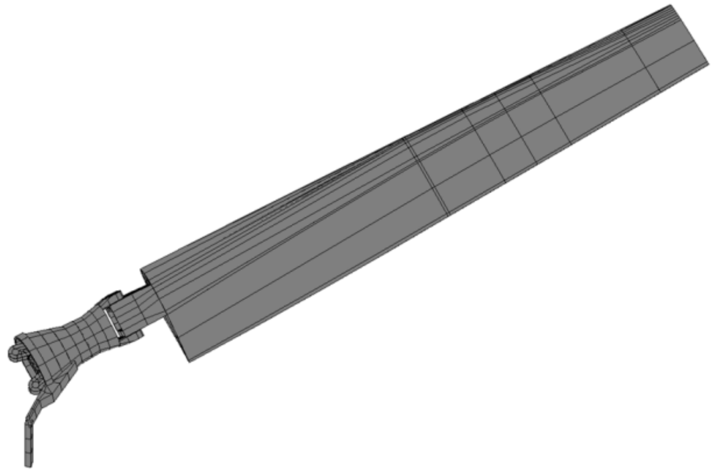


Figure 5.6 Full single-blade model, with dual load path and pitch control

Table 5.7 provides the nonrotating frequencies for the full single-bladed rotor model. With the vertical slider representing the swashplate locked (second row), the pitch degree of freedom is too stiff, leading to a torsion frequency of 262 Hz, implying the slider spring is needed to model swashplate flexibility (which is consistent with actual aircraft and test articles). Setting the stiffness of the spring equivalent to that used earlier on the inboard pitch bearing to match torsional stiffness (third row) yields a torsional frequency which is too low. This is due to the additional flexibility in the pitch horn and pitch link. The final row in the table shows slider stiffness required to match the torsional frequency, and thus properly model control system stiffness.

Table 5.7 Blade frequencies (Hz): Nonrotating, zero collective, dual load path X3D model with varying inboard pitch bearing stiffness

	Slider Stiffness	Modal Frequencies (Hz)						
	(N/m)	F	L	F	F	L	T	F
Measured	-	13.9	24.1	46.3	124	195	227.8	358
Analysis	Locked	14.1	52.0	62.3	167	301	261.7	315
Analysis	949000	14.1	52.0	62.3	166	266	200.2	315
Analysis	1450000	14.1	52.0	62.3	167	268	227.0	315

5.1.3.5 Single-Bladed Model, Rotating

The rotating rotor frequencies for the single load path model, with the 3-D blade attached to an equivalent root spring, are shown in Figure 5.7 for three blade meshes. These modal frequencies are for a single cantilevered blade (locked gimbal). The corrected root stiffness values are used so the effect of mesh resolution is seen only in the higher modes.

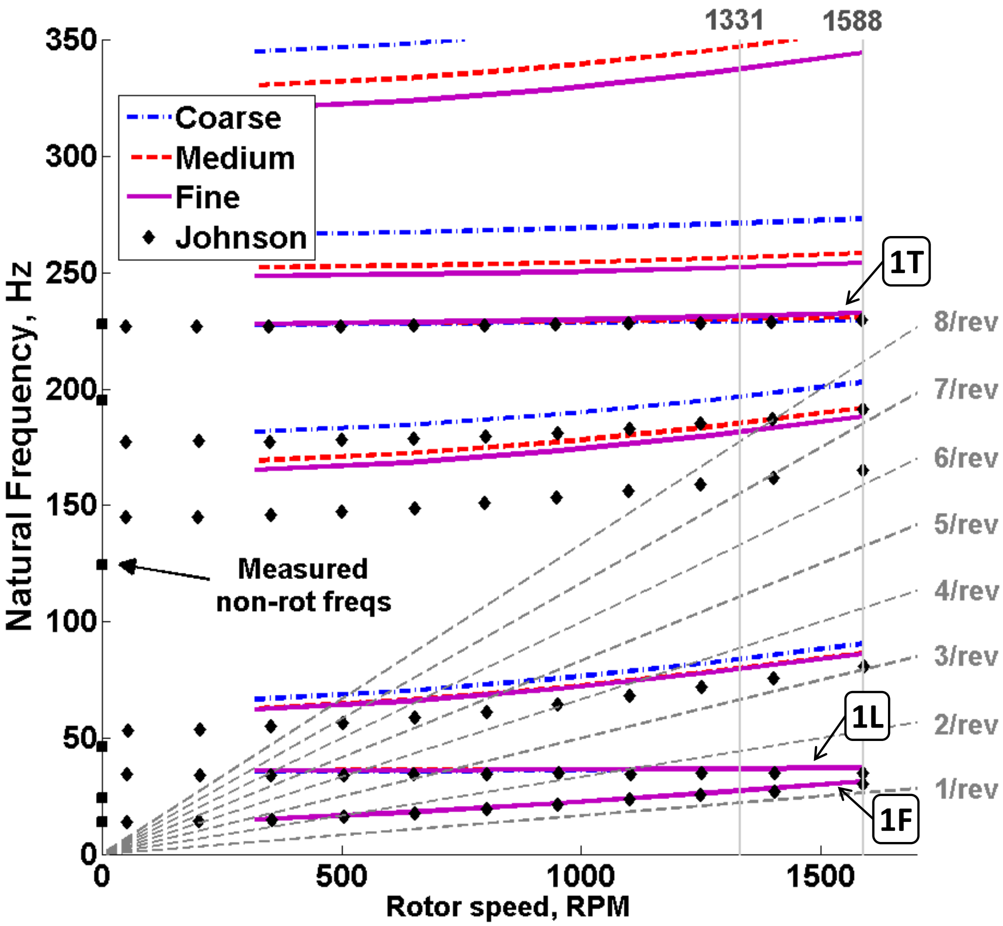


Figure 5.7 Blade frequencies (Hz): Rotating, zero collective, single load path model, three mesh sizes; first flap, lag, and torsion mode are marked, others show significant coupling

The higher modes show discrepancies consistent with those seen earlier in the non-rotating frequencies. Predictions from Johnson's analysis (corresponding to beam cantilever modes) are shown for reference. The main discrepancies are in the prediction of second flap and second lag frequencies. Both of these errors appear to stem from inaccurate blade material properties. The fine and the medium mesh results are close to one another, implying satisfactory convergence. The medium mesh blade is used for all further analysis, of rotating frequencies for the dual load path, full single blade, and three-bladed models, and for all aeromechanics predictions.

The rotating frequencies for the dual load path without pitch control model and the full model are compared with the single load path model (all with the medium blade mesh) in Figure 5.8. The medium blade mesh is used in all models. The results are almost identical in flap and torsion; the major discrepancy is in lag. The first lag mode, expected to be just above 1/rev, is almost at 2/rev; the second lag, expected to be just above 7/rev, is above 10/rev. The deviation of the first lag frequency from the refined models compared to those from the single load path model is examined further in Section 6.1.

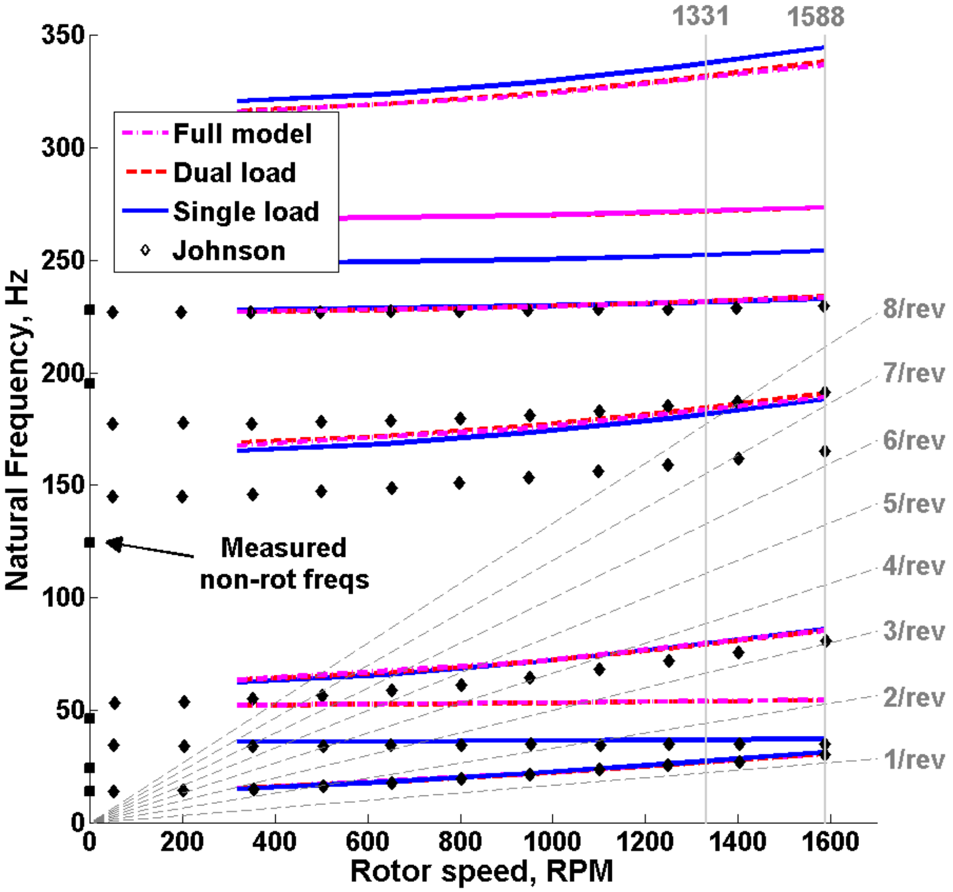


Figure 5.8 Blade frequencies (Hz): Rotating, zero collective, all models, medium blade mesh

The frequencies at the helicopter mode RPM are listed in Table 5.8. Predictions from Johnson are again shown for comparison. Note that Johnson and the single load path (SLP) predictions are with equivalent root stiffnesses based on measured data. The dual load path (DLP) and full models predict the root stiffnesses from first principles. The mode shapes 3-6 of the full model (beyond the first flap and lag modes at 1.15 and 2.02/rev respectively) are shown below. Mode 3 is second flap (3.22/rev, Figure 5.9), mode 4 is third flap (7.14/rev, Figure 5.10), mode 5 is first torsion (8.8/rev, Figure 5.11), and mode 6 is a lag-torsion mode (10.3/rev, Figure 5.12). The flap and torsion modes are uncoupled. However, there appears to be coupling between the torsion and lag modes.

Table 5.8 Blade frequencies (Hz): 1588 RPM, zero collective; Johnson beam model; single load path (SLP), dual load path (DLP), and full (including pitch control) 3-D models with medium blade mesh

Mode	Rap Test (0 RPM)	CAMRAD	SLP	DLP	Full
F	13.9	13.8	14.3	13.9	13.9
L	24.1	34.1	36.8	36.0	35.8
F	46.3	53.4	70.8	61.5	61.0
F	124.4	145.0	195.8	168.3	164.3
L	195.0	177.0	289.7	256.3	249.8
T	227.8	202.0	265.6	235.7	229.0
F	358.0	317.0	381.5	329.8	319.8

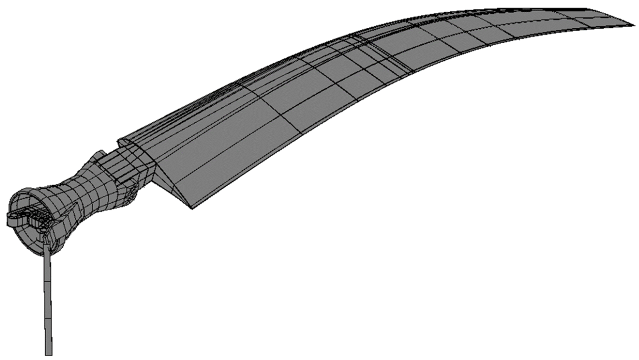


Figure 5.9 Rotating, full model, medium mesh: Mode 3, second flap (3.22/rev)

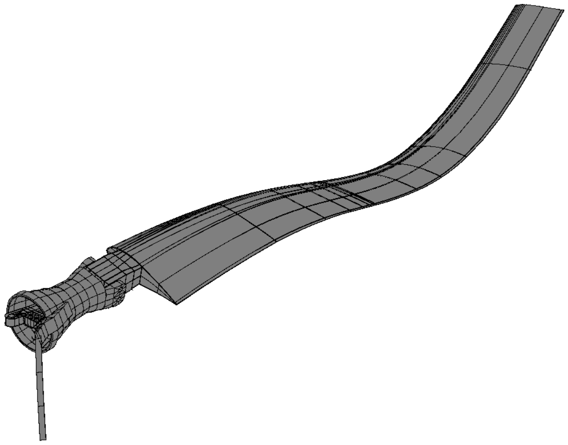


Figure 5.10 Rotating, full model, medium mesh: Mode 4, third flap (7.14/rev)

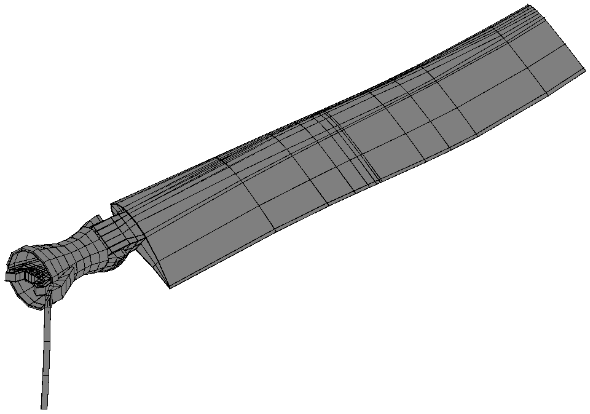


Figure 5.11 Rotating, full model, medium mesh: Mode 5, first torsion (8.8/rev)

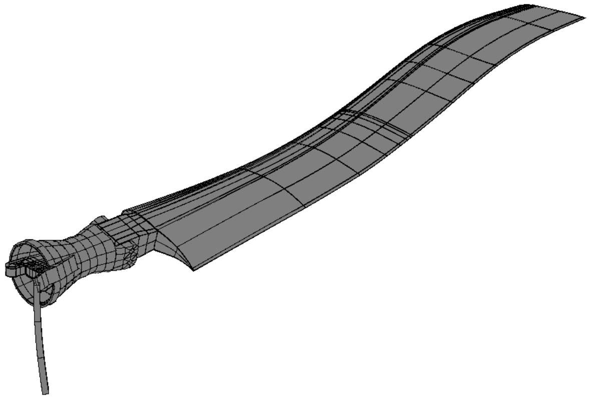


Figure 5.12 Rotating, full model, medium mesh: Mode 6, lag-torsion (10.3/rev)

The 3-D problem sizes and computational times are summarized in Table 5.9. The flexbeam supplies more degrees of freedom (DoFs) than the main blade due to its shape, large number of ply layers, and three bearing attachment points (gimbal and the two centering bearings). The dual load path between the pitch cone and the flexbeam also increases the average bandwidth of the problem (maximum 5,000). The disproportionate increase in computational time from the single load path model to the dual load path models is more due to the increase in bandwidth than the number of degrees of freedom. Convergence problems were observed for the coarse blade mesh stemming from the relatively large inboard spanwise elements. Convergence for the dual load path models were lower (up to 0.1-0.3% relative error) than the single load path model (0.001% and greater) regardless of mesh size.

Table 5.9 Summary of computational times for the X3D single load path (SLP), dual load path (DLP), and full blade models with the medium blade mesh

	X3D SLP	X3D DLP	X3D Full
Degrees of Freedom	27,780	61,136	61,541
Time, per iteration (min.)	3	13	13
No. of iterations	4-5	5-8	5-8
Time, total (min.)	12-15	60-100	60-100

5.1.3.6 Three-Bladed Model

As described in Section 4.4, a gimbal model with a single blade can be introduced by suppressing integer multiples of N_b/rev harmonics of flapping in the trim solution. The key physical explanation of this model is that a three-bladed gimbaled rotor acts in a manner similar to a hingeless (cantilevered) rotor blade in collective modes (which respond to 0/rev, 3/rev, 6/rev, … airloads) and as an articulated-in-flap rotor blade in the cyclic modes (which respond to 1/rev, 2/rev, 4/rev, 5/rev, … airloads). To illustrate the nature of these modes, the rotating modal frequencies of the three-bladed TRAM rotor model (Figure 4.22) are compared to those of the single-bladed model (Figure 4.21) with both root conditions.

Figure 5.13 shows the first seven rotating natural frequencies for the single-bladed rotor. The solid red lines show frequencies for the model when the rotor has its flap hinge at the hub center locked. For this case, the blade acts like a hingeless rotor, as reflected in its first flap frequency, which starts at a nonrotating value of 14 Hz and ends just above 1/rev at the operating speed. The dashed blue lines indicate the rotor modes when the root joint is free in flap. Note that this makes the rotor articulated in flap with zero hinge offset, so first flap is at 1/rev (actually slightly higher, due to a small amount of flap stiffness from the pitch link connection). Because the first lag mode is still locked, the first flap mode for both the fixed and flapping hub rotors are identical, with a frequency higher than 1/rev, as expected for the stiff-in-plane rotor.

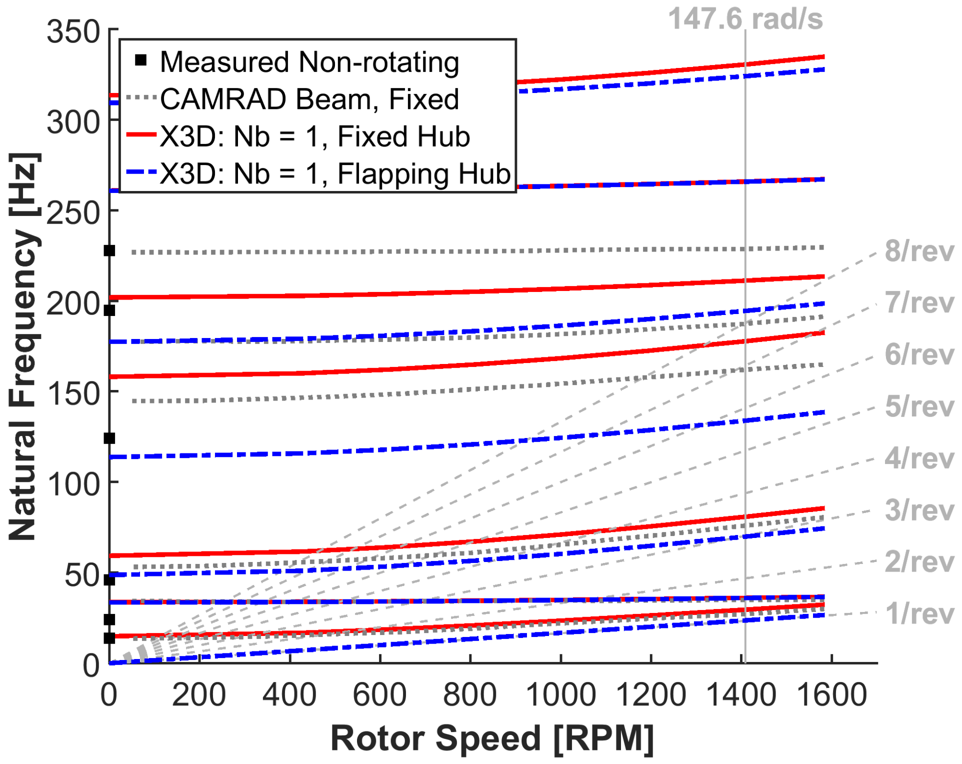


Figure 5.13 Rotating modal frequencies for the one-bladed TRAM structural analysis model, fixed and flapping hub

The frequencies for the 3-bladed rotor with the gimbal locked are shown in Figure 5.14. The frequencies for the locked gimbal case are nearly identical to those for the single-bladed model with the hub joint fixed (no flapping), though the frequencies at each mode occur three times (one for each blade), as indicated in Table 5.10. This is expected, as the locked gimbal results in three identical cantilevered blades. Small differences in frequencies occur between the 3-bladed model and the 1-bladed case, and can be attributed to the fact that the rigid boundary condition is being approximated as a very stiff joint and is not fully locked. Variations within frequency triplets for the 3-bladed fixed model may be due to slight differences in positions, orientations, or material properties after rotation due to numerical rounding.

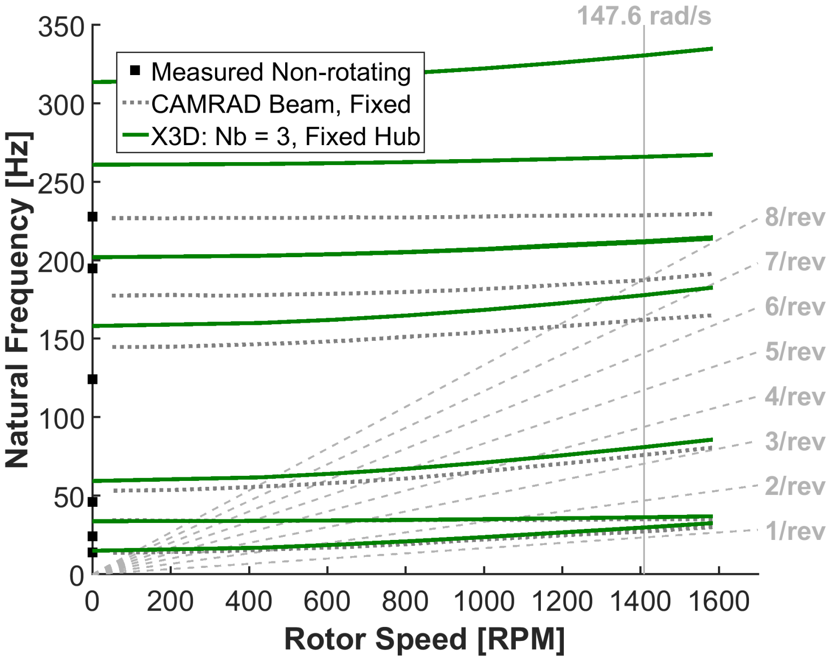


Figure 5.14 Rotating modal frequencies for the three-bladed TRAM structural analysis model, fixed hub

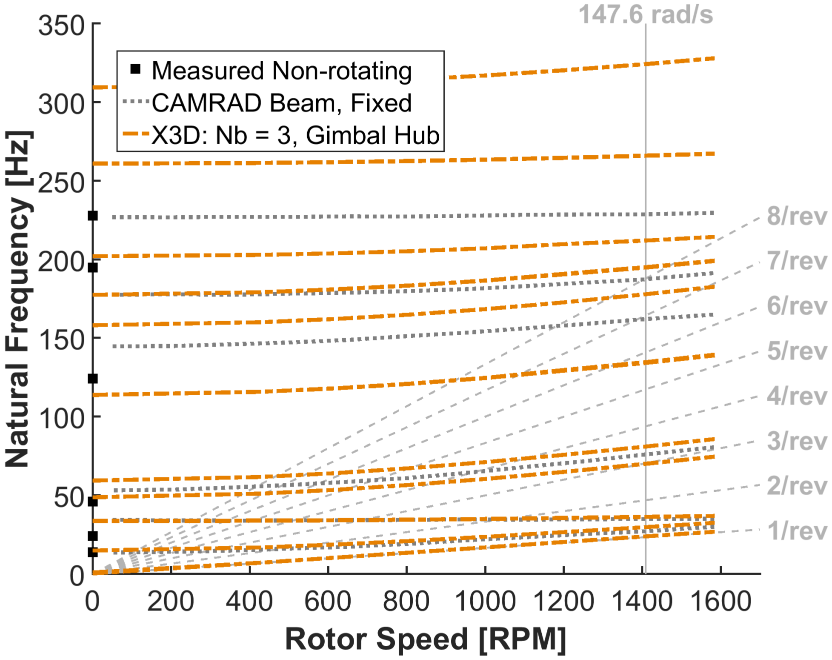


Figure 5.15 Rotating modal frequencies for the three-bladed TRAM structural analysis model, gimbaled hub

Table 5.10 Rotating (1409 RPM, TRAM test speed) natural frequencies of the one-bladed and three-bladed TRAM models, all frequencies are reported in /rev, values for 1-blade in red italic type are fixed, those in black roman type are free

1-Blade	3-Blade Fixed	3-Blade Free
1.012	-	1.021, 1.021
1.263	1.267, 1.269, 1.269	1.270
1.535, *1.537*	1.543, 1.543, 1.543	1.544, 1.544, 1.546
2.976	-	2.979, 2.983
3.441	3.445, 3.445, 3.446	3.447
5.693	-	5.704, 5.726
7.554	7.555, 7.556, 7.557	7.558
8.264	-	8.277, 8.288
8.971	8.981, 8.992, 9.012	9.004
11.29, *11.29*	11.30, 11.30, 11.30	11.29, 11.29, 11.30

Examining Table 5.10 shows that articulated flapping modes which appear in the flapping hub one-bladed model appear in pairs for the free gimbaled rotor. These are cyclic modes, including the first two modes which represent gimbal pitch (θ_y) and roll (θ_x). The modes which appear in the fixed one-bladed model appear individually in the free gimbaled three-bladed model. These are collective modes dominated by elastic flap bending, as typified by the third mode of the gimbaled rotor which represents coning of the three blades. Those which appear in both the flapping and the fixed one-bladed models are elastic lag- and pitch- dominated modes, and appear in triplets for the free gimbaled model (only the articulated flapping modes split into two gimbal modes). It should be noted that the TRAM rotor exhibits a high degree of coupling between its modes due to the large twist of the blades.

As mentioned above in Section 4.4, the gimbal is actuated with 0/rev and 3/rev signals to drive the net motion to zero, and any such actuation always requires a spring. Thus the present method of gimbal motion suppression in the time domain comes with the penalty of introducing a small stiffness that is not strictly part of the rotor system. However, as shown in Table 5.11, this slightly-stiff model has a negligible impact on natural frequencies at the operational rotation speed of the TRAM rotor (1409 RPM).

Table 5.11 Rotating (1409 RPM, TRAM test speed) natural frequencies (/rev) of the free one-bladed and the slightly-stiff one-bladed model used to represent the gimbal

1-Blade Free	1-Blade Gimbal
1.012	1.014
1.535	1.535
2.976	2.979
5.693	5.699
8.264	8.265
11.288	11.288

5.1.4 Blade Structural Properties

Discrepancies in higher mode frequency predictions between 3-D and beam-based analysis suggests a difference in the blade structural model (as opposed to lower frequencies, which are dominated my blade root modeling). The beam model used documented stiffness properties (EI, GJ) taken from TRAM blade measurements, whereas the 3-D analysis models the blade from first principles, based on the internal structure and the material properties. Comparing the two models requires calculating the sectional stiffness properties from the 3-D model.

Predictions of blade sectional properties for the medium TRAM blade mesh, obtained using the method for extracting beam-like sectional properties from 3-D models discussed in Section 2.7.1, are shown in Figure 5.16. The plots reveal a good match of flapwise bending stiffness EI_y and torsional stiffness GJ_x, but chordwise bending stiffness EI_z shows a considerable difference compared to the experiment. The chordwise bending stiffness can have a significant impact on the higher lag modes, and this error is suspected to be the cause of the differences between beam-based and 3-D modal frequency predictions (Figure 5.13). The effect of the blade sectional properties on the modal frequencies is discussed further in Section 6.1.3.

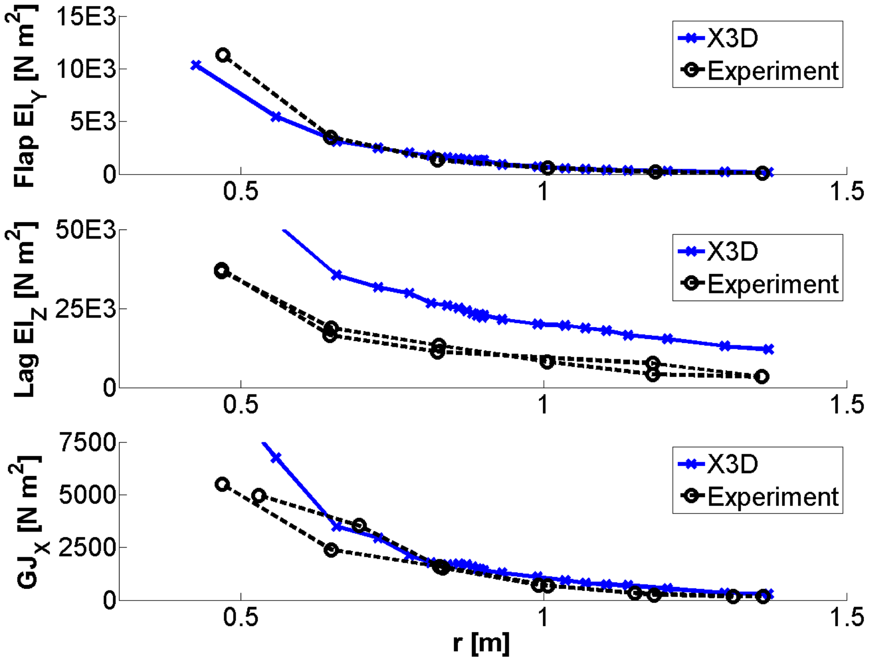


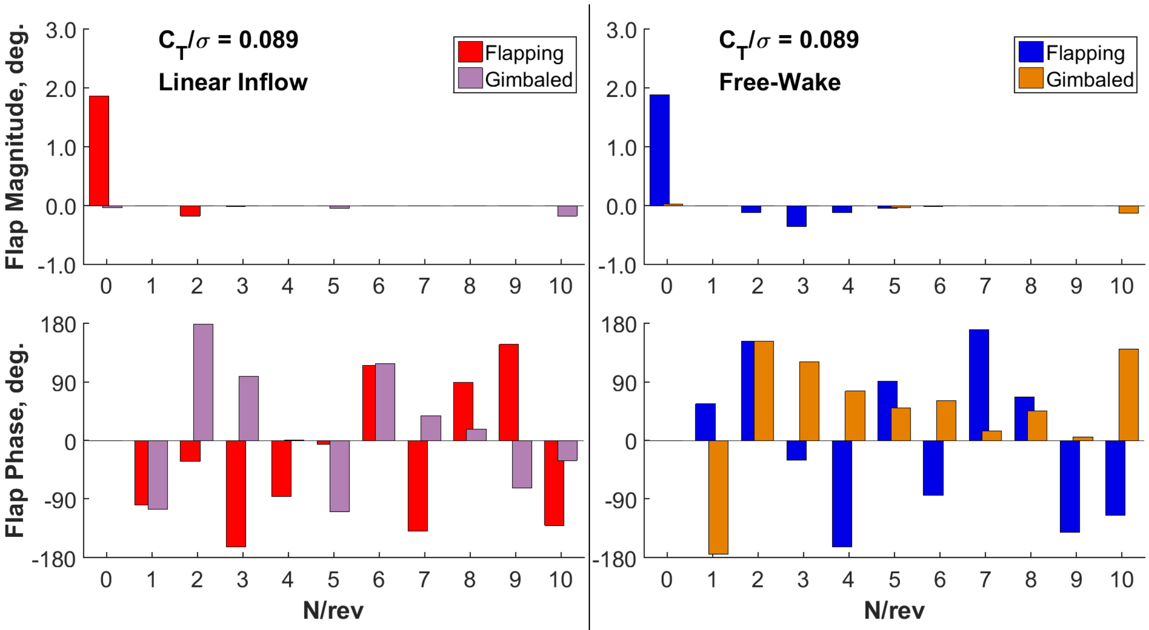
Figure 5.16 TRAM structural analysis model blade sectional properties: flap bending, chordwise (lag) bending, and torsional stiffness

The error in chordwise stiffness could be due to deficiencies in the structural model or the material properties. The 3-D CAD for the blade was built using cross-sectional manufacturer’s drawings at eight stations. At these sections, confidence in the accuracy of the structural analysis model is high. In between, there may be differences between the blade as it was actually built and the internal structure of the CAD model. The fact that the chordwise stiffness errors are no better where the available sections are closely spaced suggests that material discrepancies, and not the blade structural model itself, may be the source of the error.

5.1.5 Gimbal Flapping

The effect of changing the behavior of the root joint to simulate the gimbal can be seen by examining the flapping of that joint during edgewise flight. Figure 5.7 shows the magnitude and phase of the flap rotation angle of the root joint for the TRAM rotor at a moderate thrust condition (nominal $C_T/\sigma = 0.089$) at an advance ratio (μ) of 0.15 and a shaft angle of 2° into the wind using two different aerodynamic models, linear inflow and free-wake. Results are shown for the single blade model with both the flapping and simulated gimbal root hinges. For both cases wind tunnel trim is used, with zero 1/rev flapping targeted.

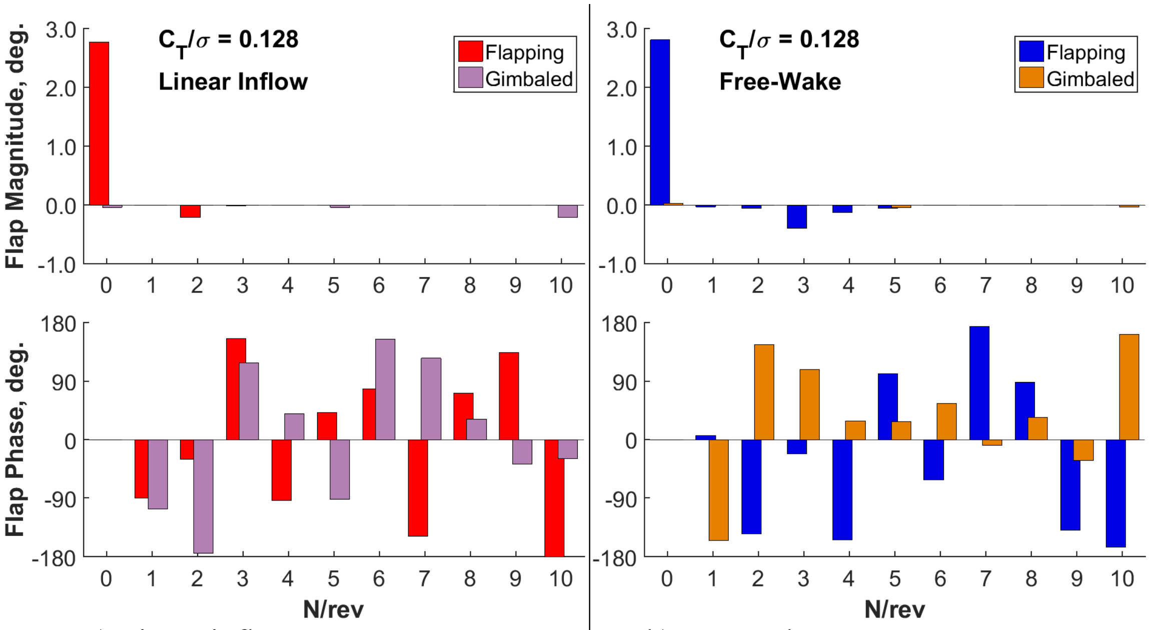
In both cases there is a significant steady coning of the blades (0/rev) when flapping is allowed; modeling the gimbal suppresses the steady flapping, causing the coning to be taken up by elastic deformation of the flexbeam. Note that there is some 0/rev flapping in the gimbaled cases; this is because flapping is suppressed by means of an extra trim variable, and the value seen represents the trim tolerance of 0.1° selected. Looking at the free-wake case, the 3/rev flapping is also eliminated by the gimbal model as intended. Further changes between the models can be seen at 2/rev for both aerodynamic models and 4/rev for the free-wake model, where the flapping is also eliminated. It is not clear if this is due to changes in the rotor aerodynamics or an unintended consequence of suppressing the 3/rev motion. The gimbaled rotor also adds slight 5/rev and 10/components to flapping. A test at similar flight conditions but higher thrust (nominal $C_T/\sigma = 0.128$) shows similar trends (Figure 5.18).



a) Linear inflow

b) Free-wake

Figure 5.17 Blade root flapping harmonics, with flapping and gimbaled root hub models, at moderate thrust



a) Linear inflow

b) Free-wake

Figure 5.18 Blade root flapping harmonics, with flapping and gimbaled root hub models, at high thrust

5.2 Hover Performance

Figure 5.19 presents ideal hover performance predictions in terms of power coefficient (C_P/σ) and figure of merit (FM) versus blade loading (C_T/σ). Results are plotted for BEMT model with and without the fairing, and experimental data for both the TRAM rotor and an earlier JVX (a 0.656-scale V-22) rotor test are provided for comparison. When the fairing is not included, the analysis greatly under-predicts power at high thrust, with an error of nearly 20% at a C_T/σ of 0.14, and slightly over-predicts power at lower thrust. Adding the fairing brings the power predictions much closer to the experimental measurements, with an under-prediction of power by approximately 6% at C_T/σ of 0.14. This smaller error can be attributed to differences in the airfoils between the TRAM as and the analysis model, as discussed in Section 4.3.1.

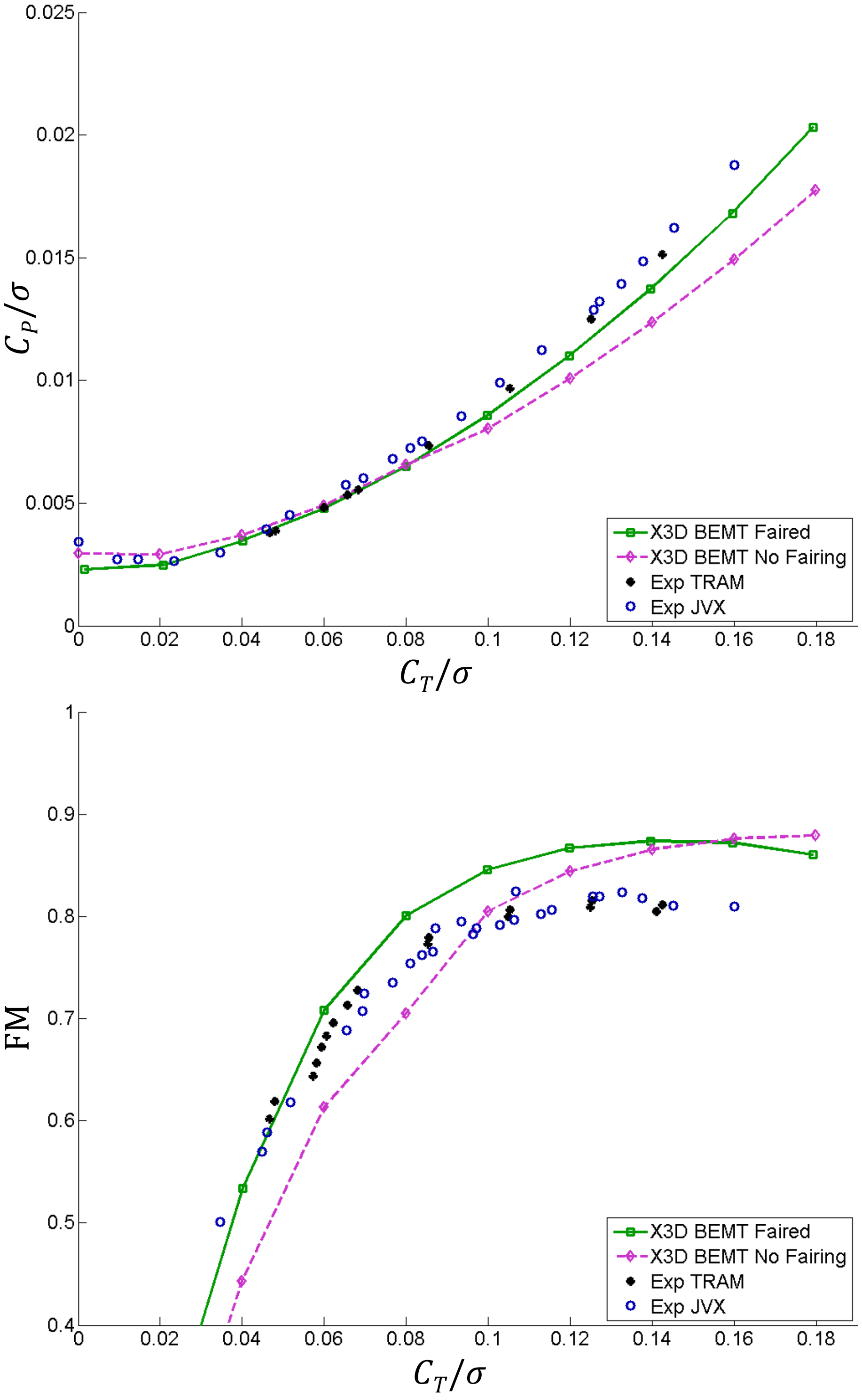


Figure 5.19 Performance (power and figure of merit) for the TRAM rotor in hover, X3D coupled aero/structures, BEMT with and without a fairing model

5.3 Edgewise Flight

The TRAM rotor was analyzed in an edgewise flight using both lower order aerodynamics models and Helios CFD. Performance and airloads predictions are compared to experimental measurements from DNW wind tunnel tests. The analysis was carried out at two thrust conditions: a high thrust condition with a 0.128 nominal blade loading (C_T/σ) and a moderate thrust condition with a 0.089 nominal blade loading. Most data were taken at an advance ratio (μ) of 0.15; the flight conditions considered are provided in Table 5.12 and Table 5.13, and were chosen to match the experimental flight conditions provided by Johnson [95]. Further prediction were made at advance ratios of 0.125, 0.175, and 0.200; these cases were all at nominal test conditions given in Table 5.14. The shaft angle values α_s in these tables are corrected for wind tunnel wall effects. Shaft angles are considered negative when tilting forward into the wind. CFD results are presented only for the nominal shaft angle of $-2°$ and at an advance ratio of 0.15 (bold in Table 5.12 and Table 5.13).

Table 5.12 Test conditions for the TRAM rotor, high thrust condition, μ = 0.15; CFD results presented for bolded test case

High Thrust Test Cases, $\mu = 0.15$						
Run	607	605	**605**	605	603	603
Point	68	252	**177**	68	13	39
T (°C)	16.61	18.13	**17.62**	16.48	14.94	15.87
ρ (kg/m3)	1.198	1.198	**1.200**	1.207	1.220	1.216
α_s (°)	-11.32	-7.34	**-3.43**	0.59	4.60	8.69
V (m/s)	32.19	32.12	**32.07**	32.32	32.19	32.13
Ω (rad/s)	147.6	147.6	**147.6**	147.6	147.8	147.8
C_T/σ	0.127	0.126	**0.124**	0.127	0.127	0.126
β_{1c} (°)	0.06	0.26	**0.09**	0.10	0.22	0.23
β_{1s} (°)	-0.03	0.01	**0.00**	0.09	0.31	0.26

Table 5.13 Test conditions for the TRAM rotor, moderate thrust condition, μ = 0.15; CFD results presented for bolded test case

High Thrust Test Cases, $\mu = 0.15$						
Run	607	605	**605**	605	603	603
Point	13	231	**122**	10	7	72
T (°C)	15.3	17.9	**17.0**	15.1	14.4	16.4
ρ (kg/m3)	1.203	1.199	**1.204**	1.213	1.223	1.214
α_s (°)	-10.92	-6.94	**-2.97**	1.04	4.98	9.02
V (m/s)	32.26	32.19	**32.25**	32.11	31.99	32.22
Ω (rad/s)	147.6	147.6	**147.6**	147.6	147.7	147.7
C_T/σ	0.088	0.088	**0.088**	0.089	0.088	0.089
β_{1c} (°)	-0.04	0.07	**0.09**	0.03	-0.14	-0.30
β_{1s} (°)	-0.08	-0.09	**0.16**	0.10	-0.13	-0.33

Table 5.14 Nominal test conditions for TRAM X3D analysis, μ = 0.125, 0.175, 0.200

Advance Ratio Sweep	
T (°C)	15.0
ρ (kg/m3)	1.225
α_s (°)	Same as μ = 0.150 sweeps in Table 5.12 and Table 5.13
V (m/s)	37.422
Ω (RPM)	147.7
C_T/σ	0.089 or 0.128
β_{1c} (°)	0.0°
β_{1s} (°)	0.0°

The wind tunnel wall correction factor applied was:

$$\Delta\alpha = \delta 0.02881 \frac{C_L/\sigma}{\mu^2}, \tag{5.1}$$

where $\Delta\alpha$ is the angle correction, δ is the wall correction constant equal to -0.147 for TRAM in the DNW wind tunnel, μ is the ratio of tunnel velocity to rotor tip speed, and C_L/σ is the normalized rotor lift coefficient ($\approx C_T/\sigma$).

5.3.1 Edgewise Flight Performance

Predictions of edgewise performance using blade element theory with linear inflow, lifting-line theory with free-wake, and RANS CFD are shown in Figure 5.20 (power) and Figure 5.21 (propulsive force), varying with shaft tilt angle. Power (Figure 5.20) is under-predicted by X3D at high thrust when using the linear inflow model for values of shaft tilt angle less than +5°. Similarly, linear inflow under predicts power at moderate thrust and low or negative shaft angles, though not as severely. Increasing the fidelity of the aerodynamics model by adding free-wake improves power predictions at high thrust, with only a slight under-prediction at negative shaft angles. At moderate thrust, however, the free-wake model over-predicts the power required at all shaft angles. Finally, the CFD model provides the best power predictions, with predictions at high thrust similar to those from free-wake, and predictions at low thrust alleviating the over-prediction problem of free-wake. Note that these free-wake results use only a single tip-vortex; a discussion of the effects of adding an inboard wake trailer is provided in Section 6.2.2.

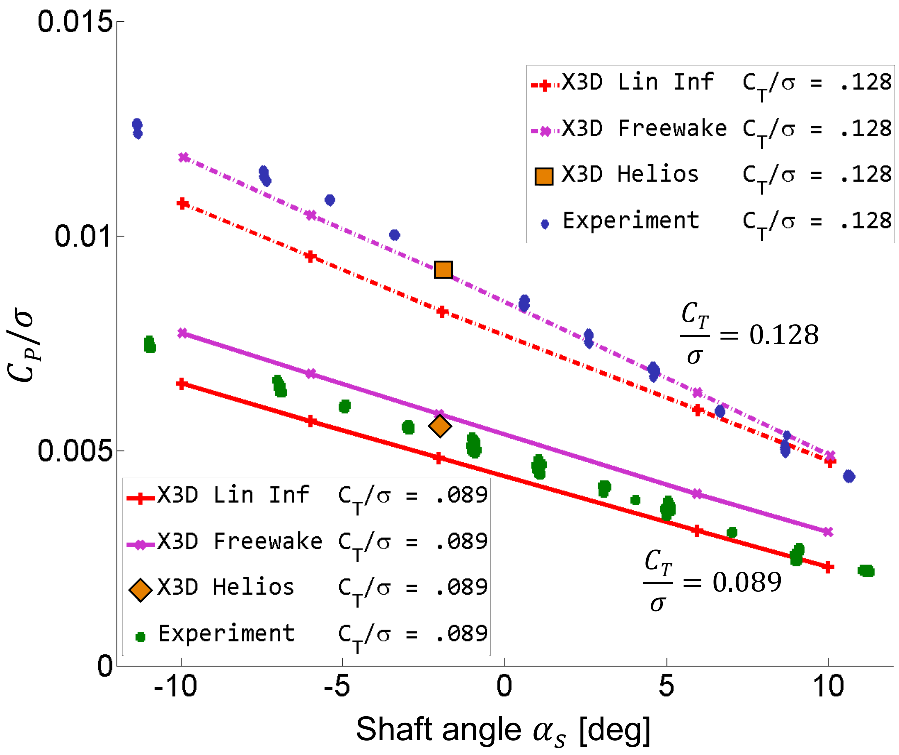


Figure 5.20 Predictions of edgewise rotor performance (power coefficient) with three different aerodynamic models at high and moderate thrust

Propulsive force (Figure 5.21) predictions show a different trend than those for power. Here, linear inflow over-predicts the propulsive force at all shaft angles for both blade loadings. The free-wake model improves the prediction of propulsive force considerably at moderate thrust, leading to a good reproduction of the trend. The results at high thrust are also improved by adding a tip vortex, but the over-prediction problem still remains. Adding CFD actually degrades the prediction of propulsive force, bringing it close to the values for the linear inflow model. This is an interesting finding, and similar to what Johnson observed earlier with lifting models – improvements in airloads (shown next in Section 5.3.2.1) are generally accompanied by degradation of propulsive force predictions.

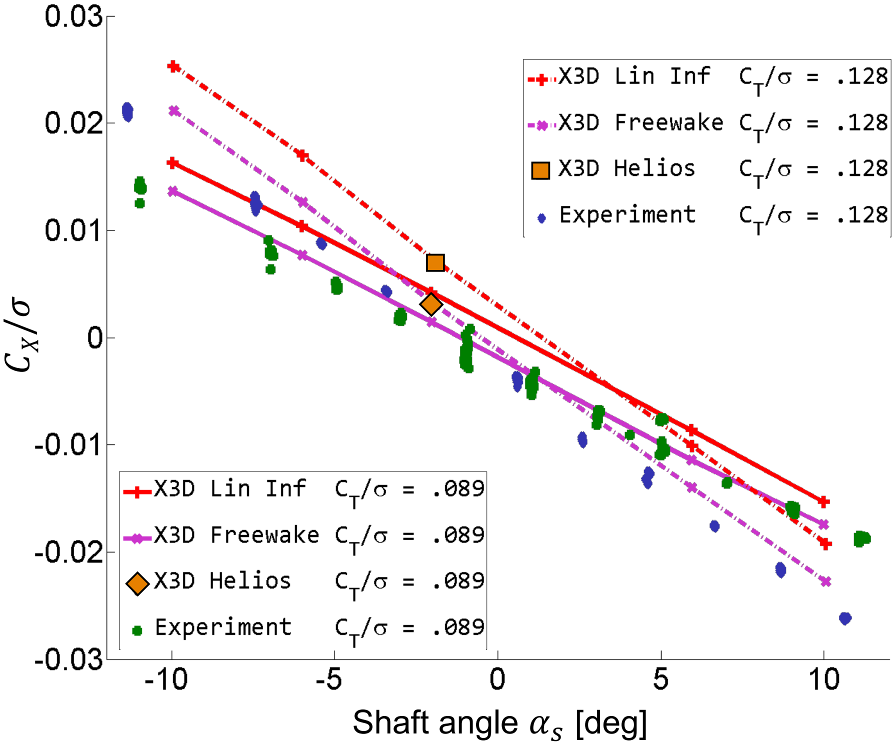


Figure 5.21 Predictions of edgewise rotor performance (propulsive force coefficient) with three different aerodynamic models at high and moderate thrust

5.3.2 Edgewise Flight Airloads

5.3.2.1 Airloads Time History

Figure 5.22 compares oscillatory sectional aerodynamic normal force predictions at the blade tip for the TRAM single blade model with a flapping hub to experimental measurements at a 0.15 advance ratio (μ). Predictions are made using three aerodynamic models: linear inflow, free-wake, and RANS CFD. Data is shown at both moderate (nominal $C_T/\sigma = 0.089$) and high thrust conditions (nominal $C_T/\sigma = 0.128$). Comparing the three models, the linear inflow model is only capable of capturing the gross 1/rev characteristics of the airloads, with lower normal force in the second quadrant and higher in the fourth. With the free-wake model, the shape of the response is better captured on the advancing and retreating sides where interactions with the rolled up tip vortices occur, but it does not match the magnitudes of these interactions. Free-wake predictions are also less satisfactory over the front of the rotor disk, with a spurious wake-induced impulse appearing at 180° azimuth that is not seen in the test data. Using the full RANS CFD coupled with 3-D CSD removes almost all of these discrepancies. It predicts the wake induced loadings in the first and fourth quadrants, both in phase and magnitude, and shows no spurious impulses. Only the first wake interaction at the high thrust condition is over-predicted.

Figure 5.23 shows similar results for the two lower order aerodynamic models, with the addition of airloads predictions for the simulated gimbal, which has 0/rev and 3/rev components of flapping trimmed out. The linear inflow model shows little difference at either thrust level, for it has no 3/rev content to begin with. The free-wake model, on the other hand, sees major changes when the gimbal motion is simulated, mainly due to its influence on 3/rev blade flapping and its feedback on airloads. In the first quadrant, magnitude predictions worsen with the gimbaled model, including a significant over-prediction of the magnitude of the negative dip caused by the first blade vortex interaction at 60° azimuth. Over the front of the rotor disk, however, the predictions are improved with the gimbaled model, with the effects of the spurious vortex interaction just before 180° (not seen in experimental data) reduced, particularly for the high thrust case. The retreating side also shows an effect from the gimbal, with airloads slightly improved, particularly at high thrust.

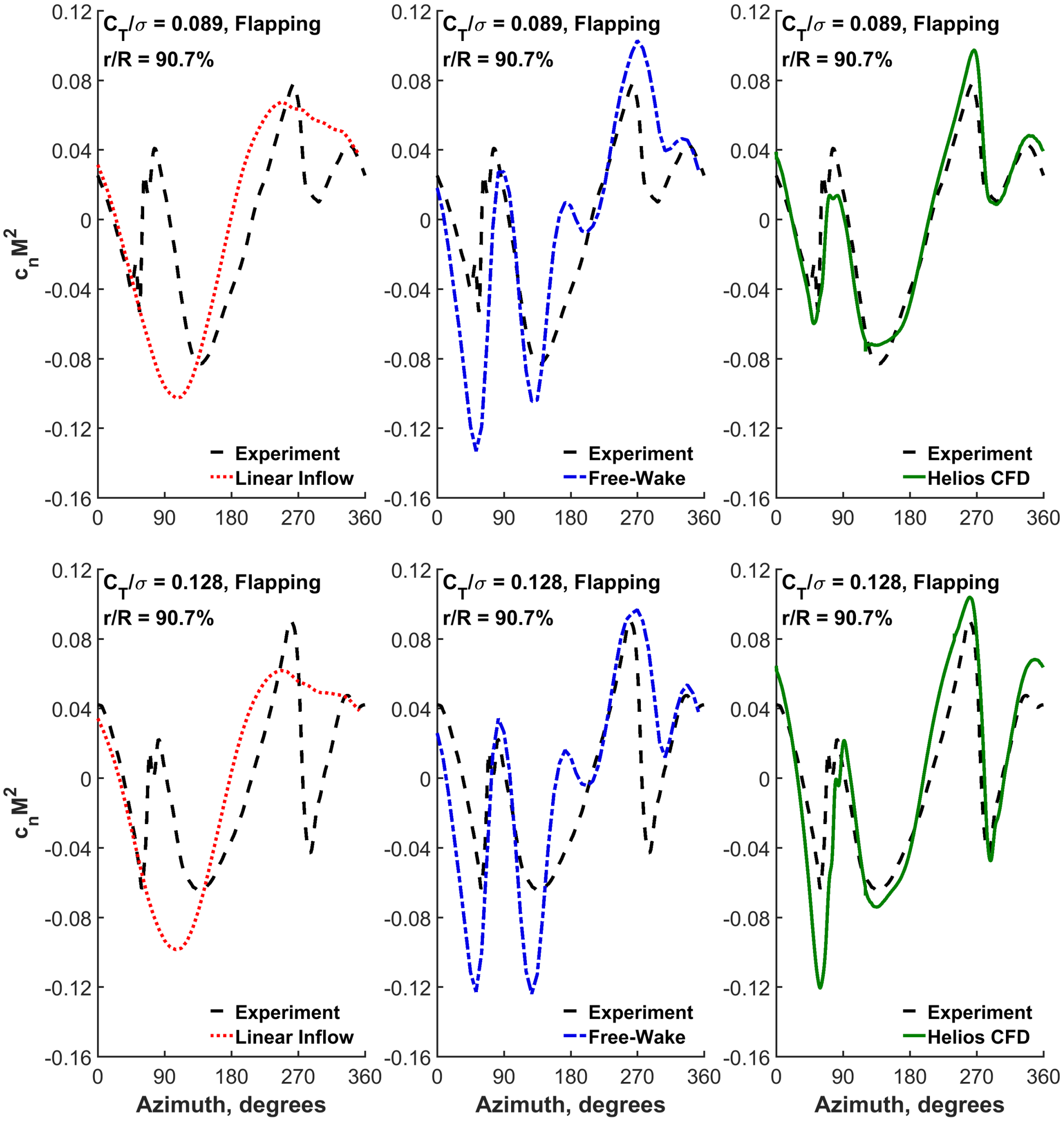


Figure 5.22 Oscillatory sectional normal force airloads, r/R = 0.907, compared to experimental data at r/R = 0.90, three aerodynamic models at two thrust conditions using the flapping hub model

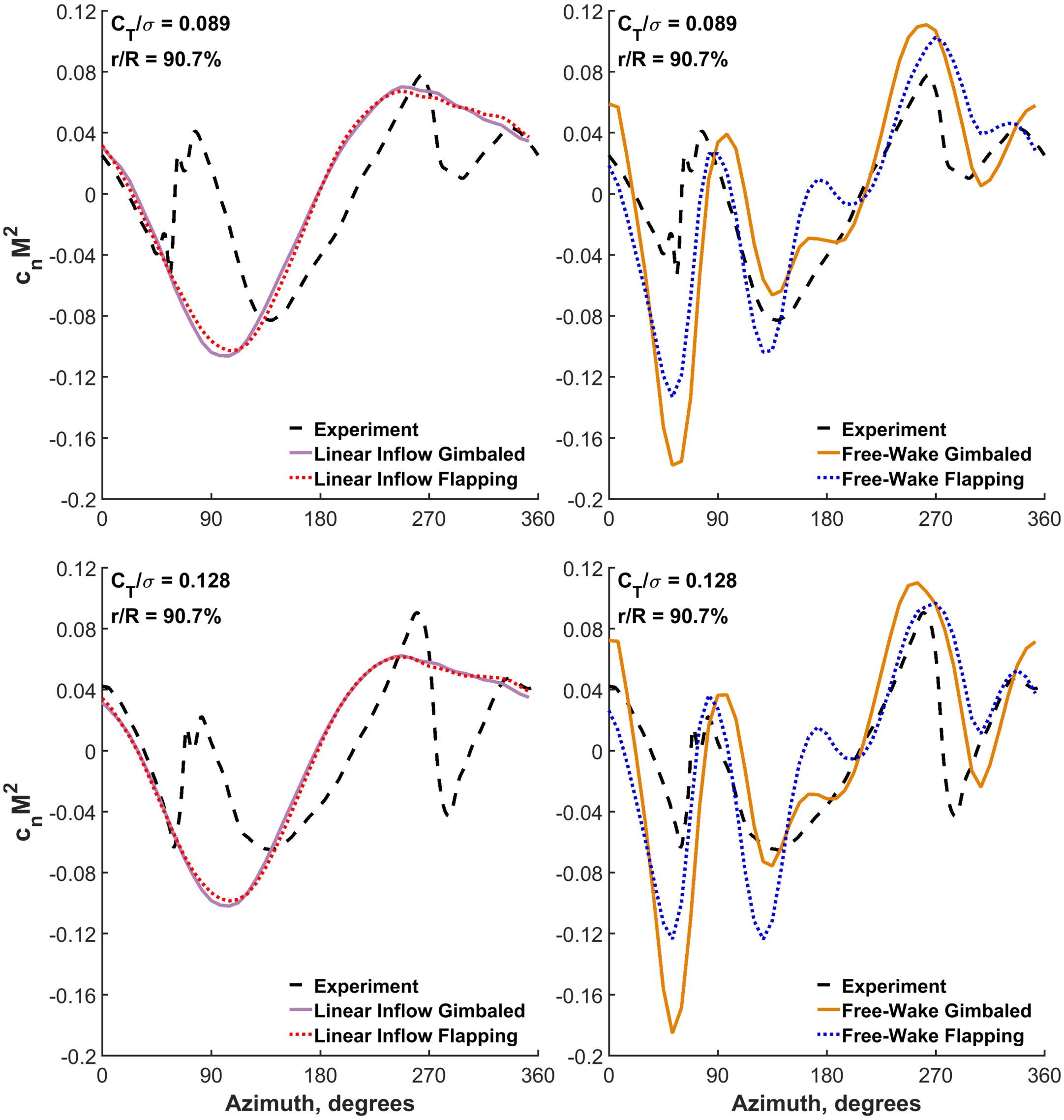


Figure 5.23 Oscillatory airloads, r/R = 0.907, compared to experiment at r/R = 0.90, flapping and gimbaled hub

Figure 5.24 similarly shows the oscillatory sectional normal force for the three aerodynamic models further inboard at 72% radius. Here once again, as expected, the linear inflow model captures none of the physics beyond the 1/rev airloads, whereas free-wake at least shows the correct mechanisms of vortex interaction – even though the magnitudes are over-predicted. The blade tip vortex roll up mechanism is particularly important at this span. This is essentially a complex viscosity driven phenomenon, beyond the capabilities of lifting line models. RANS is clearly essential to capture the vortex roll up, and hence the dynamic nature of the airloads.

Figure 5.25 shows normal force predictions using the linear inflow and free-wake models with both the flapping and gimbaled hub models at 72% radius. The linear inflow model still shows no impact from the gimbal, since it has no 3/rev loads. The free-wake model shows a significant improvement when the gimbal is simulated. The magnitude of the normal force is still over-predicted by the gimbal model at 0° azimuth, but the prediction of the interaction with the vortex on the advancing side is improved in both magnitude and phase. Neither model captures the dip in airloads over the front of the rotor disk precisely, though the gimbal model is closer in magnitude. On the retreating side of the disk, the flapping hub model captures the magnitude of the vortex interaction more accurately, but the gimbaled model shows much less phase error at the positive peak in the third quadrant.

Finally, Figure 5.26 shows the oscillatory sectional normal force around 50% radius for the flapping hub model. This far inboard, the linear inflow model is more effective as the wake interactions vanish, suggesting reduced importance of the tip vortex model, as evidenced by the dominant 1/rev variation. Trim conditions primarily influence the airloads here, and accurate predictions indicate a good trim model rather than a good wake model. The free-wake model tries but fails to capture the limited high frequency content caused by interactions with blade vortices of greater wake age. This demonstrates the difficulty in modeling the tip vortex trajectories correctly without extensive tuning of the wake parameters. The RANS results once again perform better at matching the high frequency airloads, demonstrating an accurate prediction of the wake trajectory (as opposed to the capturing the tip vortex roll-up, which dominates the airloads outboard). The gimbaled results are compared to that of the flapping blade in Figure 5.27; at this station it has little effect, which is to be expected as the gimbal influences the coning and 3/rev flap motion, both of which diminish progressively near the root.

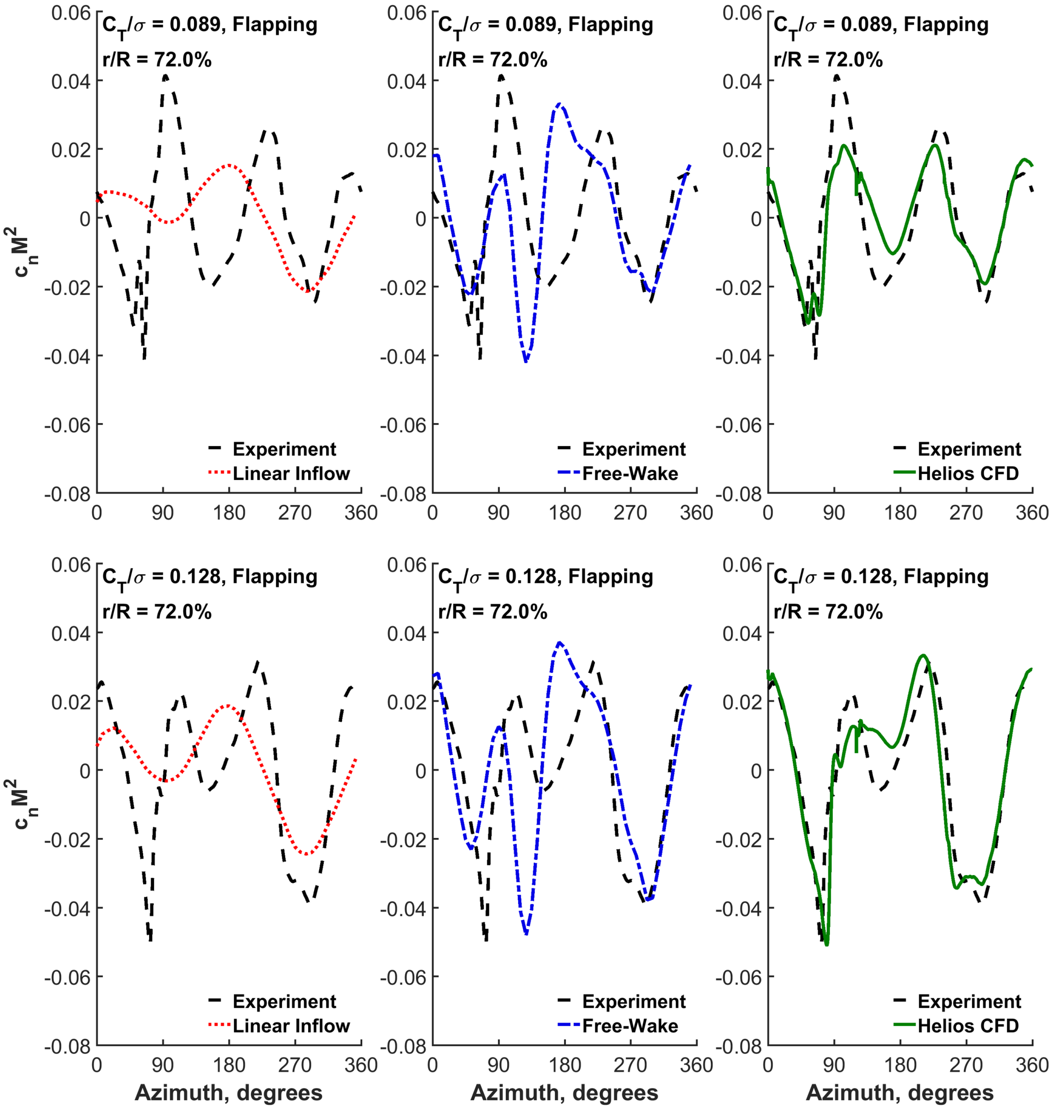


Figure 5.24 Oscillatory sectional normal force airloads, r/R = 0.720, compared to experimental data at r/R = 0.72, three aerodynamic models at two thrust conditions using the flapping hub model

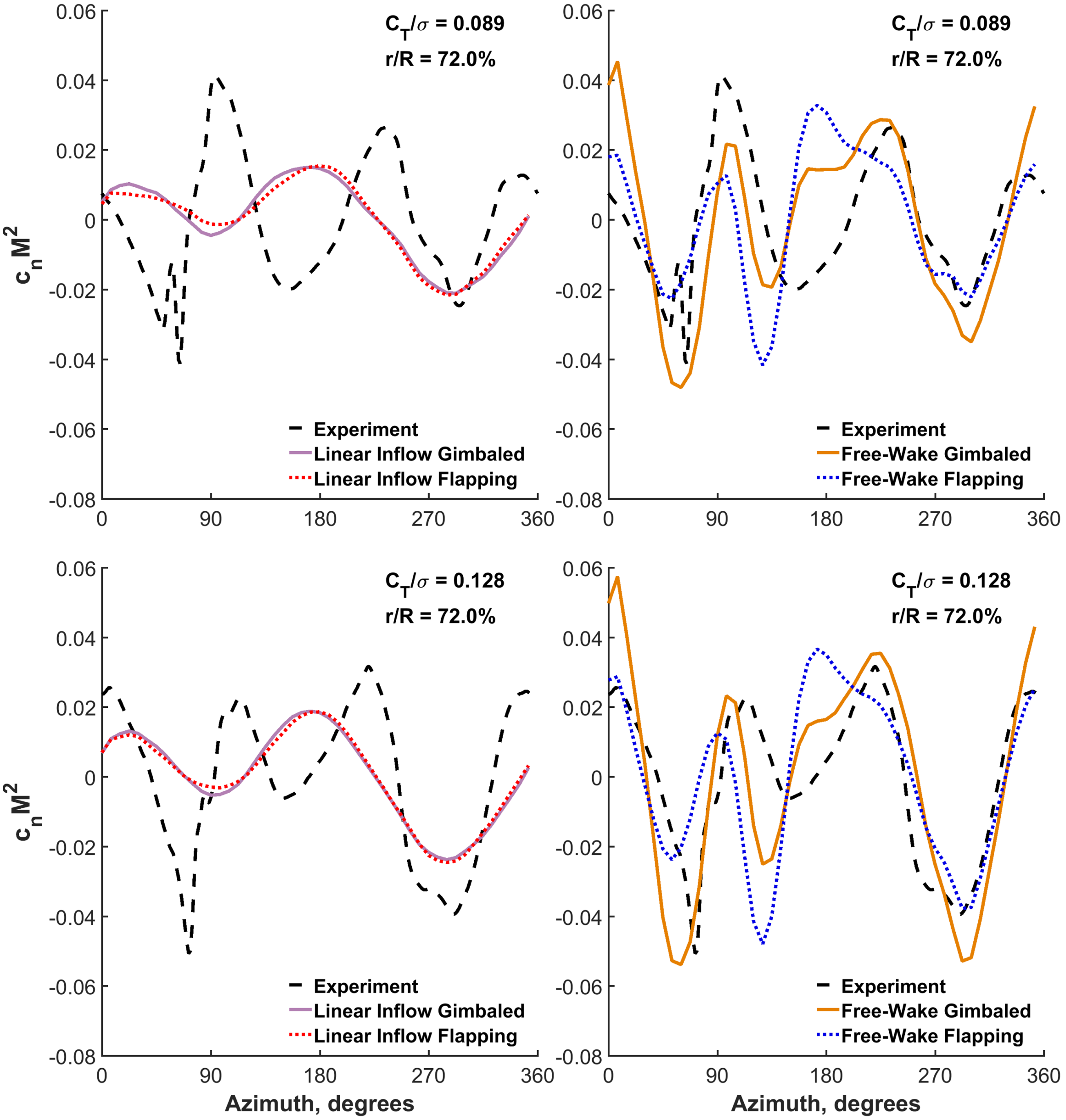


Figure 5.25 Oscillatory airloads, r/R = 0.720, compared to experiment at r/R = 0.72, flapping and gimbaled hub

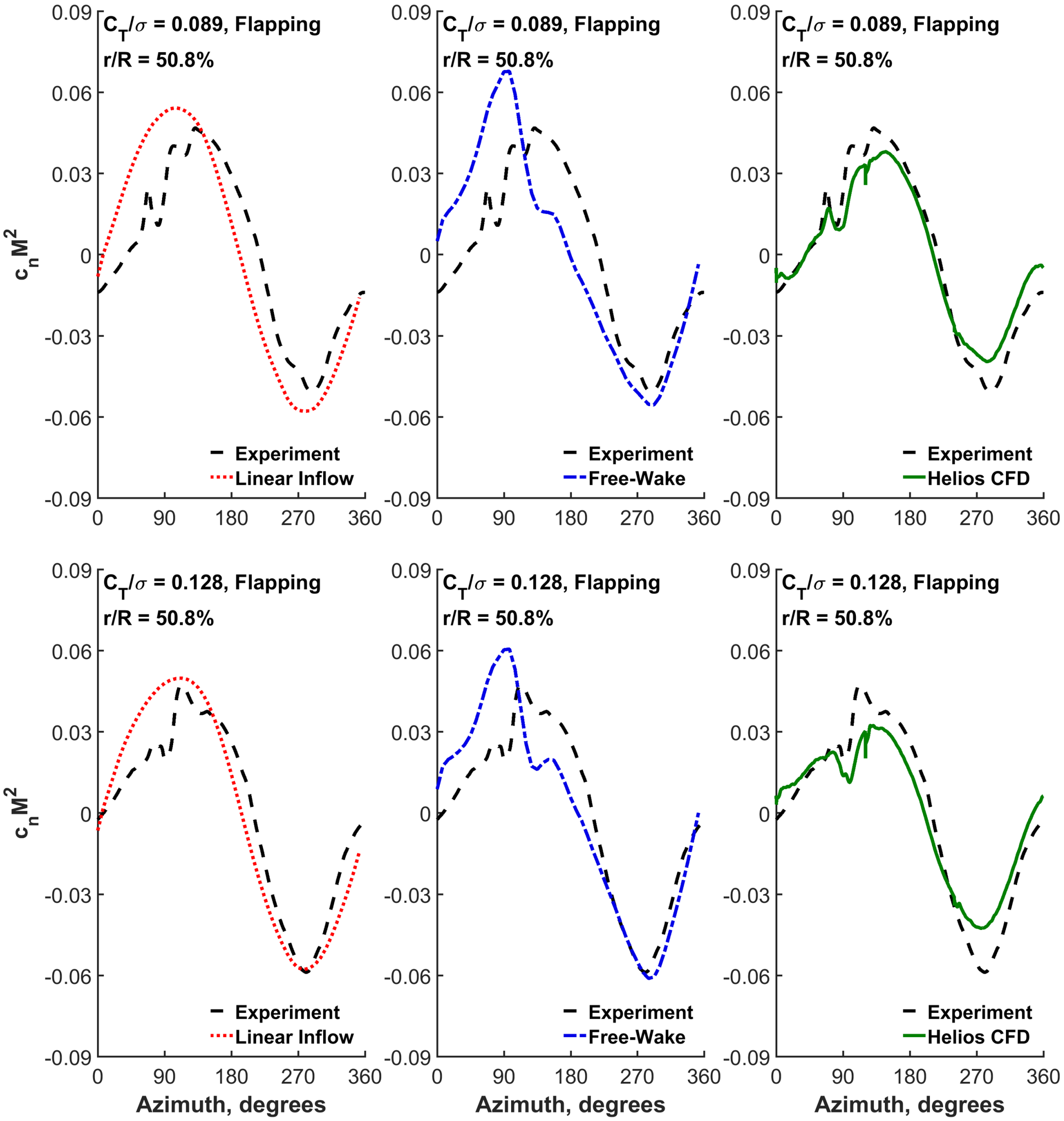


Figure 5.26 Oscillatory sectional normal force airloads, r/R = 0.508, compared to experimental data at r/R = 0.50, three aerodynamic models at two thrust conditions using the flapping hub model

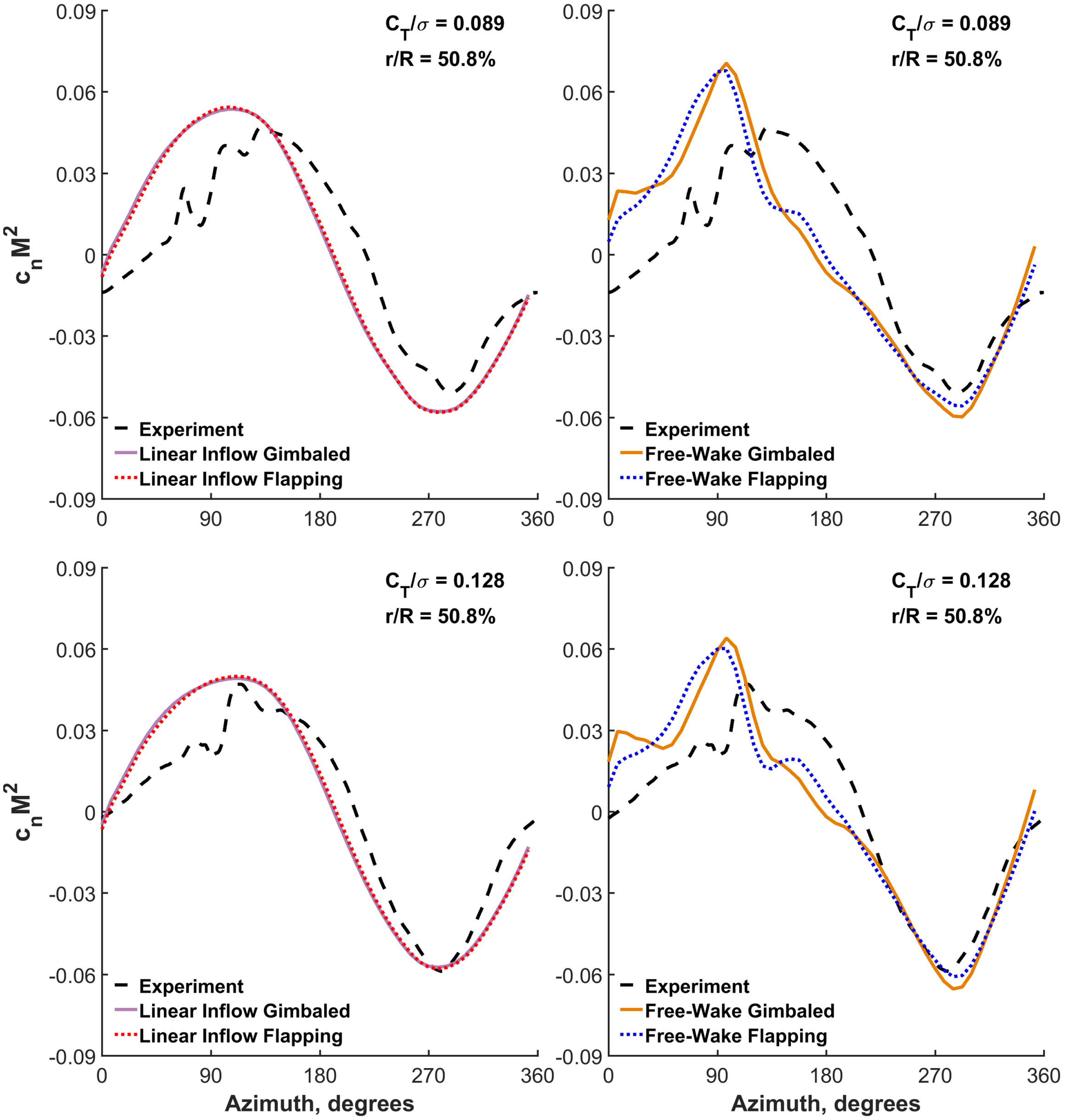


Figure 5.27 Oscillatory airloads, r/R = 0.508, compared to experiment at r/R = 0.50, flapping and gimbaled hub

Although experimental data is not available, the sectional chordwise aerodynamic force predictions for the flapping blade using all three aerodynamic models are plotted at both thrust conditions at 90% radius in Figure 5.28. The chordwise force shows a large increase of the drag over the retreating side of the rotor disk which is particularly strong for the free-wake and RANS models. The increased chordwise force in this area is expected, as a higher collective pitch is required to maintain rotor balance in the retreating region, leading to drag. The RANS model does not offer chordwise force predictions vastly different from the free-wake model, suggesting that the drag is not as dependent on accurate modeling of the wake field.

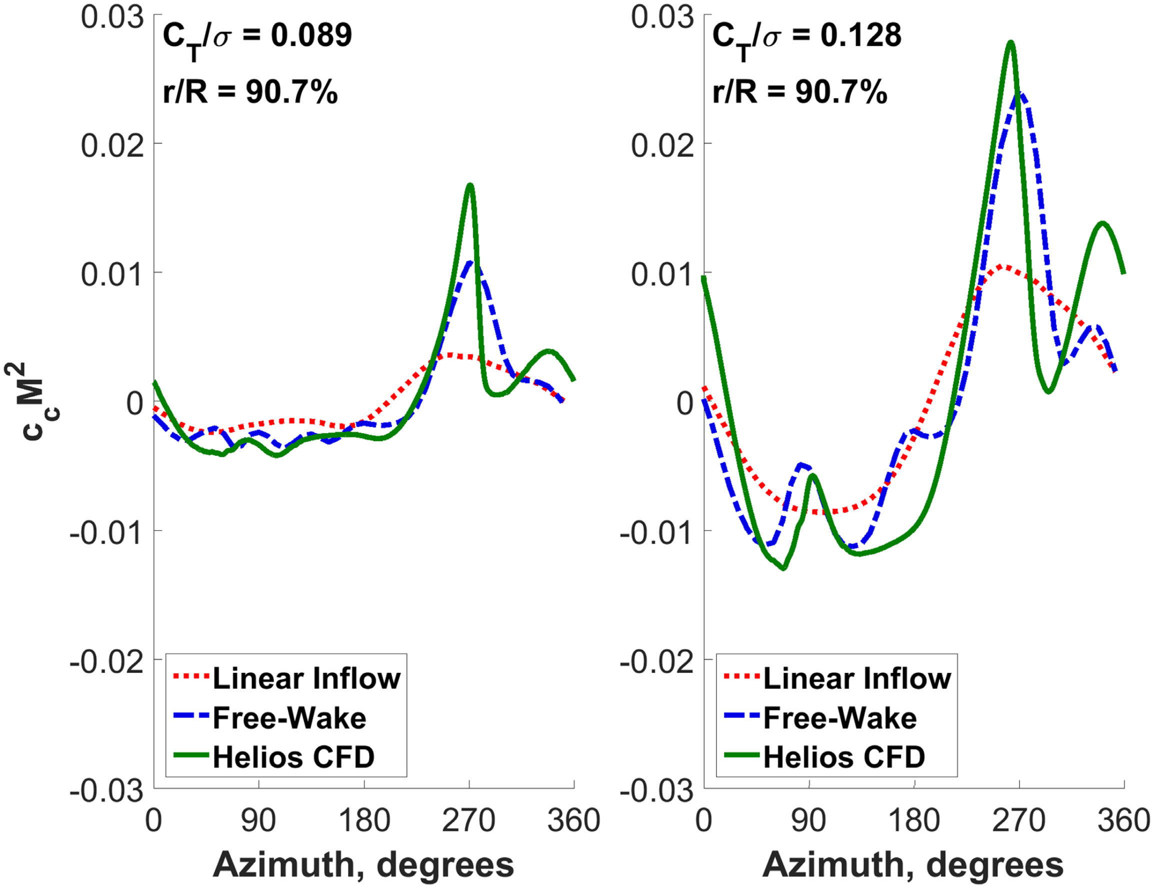


Figure 5.28 Oscillatory sectional chordwise force, $r/R = 0.907$, three aerodynamic models at two thrust conditions

More interesting is the aerodynamic pitching moment at 90% radius, shown in Figure 5.29. Unlike the chordwise force, the pitching moment shows considerable difference between free-wake and RANS, indicating the importance of the wake field o accurate prediction of the pitching moment. Particularly intriguing is a large down-up impulse in the pitching moment just after 270° azimuth, appearing only in the high thrust case. Such an impulse typically indicates the shedding of a dynamic stall vortex, a phenomenon often occurring in helicopters at high thrust and high advance ratios. This impulse, however, only occurs close to the tip of the blade, signifying a very local phenomenon. This in contrast to typical dynamic stall on helicopters, which occurs over a wider span, typically starting around 60% radius. The reason for this localized stall is found by examining the flow field predicted by CFD.

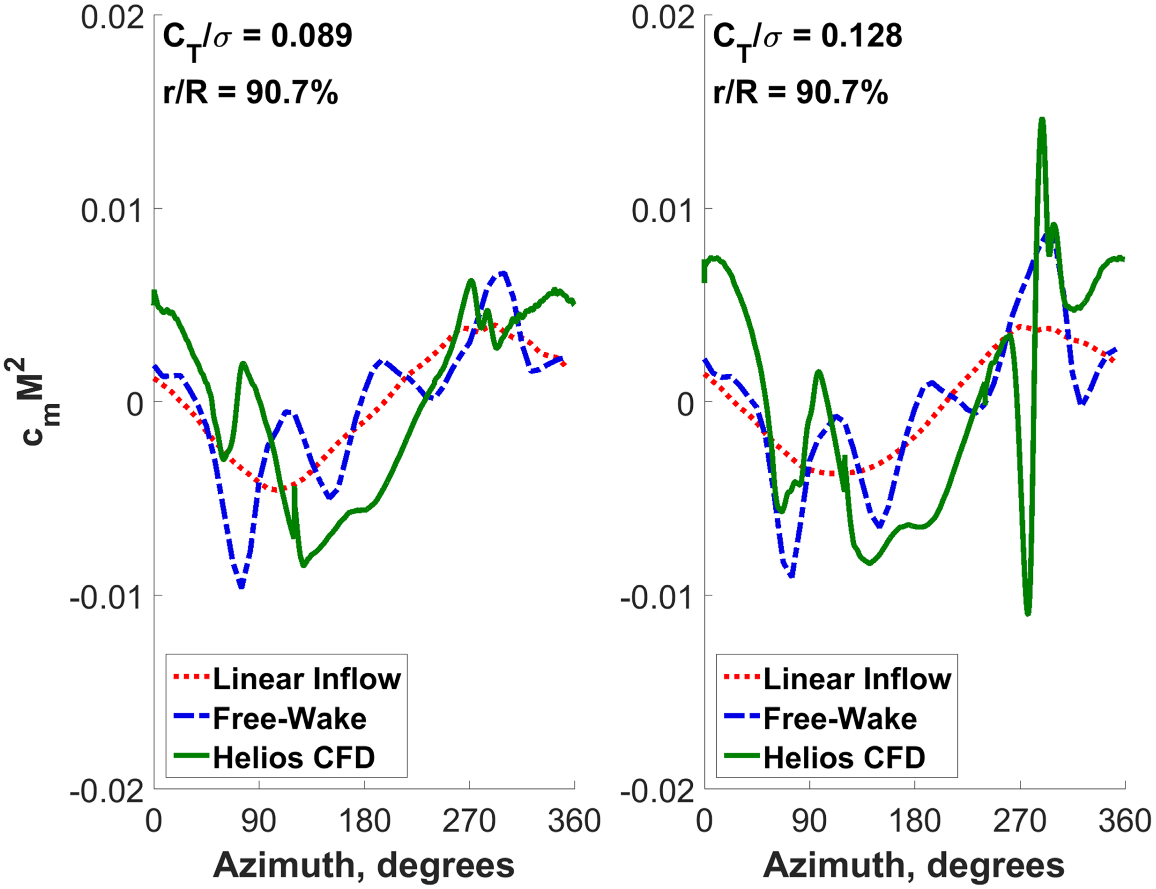


Figure 5.29 Oscillatory sectional pitching moment, $r/R = 0.907$, three aerodynamic models at two thrust conditions

Figure 5.30 shows the CFD flow field (iso-surfaces of Q-criterion, $|Q| = 10^{-5}$) of the high thrust condition as a blade passes around 270° azimuth. The shedding of a dynamic stall vortex is clearly visible when the blade interacts with the tip vortex shed by the prior blades. This suggests that local angle of attack increases due to a blade vortex interaction, inducing a dynamic stall vortex, a phenomenon noticed by Richez and Ortun during contemporary analysis of the ONERA 7A rotor at high thrust and moderate advance ratio [150]. There is no stall over the inboard portion of the rotor, despite the high thrust, as the nominal flow field is still attached. At high blade loading, such as this, a normal helicopter rotor would exhibit dynamic stall over the retreating portion of the rotor disk. The TRAM model does not experience that dynamic stall because the high RPM leads to centrifugal pumping, creating a radial flow which energizes the boundary layer and preventing separation. This behavior, known as stall delay, is inherently 3-D and viscous, and therefore beyond prediction by lifting-line models (lower order analyses instead use ad hoc airfoil $C_{l_{max}}$ corrections to account for stall delay) but critical to accurate prediction of airloads [95].

5.3.2.2 Airloads Harmonics

Greater insight is gained by examining the steady and harmonic components of the normal force airloads versus radius. The steady component for all three aerodynamic models is provided in Figure 5.31. Looking at the steady components, it can be seen that all three aerodynamics models are close to capturing the radial distribution of normal force, and the RANS model comes closest to matching the shape. There is, however, an over-prediction of the steady thrust over the outboard portion of the blade. This is caused by an error in the analysis, causing airloads over the inboard fairing to be neglected.

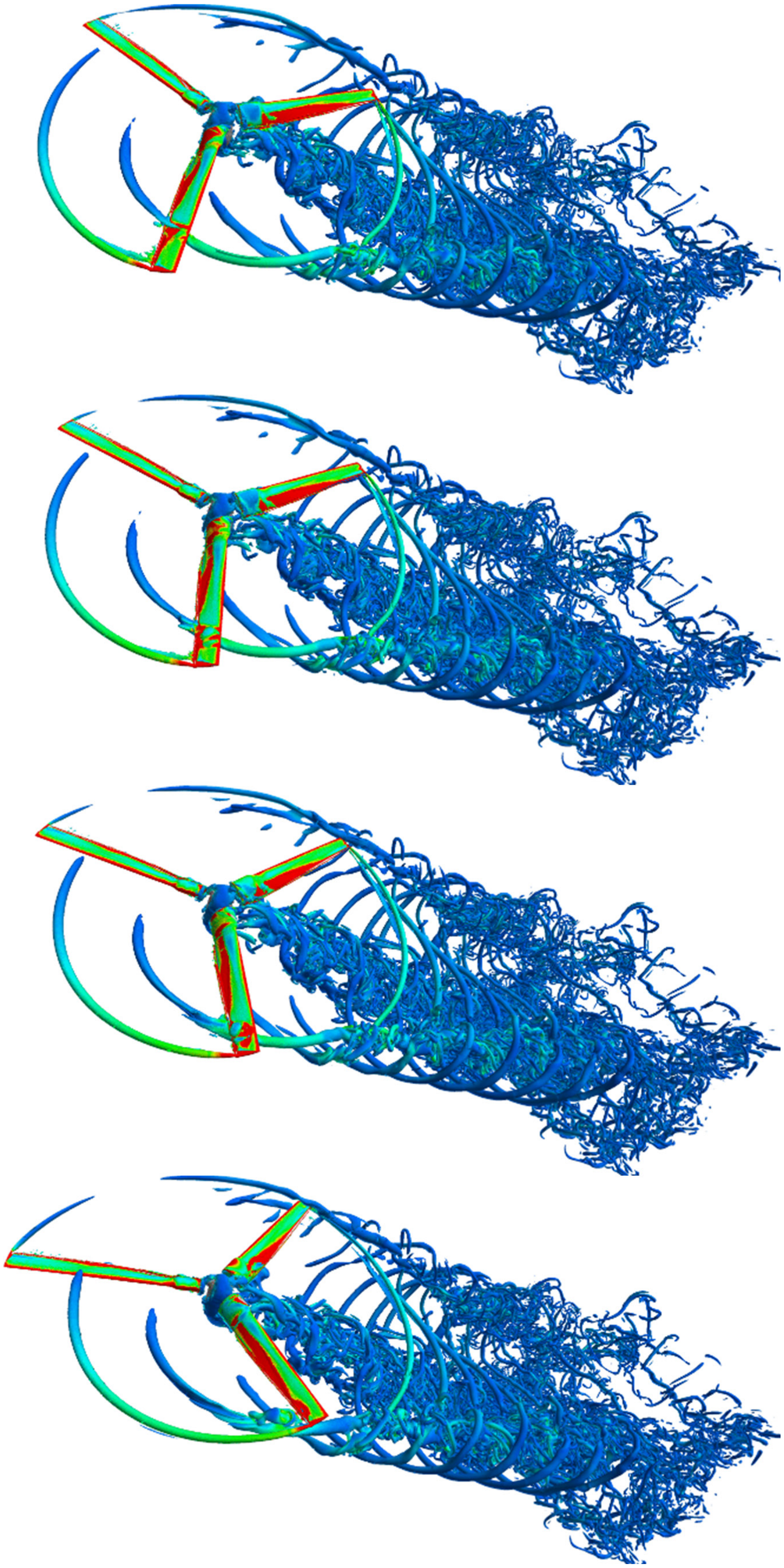


Figure 5.30 Wake vorticity at high thrust; this progression of stills shows the development of blade-vortex-interaction induced dynamic stall at the blade tip on the retreating side

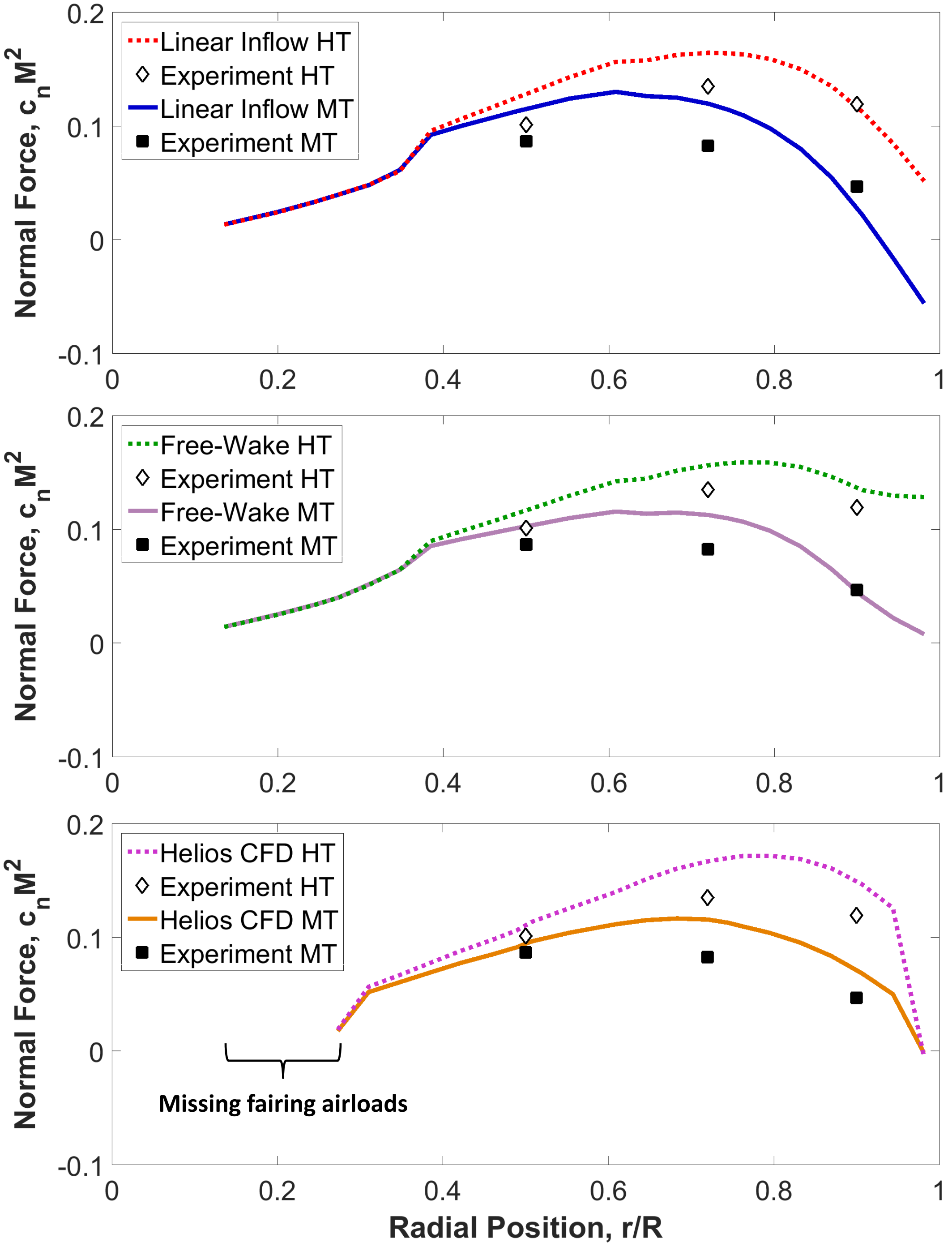


Figure 5.31 Steady component of sectional normal force, X3D compared to experimental data, three aerodynamic models at high (HT) and moderate (MT) thrust conditions

Examining the harmonic airloads for the three different levels of aerodynamic modeling with the flapping hub provides further insight into the physical sources of rotor aerodynamic loading: Figure 5.32 shows the first four sectional normal force harmonics predicted using the uniform inflow model, Figure 5.33 shows the same for the free-wake model, and Figure 5.34 presents CFD results. The linear inflow model (Figure 5.32) captures the trend in the 1/rev airloads, although the magnitudes are slightly over-predicted. The fact that the simplest model can identify the characteristics of this loading suggests that the primary contributor to 1/rev forcing is the asymmetry in flow velocity on the advancing and retreating sides due to the forward flight velocity of the aircraft, and the correct cyclic pitch input to counter it. The characteristics of the 2/rev forcing (caused by 1/rev cyclic) are fair but almost no higher frequency air loading appears.

Switching to the free-wake model (Figure 5.33) improves the magnitude prediction of the 1/rev forcing, showing the improved fidelity of the more advanced model. The wake model provides the expected content at higher frequencies. Although the radial positions of the features are imprecise, the 2/rev forcing shows the correct trend in the sectional lift, with a shape that increases with span, before dipping and then increasing again at the tip of the blade. This suggests that the tip vortices are important in inducing 2/rev loading. The tip vortices also contribute to 3/rev and 4/rev loads, and the predictions show reasonable trends. There is a discrepancy in 4/rev phase, which explains the discrepancies in azimuthal airloads shown earlier.

Finally, the CFD model (Figure 5.34) is capable of better capturing the complex wake roll up and interactions at the tip of rotor blades, as well as the inboard wake and the center body (nacelle) wake. The magnitude and phase of all four frequencies are very accurately predicted. The only major discrepancy is found at 3/rev for the high thrust cases, where the solution over-predicts sectional normal force over the outboard potion of the blade. This may suggest a difference between the trimmed flight condition predicted in the model and that of the experiment, though the exact source of this discrepancy is not clear at present.

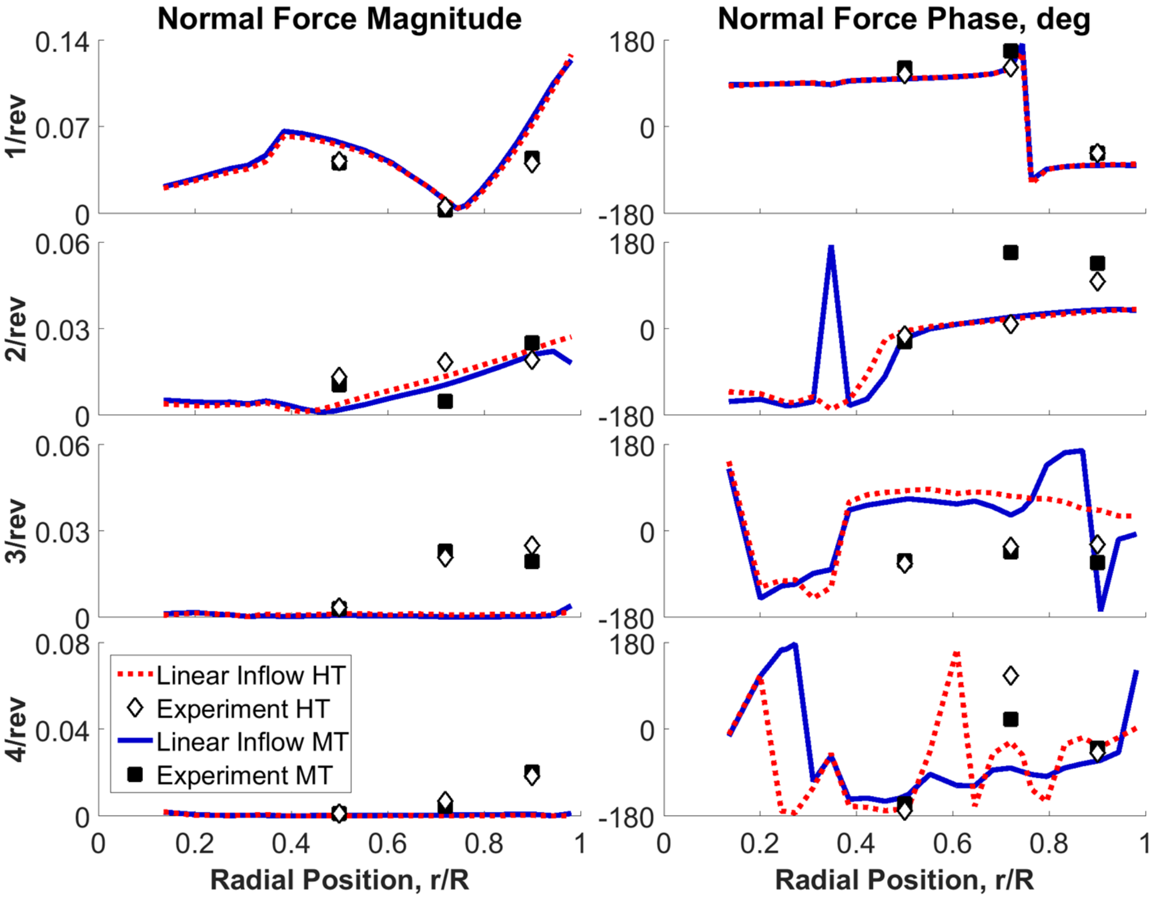


Figure 5.32 First four harmonics of sectional normal force, X3D compared to experimental data, linear inflow aerodynamic model at high (HT) and moderate (MT) thrust conditions

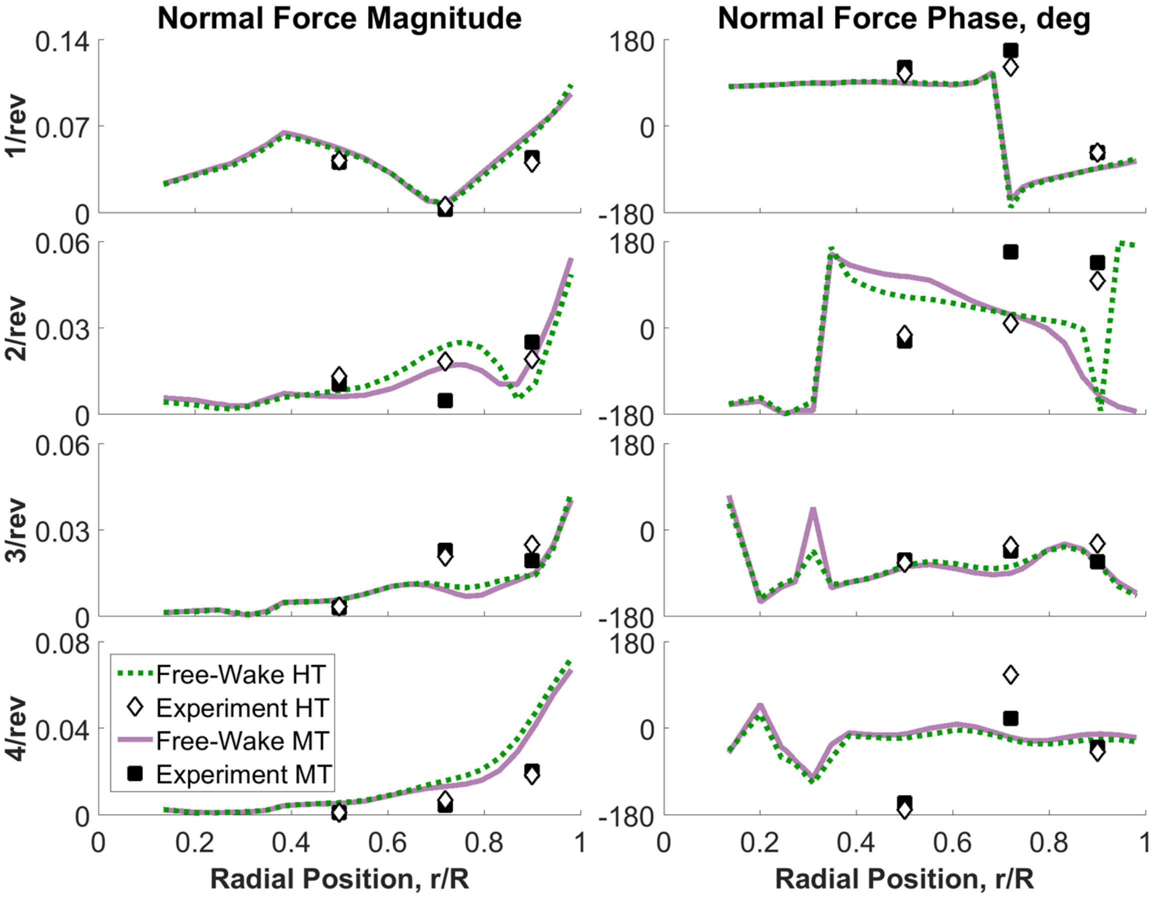


Figure 5.33 First four harmonics of sectional normal force, X3D compared to experimental data, free-wake aerodynamic model at high (HT) and moderate (MT) thrust conditions

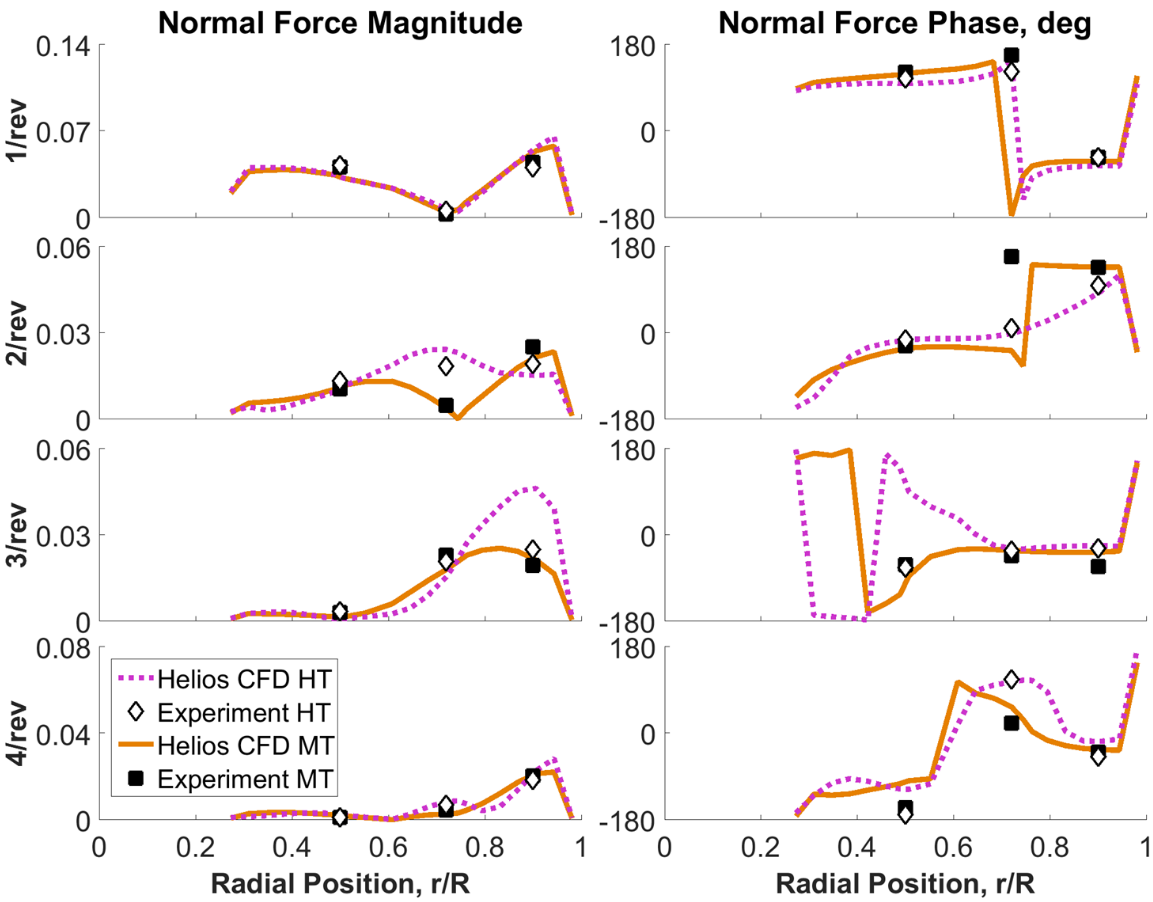


Figure 5.34 First four harmonics of sectional normal force, X3D compared to experimental data, RANS aerodynamic model at high (HT) and moderate (MT) thrust conditions

The effect of adding the gimbal model on the steady airloads is seen in Figure 5.35, which shows mean sectional normal aerodynamic force predicted by X3D using the free-wake model for both thrust levels. The differences are slight, and mostly inconsequential, although the gimbaled rotor produces more thrust over the outer portion of the blade.

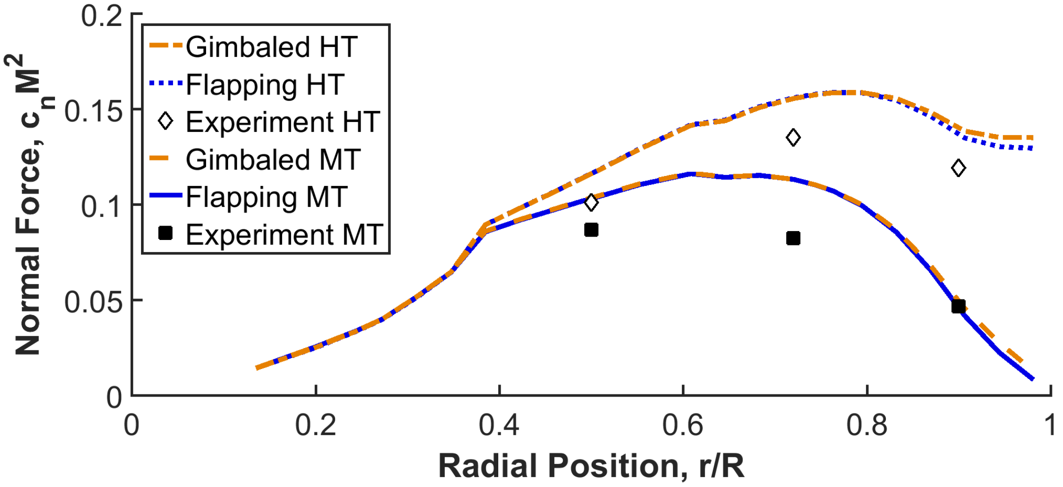


Figure 5.35 Steady sectional normal force versus radius, moderate ($C_T/\sigma = 0.089$) and high ($C_T/\sigma = 0.128$) thrust, free-wake model, flapping and gimbaled hubs

The first four harmonics of the airloads for the same cases are shown in Figure 5.36. The 1/rev airloads show no difference between the flapping and gimbaled hub rotors, or even between the two thrust cases. At 2/rev, the same trends are seen in both rotor models, though the gimbaled rotor produces slightly higher loads. Conversely, at 4/rev there is little change for the moderate thrust case but at high thrust there is a slight decrease in normal force when the gimbal is simulated. As expected, the 3/rev airloads show the most dramatic changes, with greatly increased normal force for the simulated gimbal, especially over the outboard portion of the blade. In addition to magnitude, the gimbaled rotor seems to capture the 3/rev phase slightly better over the outboard portion of the blade at high thrust, though the flapping model offers a better prediction at moderate thrust.

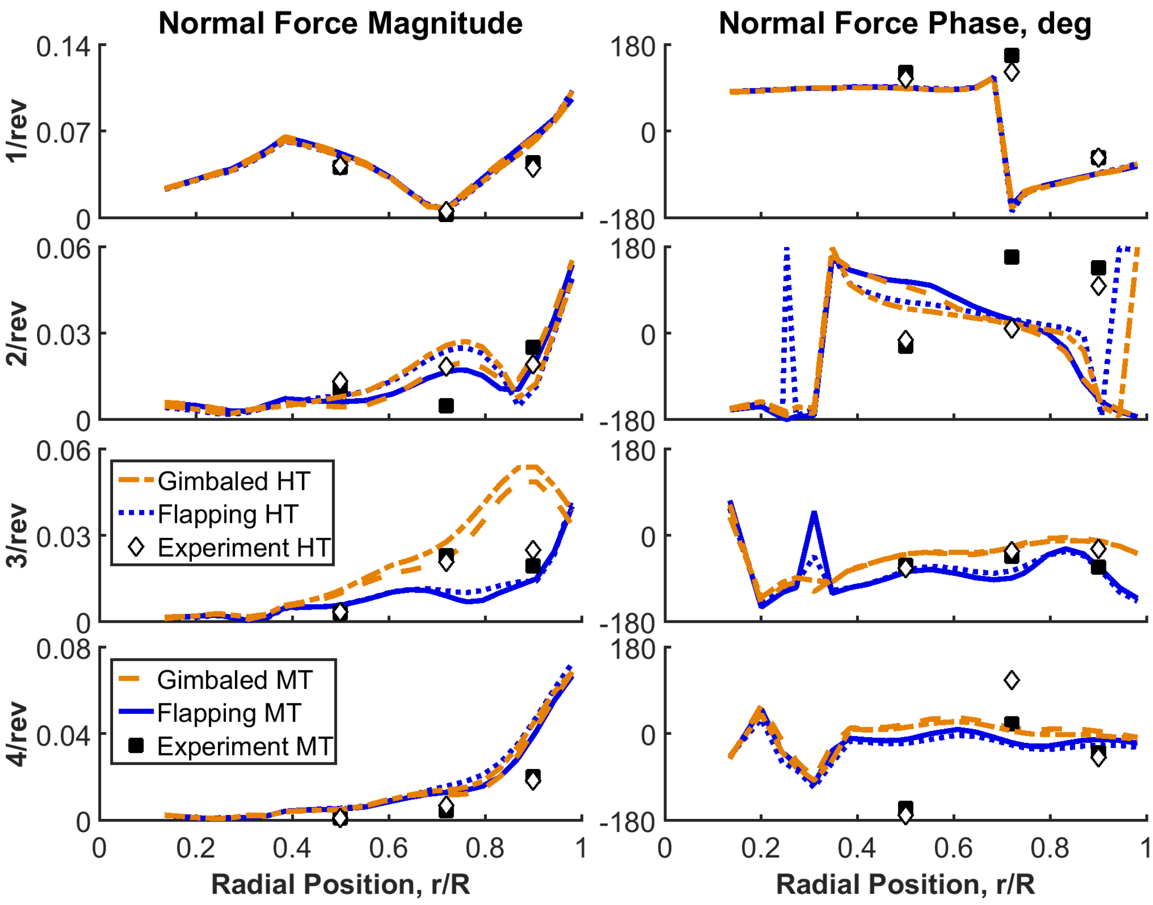


Figure 5.36 First four harmonics of sectional aerodynamic normal force versus radius, moderate (C_T/σ = 0.089) and high (C_T/σ = 0.128) thrust, free-wake model, flapping and gimbaled hubs

5.3.3 Edgewise Flight Blade Loads

5.3.3.1 Blade Loads Time History

The term blade loads refers to the internal structural bending and torsion moments carried by the blade. Although X3D provides detailed structural information in the form of stress and strain at every node in the mesh, experimental measurements evaluate blade loads in terms of sectional flap, lag, and torsion moments. Using the method for extracting sectional blade loads from 3-D strain outlined above in Section 2.7.2, the sectional blade loads are predicted for the moderate and high thrust flight conditions. Figure 5.37 presents the oscillatory flap bending moment at mid-radius using the flapping hub. At moderate thrust, the linear inflow model somewhat reflects the low frequency oscillatory trend, whereas the free-wake model shows major discrepancies, with a large 3/rev component. This suggests that the tip vortices can have a strong effect on the bending moment, and modeling the vortex interactions correctly is important to structural load predictions. The CFD/CSD model accurately predicts the low frequency trend and nearly predicts the peak-to-peak magnitude at moderate thrust, but there are differences in high frequency content. At high thrust, there is a change in the nature of the experimental data, with a large 3/rev component appearing. This is reflected in the free-wake data, which has the phase of the peaks very well captured, though the magnitude is greatly over-predicted. The CFD/CSD coupled analysis does not reflect this trend at all.

Oscillatory lag bending moment predictions with the flapping hub rotor are shown Figure 5.38. All three models capture the dominant low frequency characteristic at moderate thrust, including the peak-to-peak magnitude. The free-wake model shows some high frequency content in the second quadrant that reflects the shape of the experimental data, but phase and magnitude are not well predicted. The CFD model shows high frequency content at all azimuthal locations, though the magnitude is lower than the test data. Like the flap bending moment, at high thrust a strong 3/rev signal appears in the test data. The free-wake model does appear to show some impulse over the front of the rotor disk, but the magnitude and phase are incorrectly predicted. The other two models fail to capture this impulse.

The torsional blade loads (Figure 5.39) are characterized by significant high frequency content. Both the free-wake and CFD models reflect this trend, but do not accurately capture the shape of the experimental signal. The CFD improves the peak-to-peak magnitude of the torsion moment compared to free-wake, although a large peak in the data is still missed in the fourth quadrant. It is unlikely for structural loads to exhibit such high frequency impulsive variations, so the tests data here is suspect.

To examine the effect of the gimbal, Figure 5.40 presents the flap bending moment at both thrust conditions using the flapping and gimbaled hub rotor models. As with the airloads, there is little difference when the gimbal is added to the linear inflow model. The free-wake model, on the other hand, shows a reduction in the 3/rev flap bending moment at low and high thrust. For the high thrust case, this brings predictions closer to the experimental measurements in the second quadrant and on the retreating side; however, it degrades the prediction around 0° azimuth. The lag bending moments (Figure 5.41) show less variation around the disk when the gimbal is modeled, although at high thrust it seems to degrade predictions around 0° azimuth. Torsion moments (Figure 5.42) are also affected by the gimbal model, but it is difficult to draw clear conclusions due to the low magnitude and noisy nature of the test data.

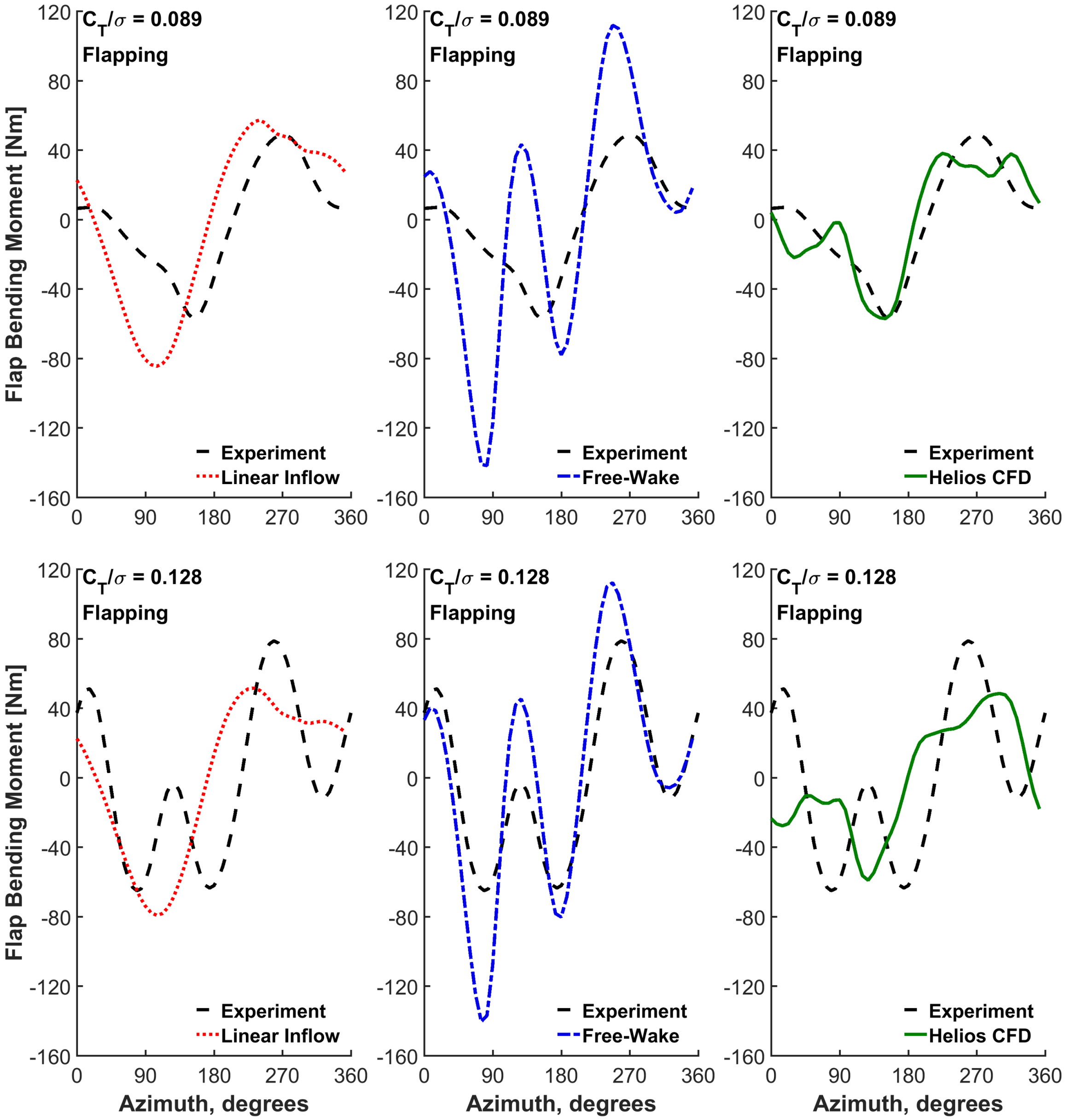


Figure 5.37 Oscillatory sectional flap bending moment, X3D predictions at $r/R = 0.48$, compared to experiment at $r/R = 0.50$, three aerodynamic models at two thrust conditions, flapping hub

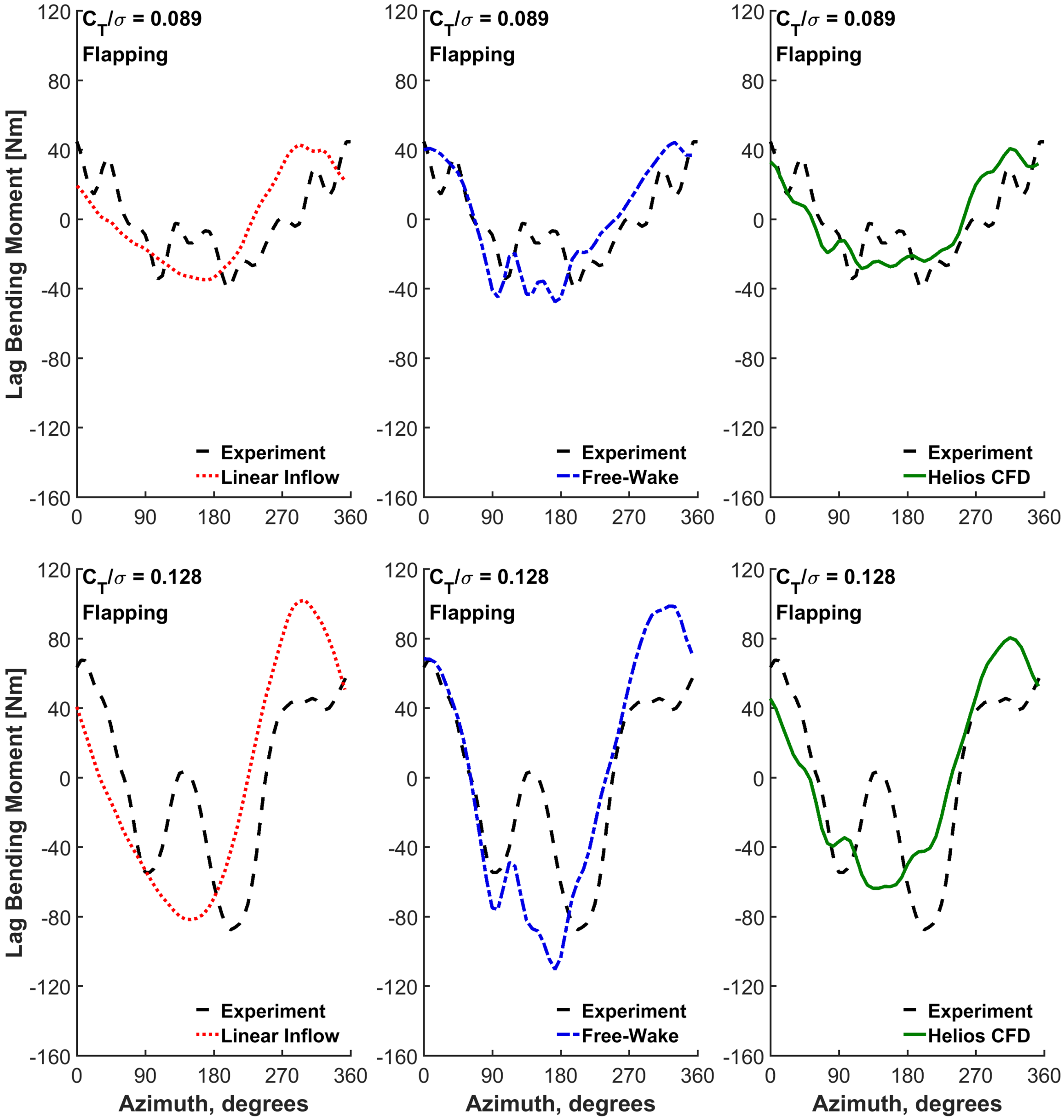


Figure 5.38 Oscillatory sectional lag bending moment, X3D predictions at $r/R = 0.48$, compared to experiment at $r/R = 0.50$, three aerodynamic models at two thrust conditions, flapping hub

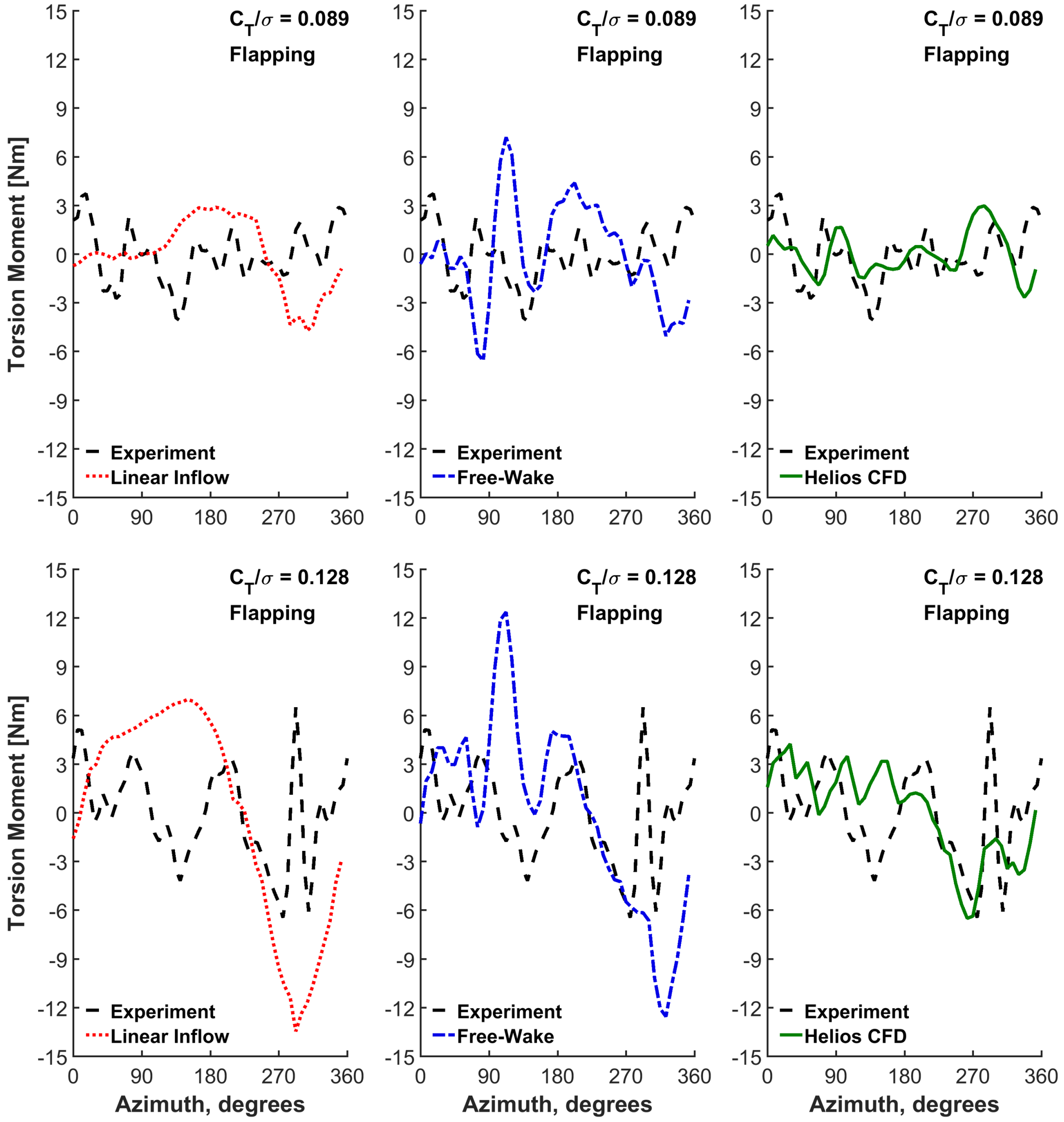


Figure 5.39 Oscillatory sectional torsion moment, X3D predictions at $r/R = 0.43$, compared to experiment at $r/R = 0.43$, three aerodynamic models at two thrust conditions, flapping hub

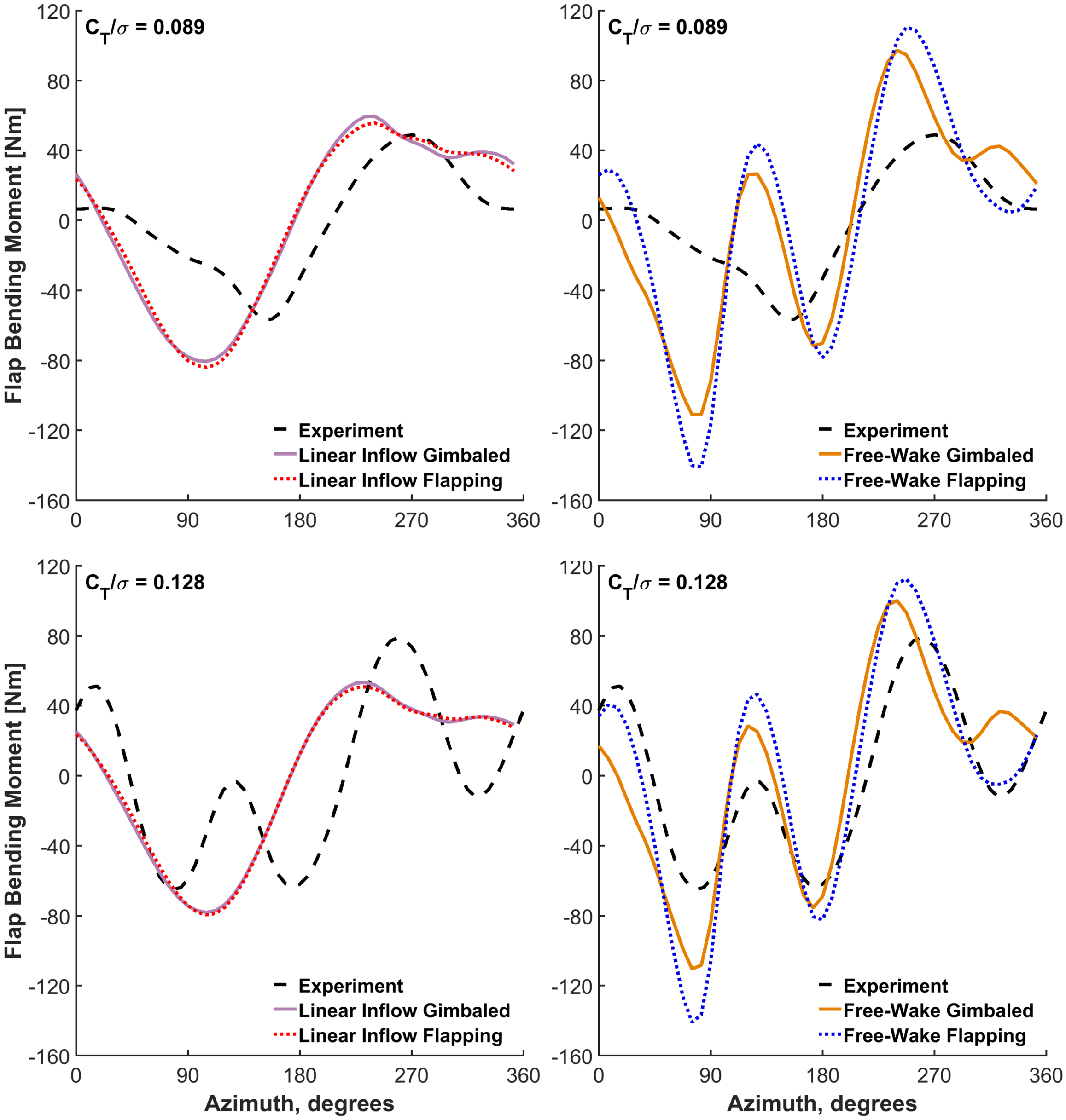


Figure 5.40 Oscillatory sectional flap bending moment, X3D predictions at $r/R = 0.48$, compared to experiment at $r/R = 0.50$, two aerodynamic models at two thrust conditions, flapping and gimbaled hub

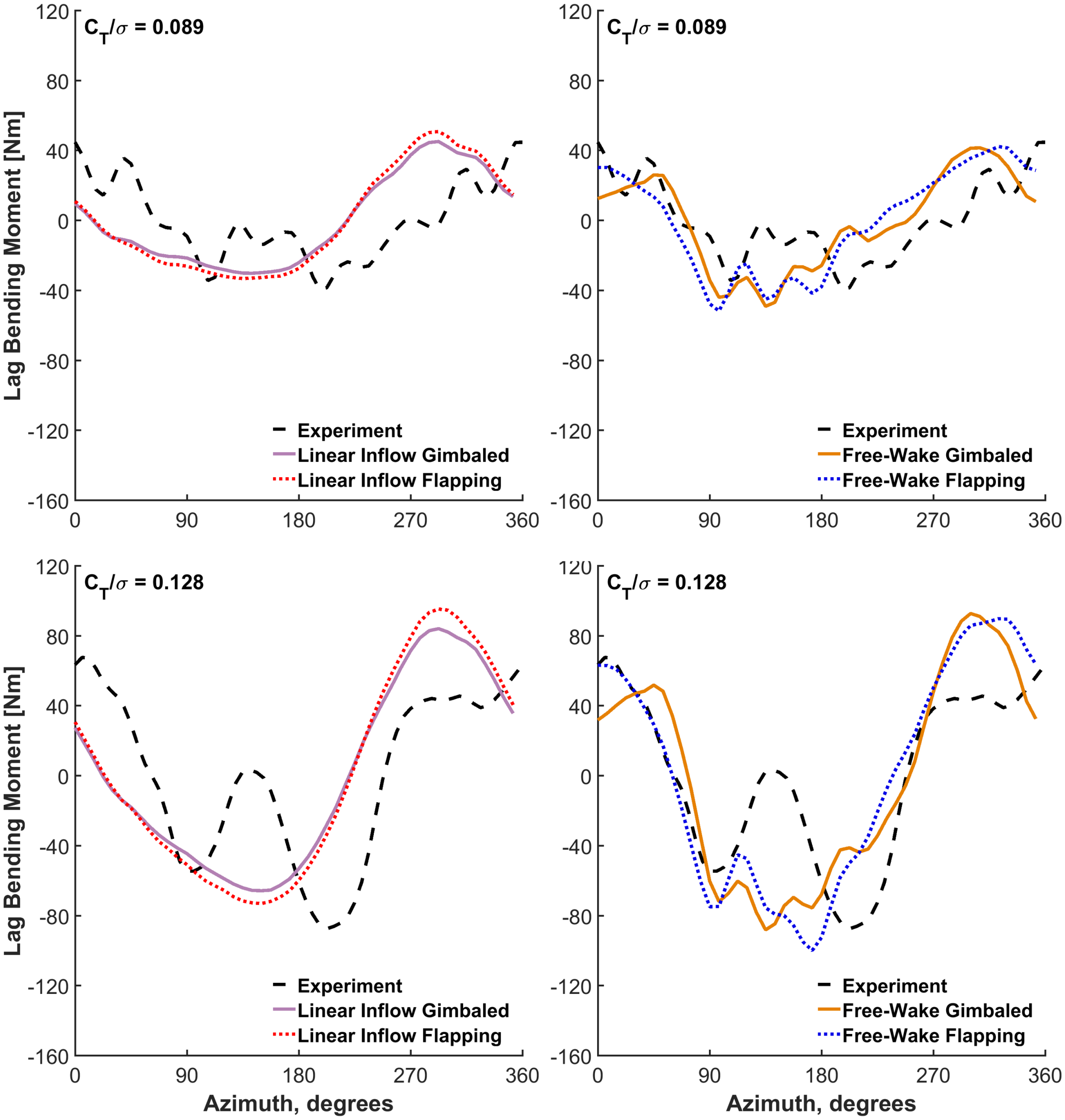


Figure 5.41 Oscillatory sectional lag bending moment, X3D predictions at $r/R = 0.43$, compared to experiment at $r/R = 0.43$, two aerodynamic models at two thrust conditions, flapping and gimbaled hub

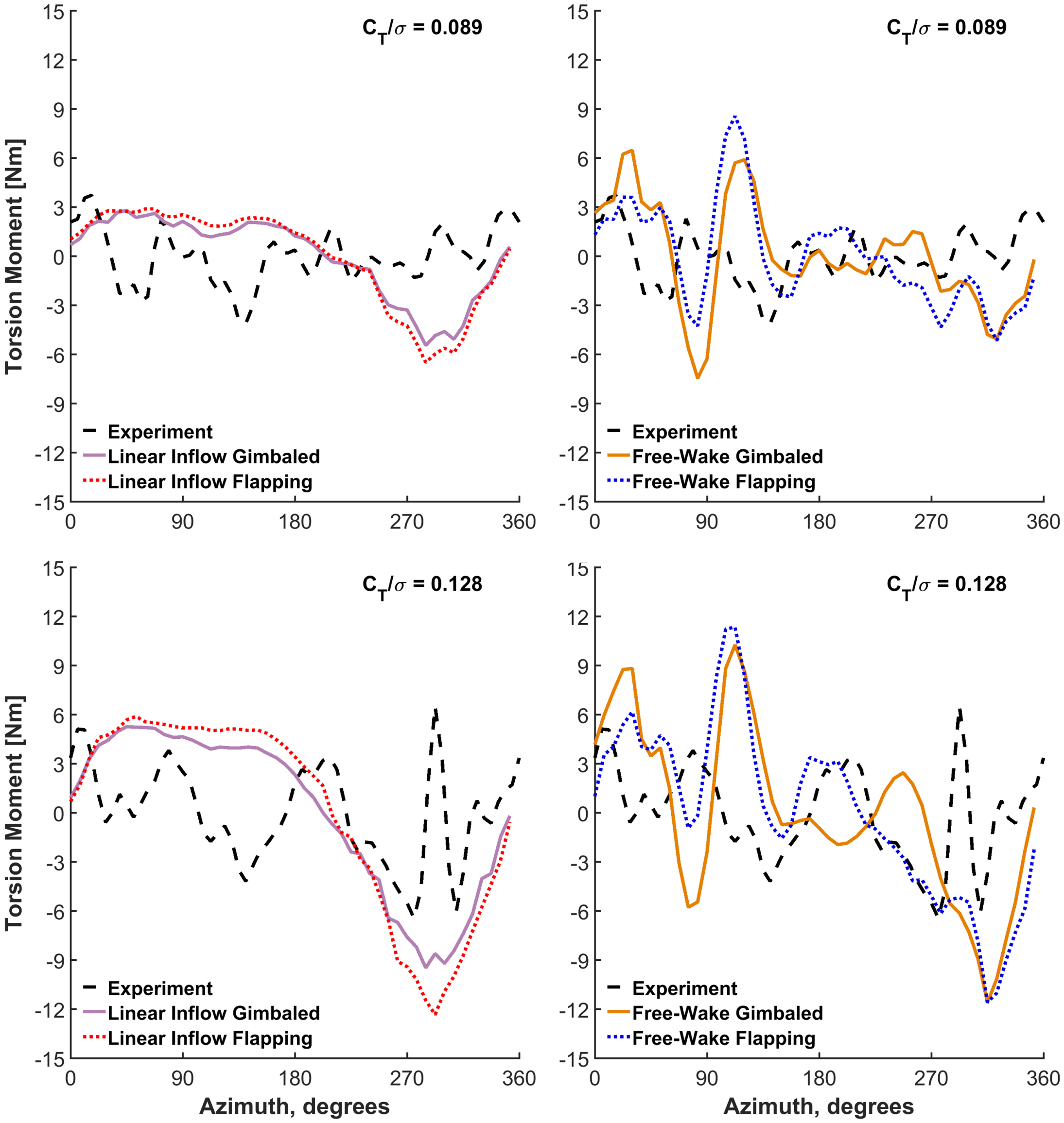


Figure 5.42 Oscillatory sectional torsion moment, X3D predictions at $r/R = 0.48$, compared to experiment at $r/R = 0.50$, two aerodynamic models at two thrust conditions, flapping and gimbaled hub

5.3.3.2 Blade Loads Harmonics

Steady flap, lag, and torsion moments are plotted in Figure 5.43 for both thrust conditions using the flapping and gimbaled hub models. Adding the gimbal has the greatest impact on the flap bending moment, where it improves the prediction. The effect on steady lag bending moment is smaller and mildly beneficial, though both models predict loads an order of magnitude higher than seen in the experiment. Similarly, the steady torsion moment is reduced with the gimbaled model, reducing the error but still over-predicting moment by a factor of seven at moderate thrust and a factor of five at high thrust. Concrete conclusions are premature without data at other radial stations.

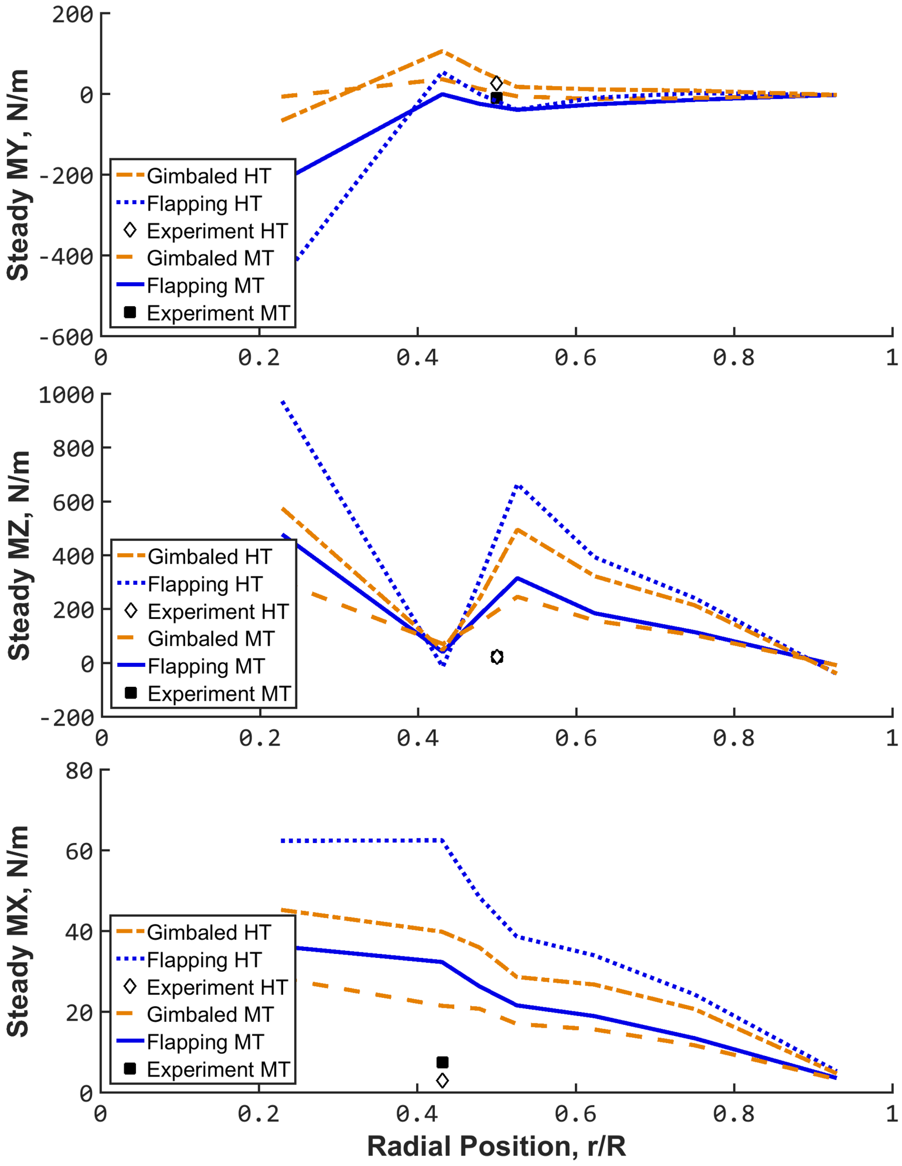


Figure 5.43 Steady flap (MY), lag (MZ), and torsion moments (MX) versus radius, moderate and high thrust, free-wake model, two hubs

Figure 5.44 compares the first four harmonics of flap bending moment at both flight conditions using both flapping and gimbaled hub models with the free-wake aerodynamic model. Both models slightly over-predict 1/rev, greatly over-predict 4/rev, and under-predict 2/rev flap bending, with only minor impact from switching to the gimbaled model. The 3/rev harmonic shows the biggest difference when the gimbal model is included, as expected, with the gimbal model predicting the proper magnitude at high thrust but both models failing at low thrust. In this case the flap bending moment is well predicted by the X3D analysis with the gimbal model at high thrust. Phase at 3/rev, however, is better predicted by the flapping hub model. Recall that the 3/rev pattern is not present in the test data at moderate thrust (Figure 5.37).

Lag bending moment harmonics are presented in Figure 5.45, with and without the gimbal model. As with the flap bending moment, the greatest difference between flapping and gimbaled rotor models appears at 3/rev, though both have similar magnitudes at 50% radius where the experimental measurements were taken. At 3/rev, phase is once again better predicted with the flapping model. Again, conclusions are premature based on data at one particular radial station. Predictions, however, present clear trends in the loading and match data at the single point reasonably well, lending confidence to prediction accuracy.

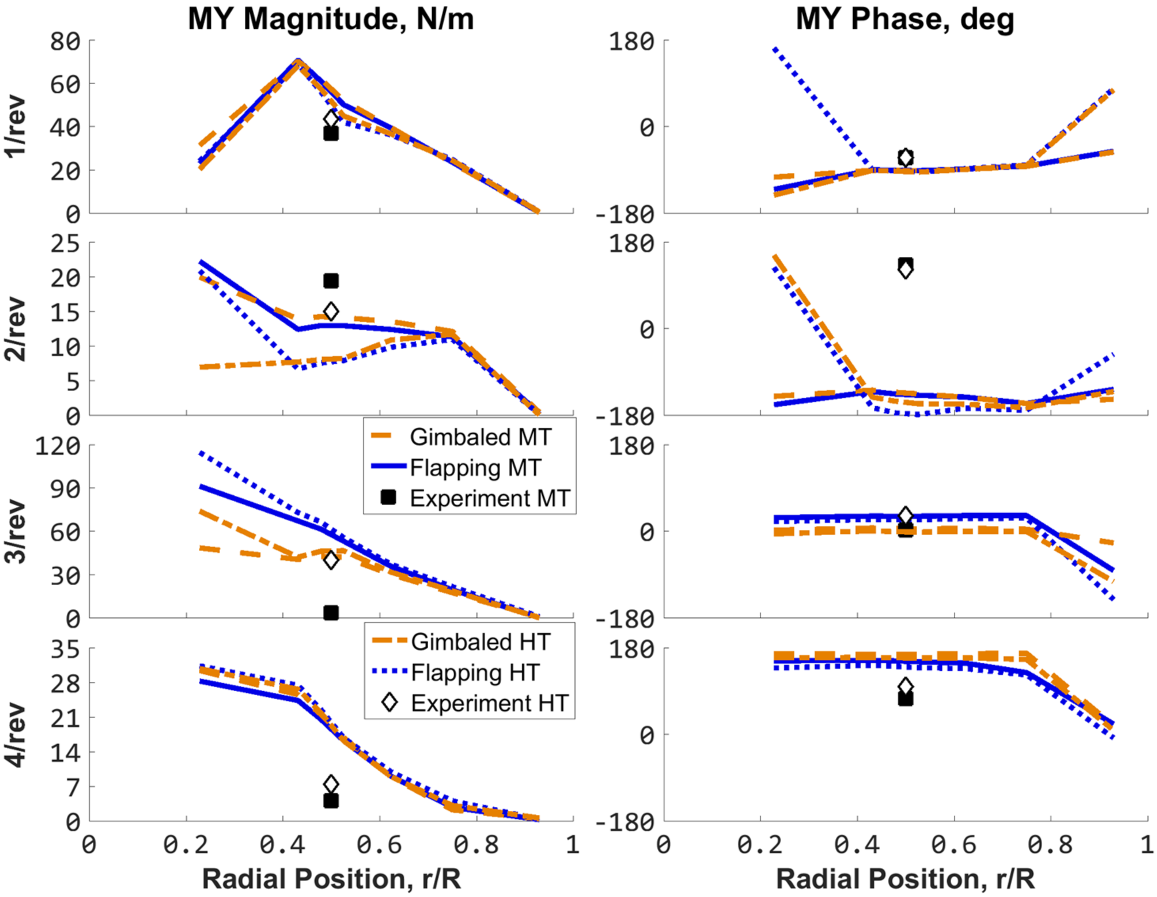


Figure 5.44 Harmonic flap bending moment (MY) versus radius, moderate ($C_T/\sigma = 0.089$) and high ($C_T/\sigma = 0.128$) thrust, free-wake model, flapping and gimbaled hubs

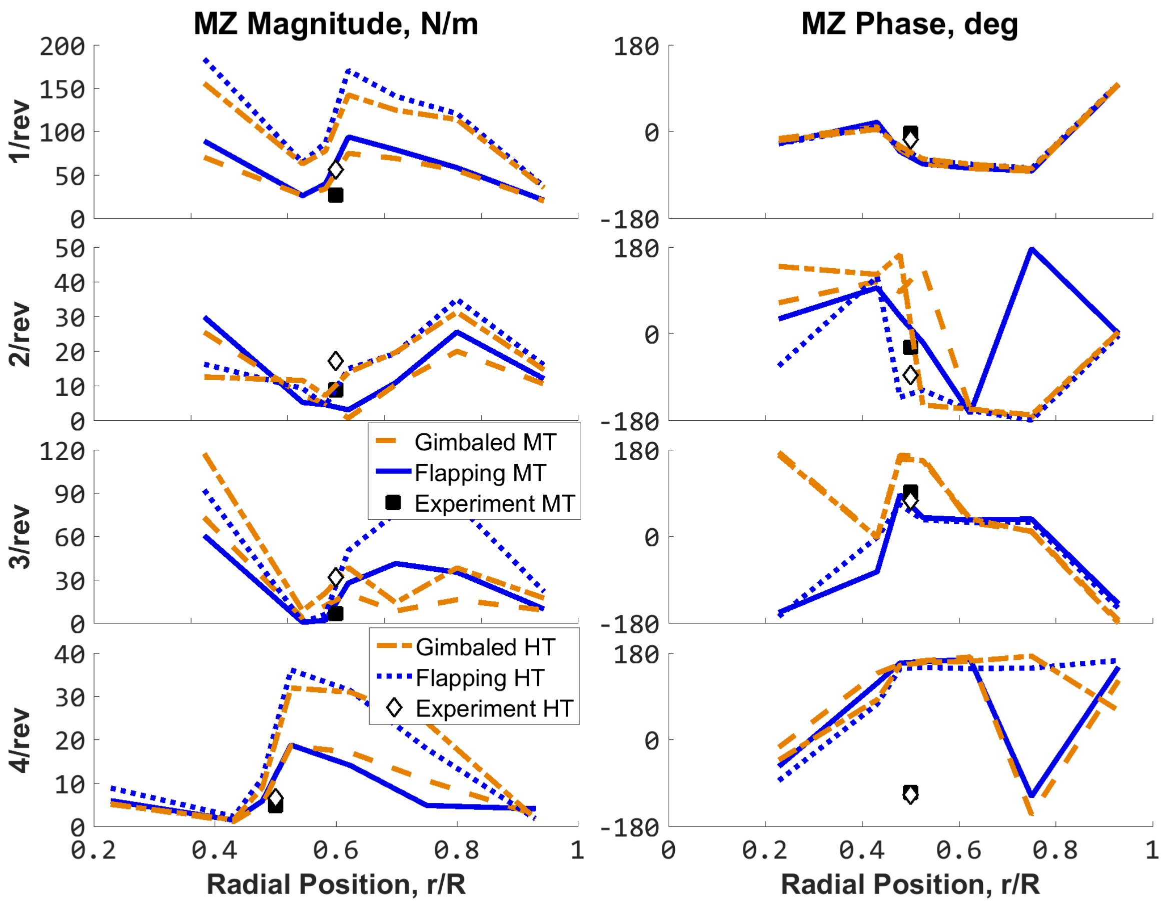


Figure 5.45 Harmonic lag bending (MZ) moment versus radius, moderate ($C_T/\sigma = 0.089$) and high ($C_T/\sigma = 0.128$) thrust, free-wake model, flapping and gimbaled hubs

5.3.4 Edgewise Flight Internal Stresses and Strains

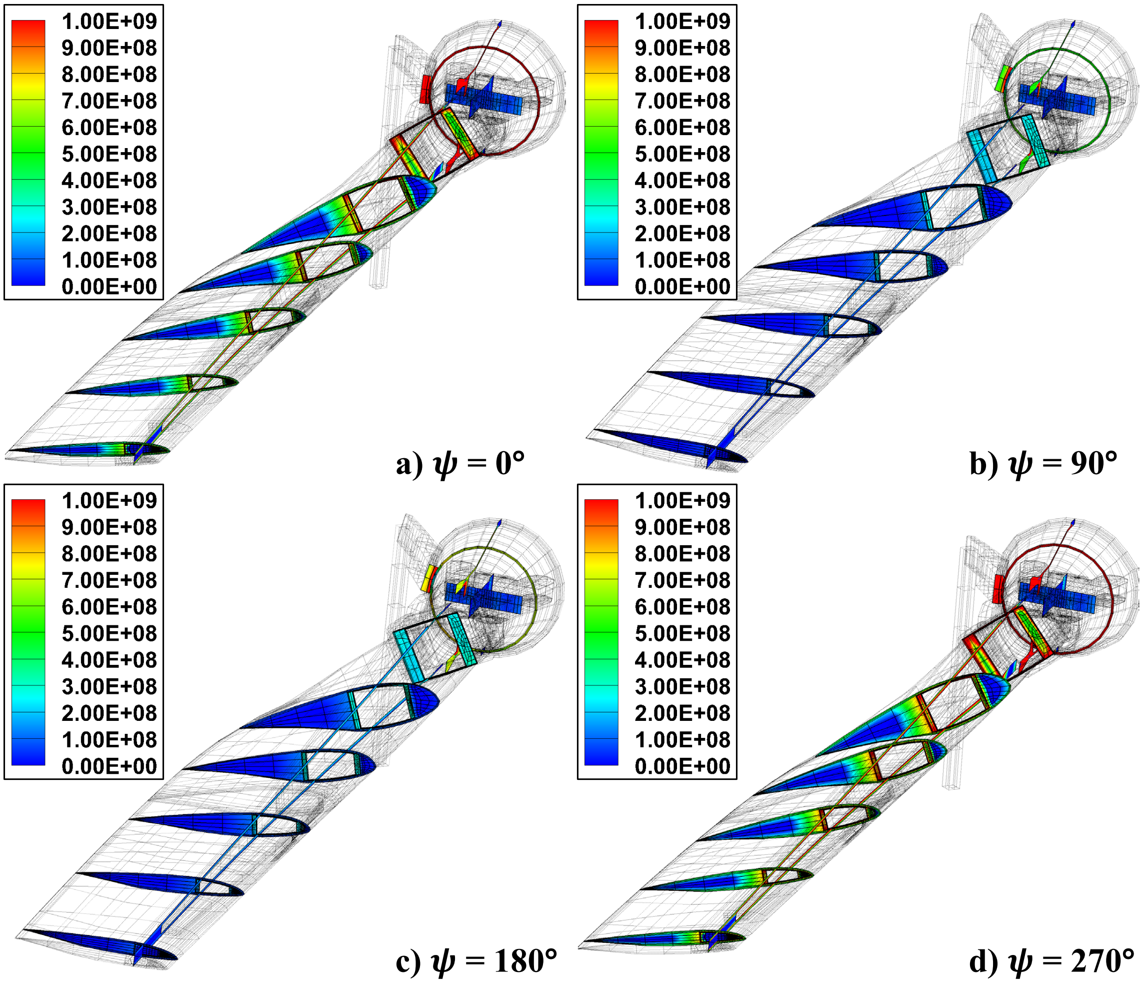
One of the intrinsic benefits to full 3-D FEA based structural modeling and analysis is the ability to predict stresses and strains throughout the rotor, including the non-slender hub components. Although there are no experimental measurements to compare predictions against, examining the 3-D stress field predicted by X3D can still be instructive. Three stresses are shown, axial/bending stress (σ_{11}), in-plane shear stress (σ_{12}), and out-of-plane shear stress (σ_{13}). The stress direction indices refer to the rotating global axes, with the 1-direction being oriented axially along radius of the un-deformed blade, the 2-direction pointing in the plane of the rotation in the direction of the leading edge, and with the 3-axis pointing vertically upward.

All stresses shown are predicted by X3D with RANS aerodynamics for the high thrust condition at 0°, 90°, 180°, and 270° azimuths. Due to the limitations of the visualization software, these figures are generated by breaking each 27-noded hex element into eight 8-noded hex elements and linearly interpolating stress across the corners; as such this visualization does not represent the second-order shape functions which in fact exist in the analysis. All 3-D stress plots shown are from coupled CFD/CSD analysis of the high thrust test case. The plots shown here provide only a small sample of the results generated for purposes of making a few key observations, but there is a wealth of detailed data available for in depth analysis.

Figure 5.46 shows the internal axial/bending stresses (σ_{11}) within the blade. Examination of the cross-sections reveals that the blade spar takes most of the stress, as expected, but there appears to be some stress carried by the blade core and the leading edge weight, most visible as a green area aft of the spar in all sections at 0° azimuth. This is simply an artifact of the linear interpolation of stress at the spar wall across the element due to the low-order visualization. The highest stresses occur from the retreating side around to 0° azimuth, which is consistent with flap and lag bending moment patterns. Higher than anticipated axial stresses can be observed at the exposed root of the blade spar (Figure 5.46, panes (a) and (d)). These predicted stresses are higher than what the blade materials can withstand, and as such cannot represent the TRAM blade as built. The source of this over-prediction is unknown, but possible sources include the lack of stress-relieving precone, errors in the material modulus (a known issue with this structural analysis model), and mesh resolution issues.

Figure 5.47 shows the same axial/bending stresses from the root end of the blade. These plots reveal that the flexbeam is not highly stressed, despite the high thrust condition. The pitchcase, on the other hand, experiences very high stresses around the retreating side to 0° azimuth. This might be an artifact of the material selection; aluminum was used in this structural analysis model, though the pitchcase in the wind tunnel model was switched to titanium (although there is no data to test this conjecture). It may also indicate improper load sharing between the pitchcase and the flexbeam, possibly indicating the importance of bearing joints, whose exact stiffness properties are unknown. Localized stress patterns are observable in the pitch case, especially at 180° azimuth.

Figure 5.48 and Figure 5.49 show the in-plane shear stress (σ_{12}) for the rotor blade and root, while Figure 5.50 and Figure 5.51 show out-of-plane shear stress (σ_{13}). As with σ_{11}, shear stress is highest from 270° around to 0° azimuth. The flexbeam is not highly stressed in shear. It can also be seen that the in-plane shear stress is higher than out-of-plane shear stress at all radial locations and azimuths. The out-of-plane shear stress shows intense localized behavior in the pitch case (Figure 5.51), and is predicted to be negative in some regions.



a) $\psi = 0°$

b) $\psi = 90°$

c) $\psi = 180°$

d) $\psi = 270°$

Figure 5.46 Blade axial/bending stress σ_{11}, N/m^2, high thrust condition

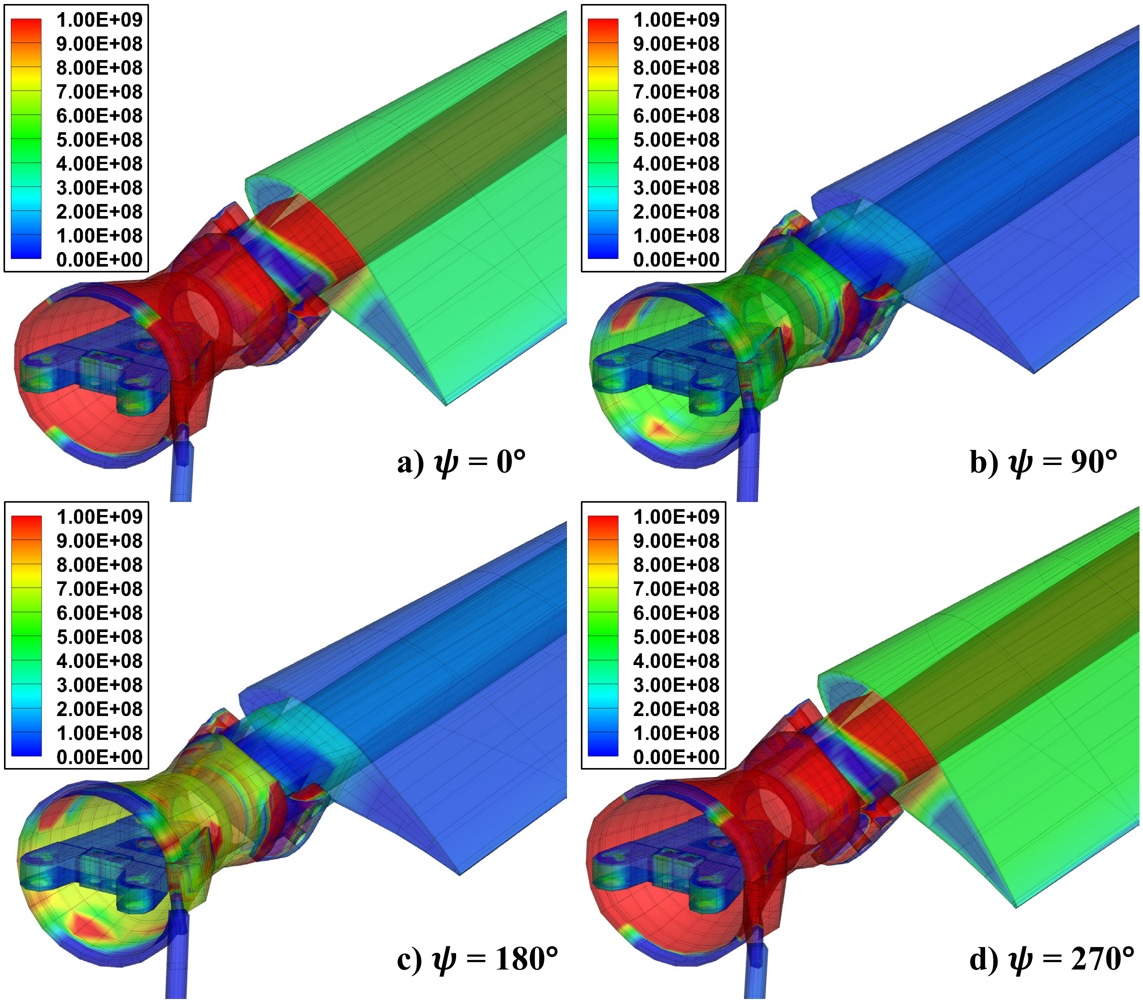
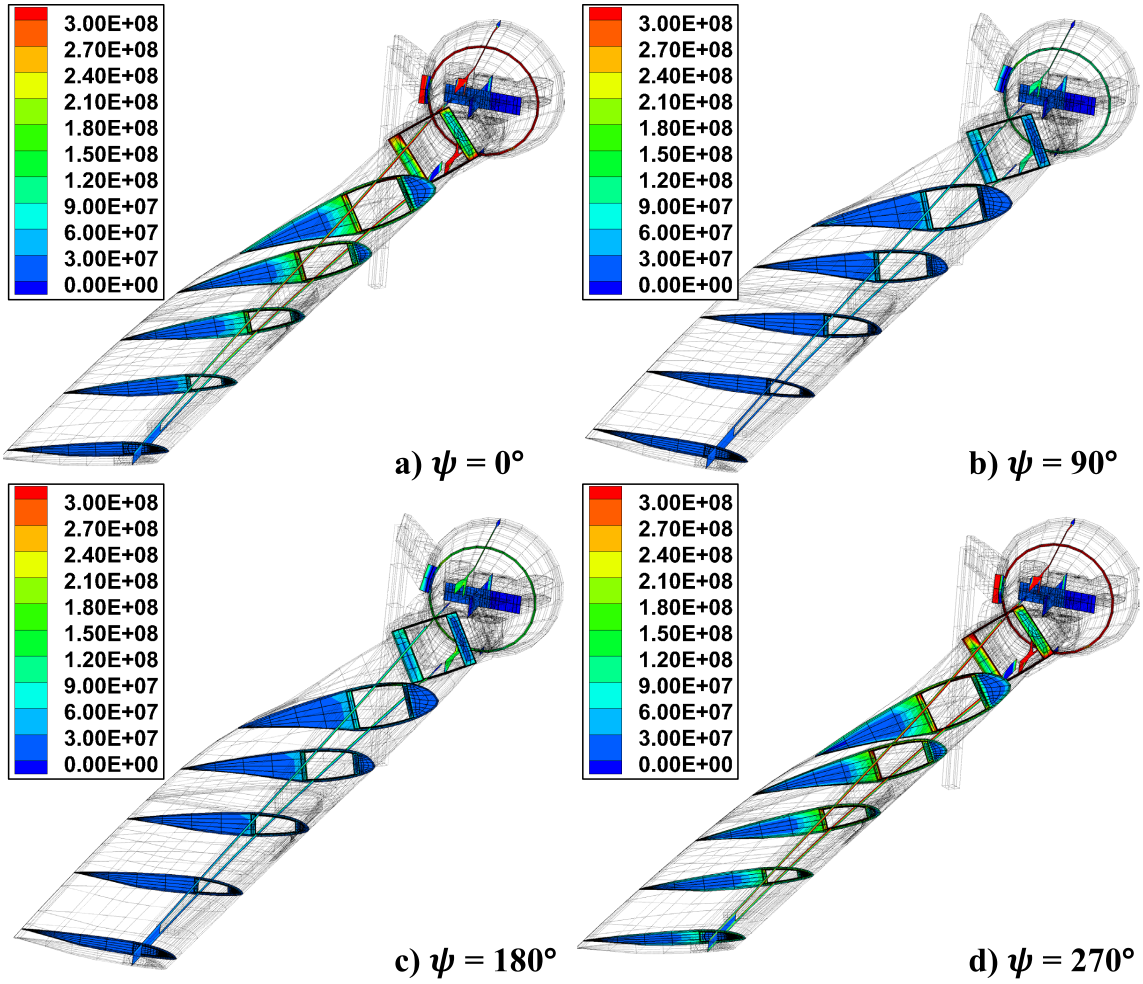


Figure 5.47 Root axial/bending stress σ_{11}, N/m^2, high thrust condition



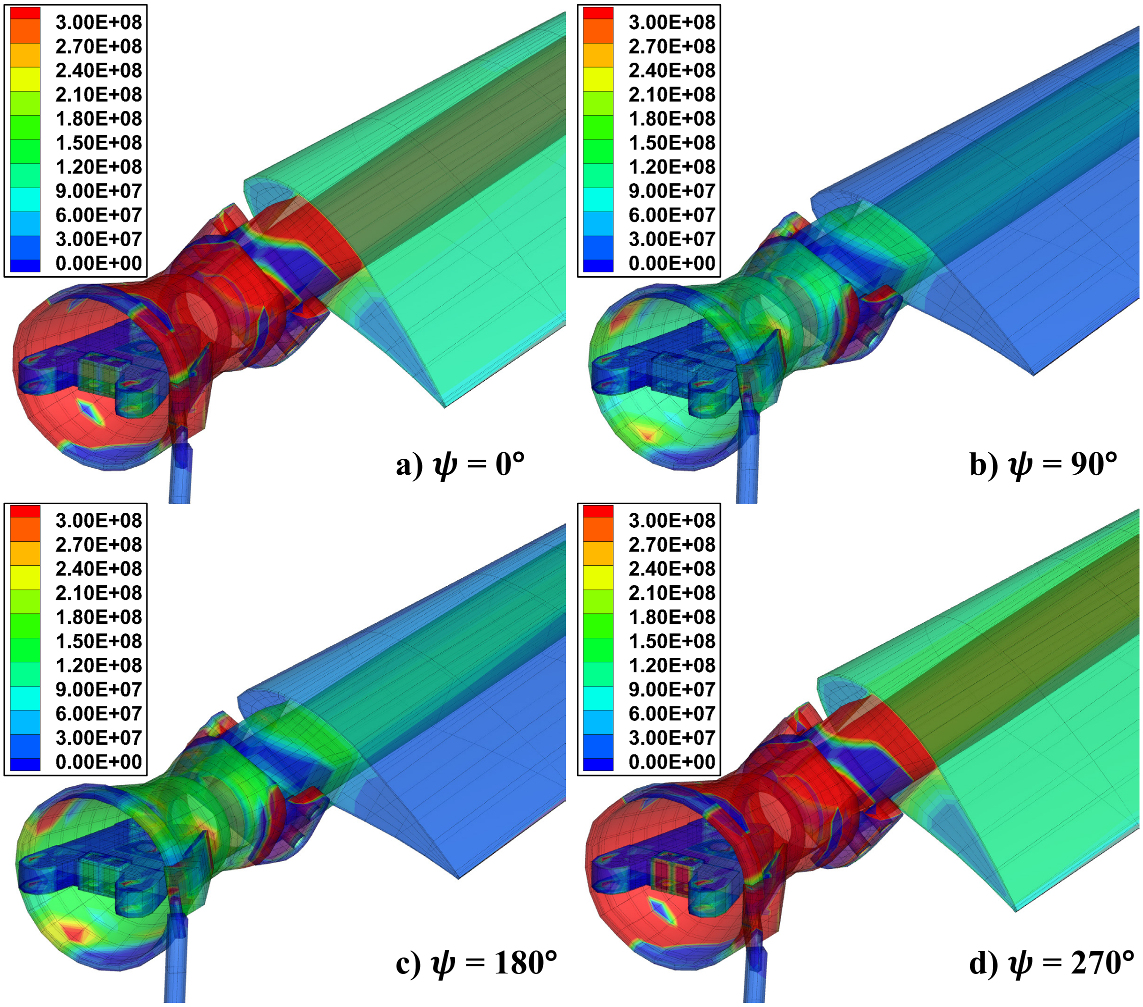
a) ψ = 0°

b) ψ = 90°

c) ψ = 180°

d) ψ = 270°

Figure 5.48 Blade in-plane shear stress σ_{12}, N/m^2, high thrust condition



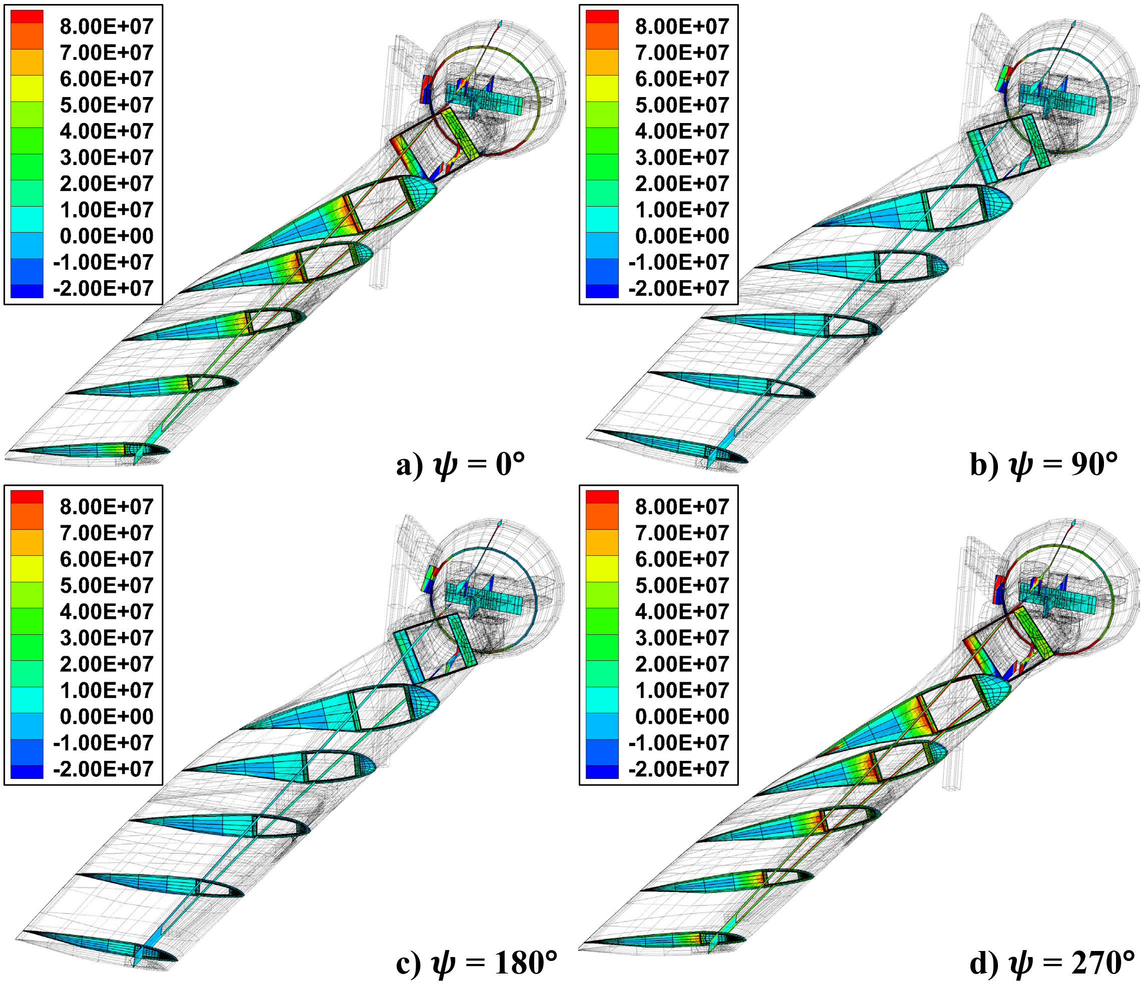
a) $\psi = 0°$

b) $\psi = 90°$

c) $\psi = 180°$

d) $\psi = 270°$

Figure 5.49 Root in-plane stress σ_{12}, N/m^2, high thrust condition



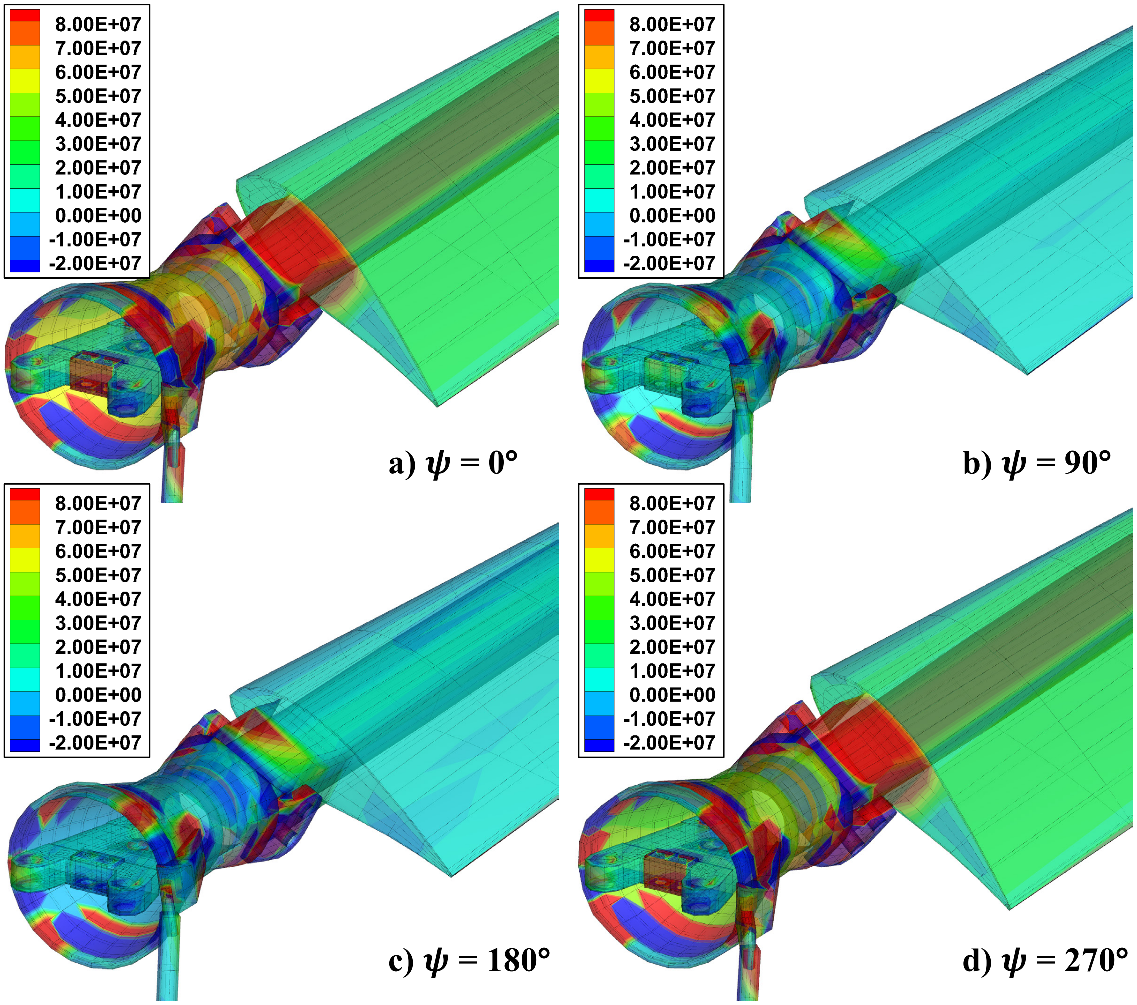
a) ψ = 0°

b) ψ = 90°

c) ψ = 180°

d) ψ = 270°

Figure 5.50 Blade out-of-plane shear stress σ_{13}, N/m^2, high thrust condition



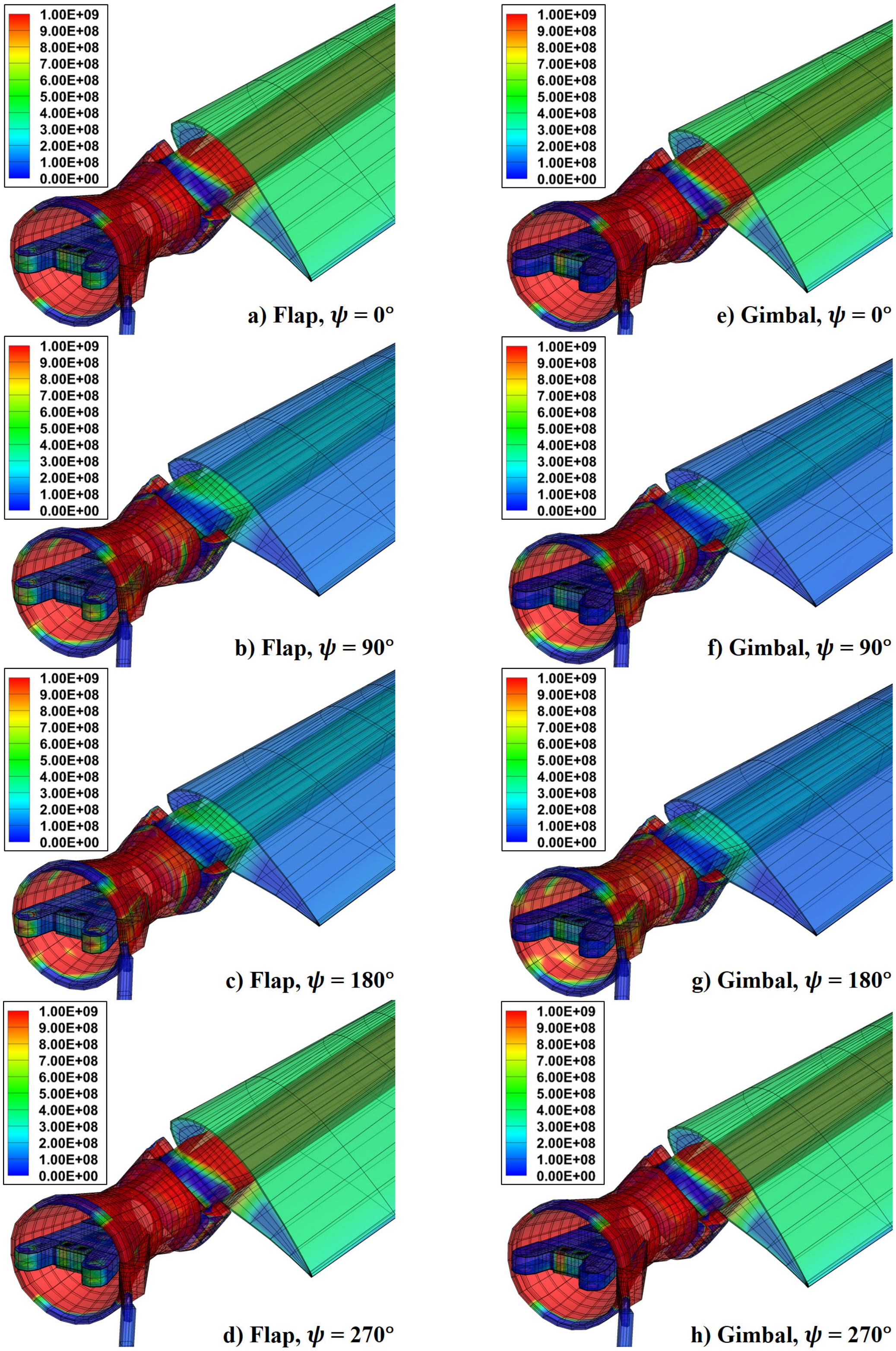
a) $\psi = 0°$

b) $\psi = 90°$

c) $\psi = 180°$

d) $\psi = 270°$

Figure 5.51 Root out-of-plane shear stress σ_{13}, N/m^2, high thrust condition



a) Flap, ψ = 0°

e) Gimbal, ψ = 0°

b) Flap, ψ = 90°

f) Gimbal, ψ = 90°

c) Flap, ψ = 180°

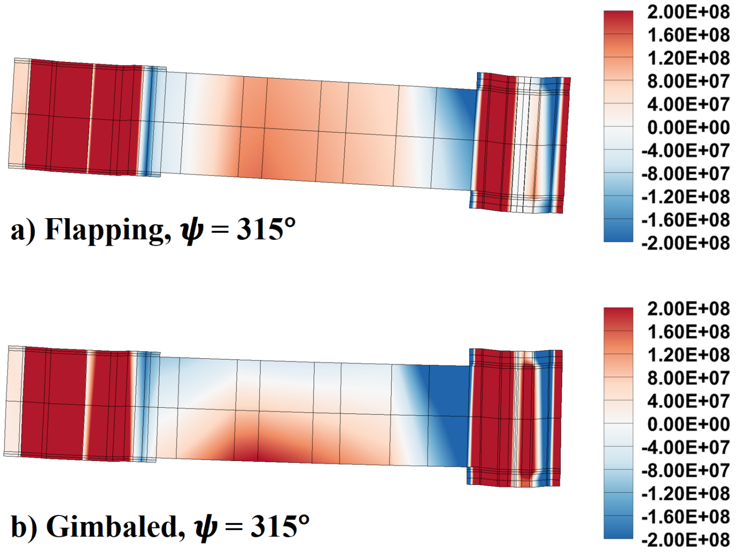
g) Gimbal, ψ = 180°

d) Flap, ψ = 270°

h) Gimbal, ψ = 270°

Figure 5.52 Axial/bending stress σ_{11} in the flexbeam, high thrust, free-wake, (a) flapping and (b) gimbaled hub models

To understand the effect of the gimbal on proprotor stress, Figure 5.52 shows these stresses at the root end of the rotor for the high thrust flight modeled using free-wake aerodynamics with both flapping and simulated gimbal hubs. The stress patterns using both rotor models are very similar, despite significant differences in airloads. Also intriguing is the flexbeam, which appears to take less stress in the gimbaled case. Looking closer and changing the color scale (Figure 5.53) reveals that the flexbeam stress is uniform in the flapping case. The gimbaled case, in contrast, has low stress on the top and higher stress locally on the bottom. This indicates that the flexbeam is undergoing elastic bending to allow coning motion, as expected



a) Flapping, $\psi = 315°$

b) Gimbaled, $\psi = 315°$

Figure 5.53 Axial/bending stress σ_{11} at the root of the rotor at four azimuths, high thrust condition, free-wake aerodynamic model, flapping and gimbaled hub model

Chapter 6: Discussion of Results

6.1 Effect of Model Properties on Rotor Structural Dynamics

The initial TRAM structural analysis model showed significant differences in the modal frequencies from experimental and prior analytical studies performed by Johnson using a beam-based solver with measured root conditions [95]. These rotating frequencies are shown in Figure 6.1. Specifically, the 3-D solver predicted the first chordwise mode to be at higher frequency than expected. Additionally, the higher modes did not match with either the experimental data or the beam-based analysis of Ref. [95]. It should be noted that Johnson's beam analysis does not match the first chordwise frequency to the measured non-rotating lag frequency either. It is believed that this non-rotating measurement is suspect, as forcibly matching it leads to an unrealistically low first lag frequency at operating RPM (< 1/rev). Instead, Johnson used a root lag spring stiffness based on a static chordwise bending test, rather than tailoring it to the nonrotating rap test frequency. It is believed that this increased first lag frequency is more realistic. To find the source of the discrepancies between the original analysis, experimental measurements, and beam-based models, three possible sources of error are investigated: the flexbeam materials, the stiffness of the multibody joints in the structural analysis model, and the blade structural model itself.

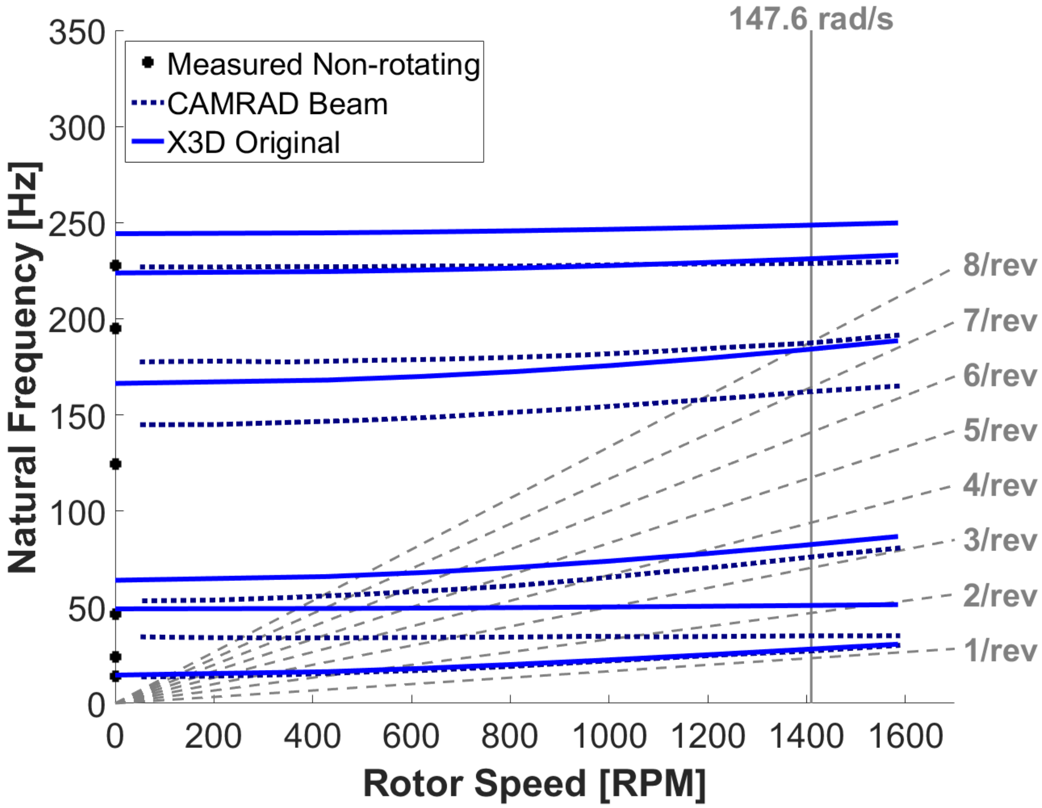


Figure 6.1 Original fan plot for the single bladed 3-D TRAM structural analysis model, gimbal locked

6.1.1 Flexbeam Materials

Difficulties in finding accurate material properties for the composites in the TRAM rotor led to suspicions that the materials were not being accurately modeled. This is particularly evident for the flexbeam. Comparison between 3-D static analysis and experimental bending tests found the flexbeam structural analysis model to be significantly stiffer in chordwise bending, although flap bending was well matched (Figure 5.3).

To evaluate whether the material properties applied to the flexbeam model were the source of the error in the rotating natural frequencies, the effect of varying the elastic and shear moduli of the flexbeam plies was examined. Sweeping the material input parameters showed that matching the first lag frequency by reducing chordwise stiffness of the flexbeam requires driving the shear modulus to unrealistically low levels. Lowering the shear stiffness of the flexbeam materials did improve the first chordwise bending mode frequency of the rotor, as seen in Figure 6.2, but it did not improve the higher modes. Because the values needed to resolve the error in first lag frequency were unrealistically low, it was decided to instead use more realistic material properties obtained from CMH-17 [142], leaving the flexbeam chordwise static bending stiffness issue unresolved.

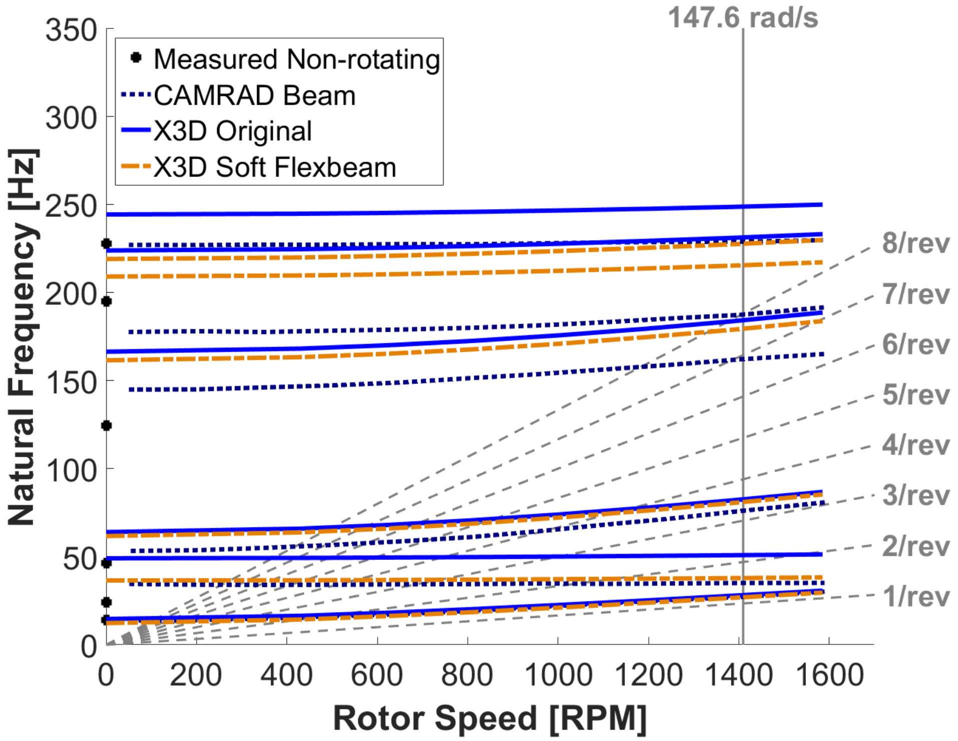


Figure 6.2 Fan plot for the TRAM rotor with a low shear modulus (G_{12}) flexbeam (orange) compared to the original model (blue)

6.1.2 Joint Stiffness

In the initial 3-D TRAM rotor model, it was assumed that most of the joints in the model were either rigid or free. The inboard centering bearing was given a vertical degree of freedom with a stiffness (K_{33}) of 1.3×10^6 N/m but the lagwise stiffness (K_{22}) was initially treated as rigid. Physically, however, this joint would have some compliance due to the fact that it represents not just the bearing, but also the carrier holding it, as seen in Figure 4.10.

To examine its effect on the TRAM modal frequencies, the values for the inboard centering bearing stiffness in the lagwise direction were swept from rigid to very compliant. As can be seen in Figure 6.3, reducing the value of the in-plane stiffness of the inboard centering bearing brings the second chordwise mode in line with Johnson's CAMRAD II analysis. The sixth modal frequency also showed a noticeable impact due to the change in the rotor lag stiffness.

Having demonstrated that softening the in-plane stiffness of the inboard centering bearing improves the first lag frequency, the out-of-plane stiffness (K_{33}) of the inboard centering bearing was next investigated to see if improvements could be made to the higher modes. It was found that lowering the vertical stiffness sufficiently could bring the higher frequencies into line with the results of the beam-based TRAM analysis (Figure 6.4). However, doing so required decreasing K_{33} significantly, from its initial value of 1.3×10^6 N/m to 0.4×10^6 N/m. Although the change improved matching of the fourth and fifth modal frequencies, it resulted in increased torsion coupling in the higher modes, eliminating a distinct first torsion mode. This result was suspected to be non-physical.

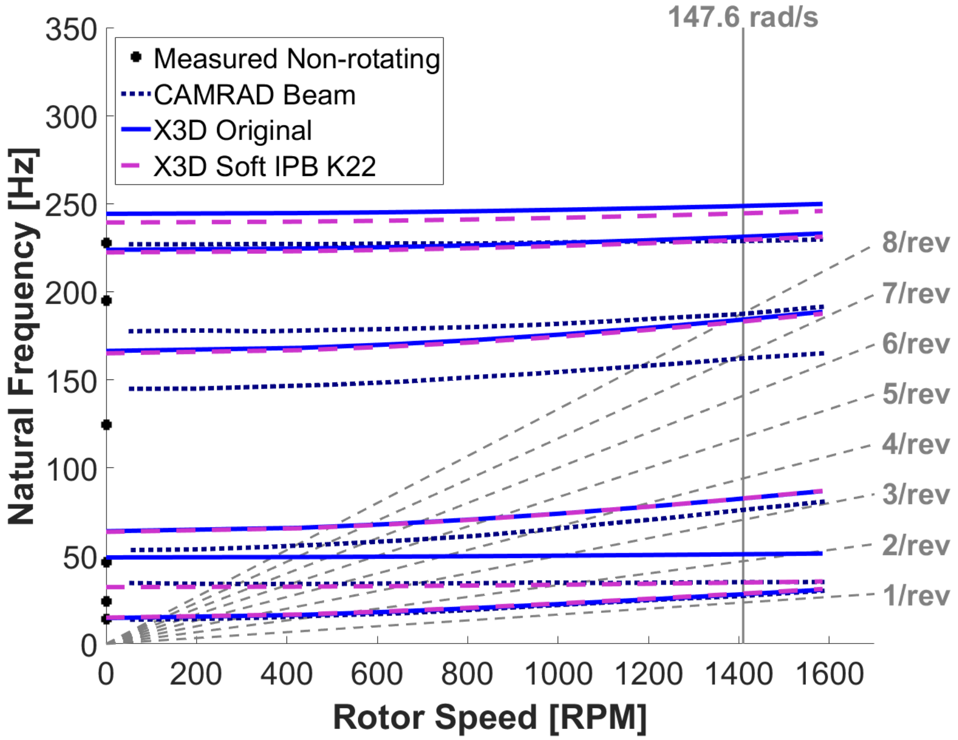


Figure 6.3 Fan plot of the TRAM rotor with softened inboard pitch bearing (IPB) in-plane stiffness, $K_{33} = 1.3E6$ N/m (purple), compared to the original rigid case (blue)

With no physical guidelines upon which to base the selection of joint stiffness, it was decided to set the K_{33} and K_{22} values both equal to 1.3×10^6 N/m, as in Figure 6.3, matching first lag but not higher frequencies. Future analysis will examine this issue, possibly simulating the inboard centering bearing carrier with 3-D FEM, thereby modeling the compliance joint from first principles, rather than relying on tailoring the stiffness.

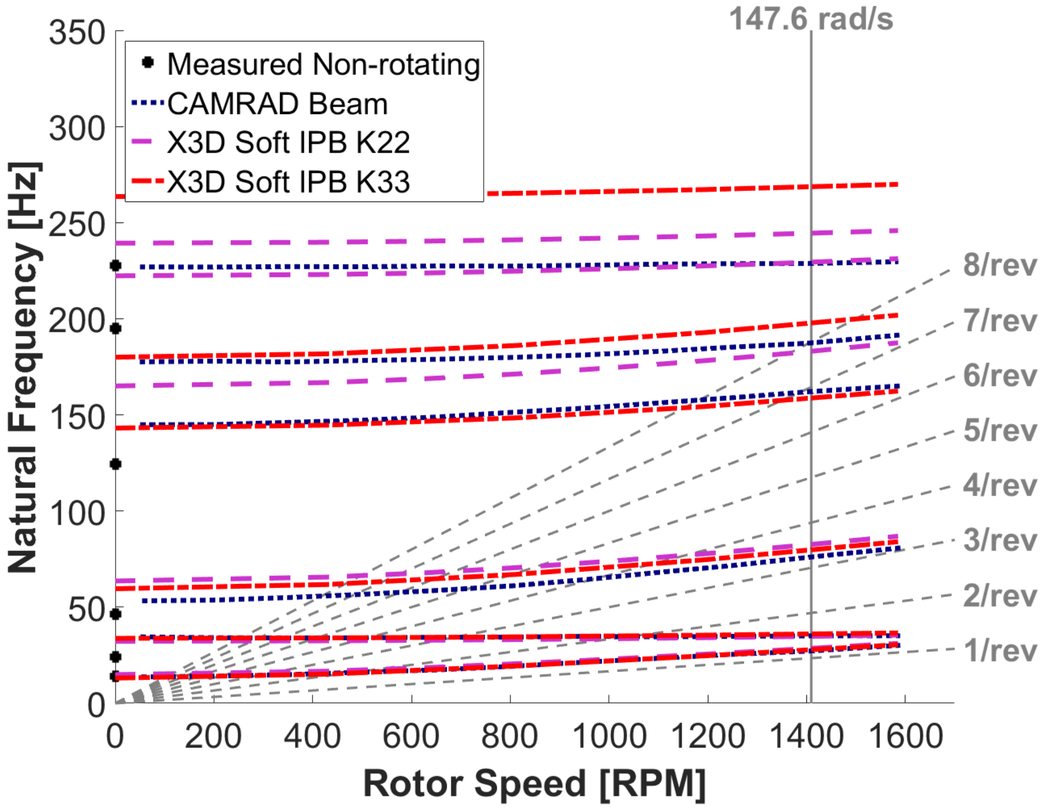


Figure 6.4 Fan plot of the TRAM rotor with softened inboard pitch bearing (IPB) out-of-plane stiffness $\boldsymbol{K_{33} = 0.4E6}$ N/m (red), compared to the original out-of-plane stiffness $\boldsymbol{K_{33} = 1.3E6}$ N/m (purple) [Both have softened $\boldsymbol{K_{22} = 1.3E6}$ N/m]

6.1.3 Blade Structure

Finally, the blade structural analysis model was examined closely to find the source of differences between the 3-D and 1-D modal analyses. When using 1-D beam structural analysis, the blade is represented using distributed masses and sectional bending stiffness (*EI*) and torsional stiffness (*GJ*). To validate the 3-D model, it is necessary to extract equivalent beam-like sectional properties from the TRAM rotor using the methodology described in Section 2.7.1.

Applying this method to the blade with the original, and suspect, material properties (Figure 6.5) reveals a good match of flapwise bending stiffness EI_y and of torsional stiffness GJ_x, but chordwise bending stiffness EI_z shows a considerable difference compared to the experiment, particularly over the outboard portion of the blade.

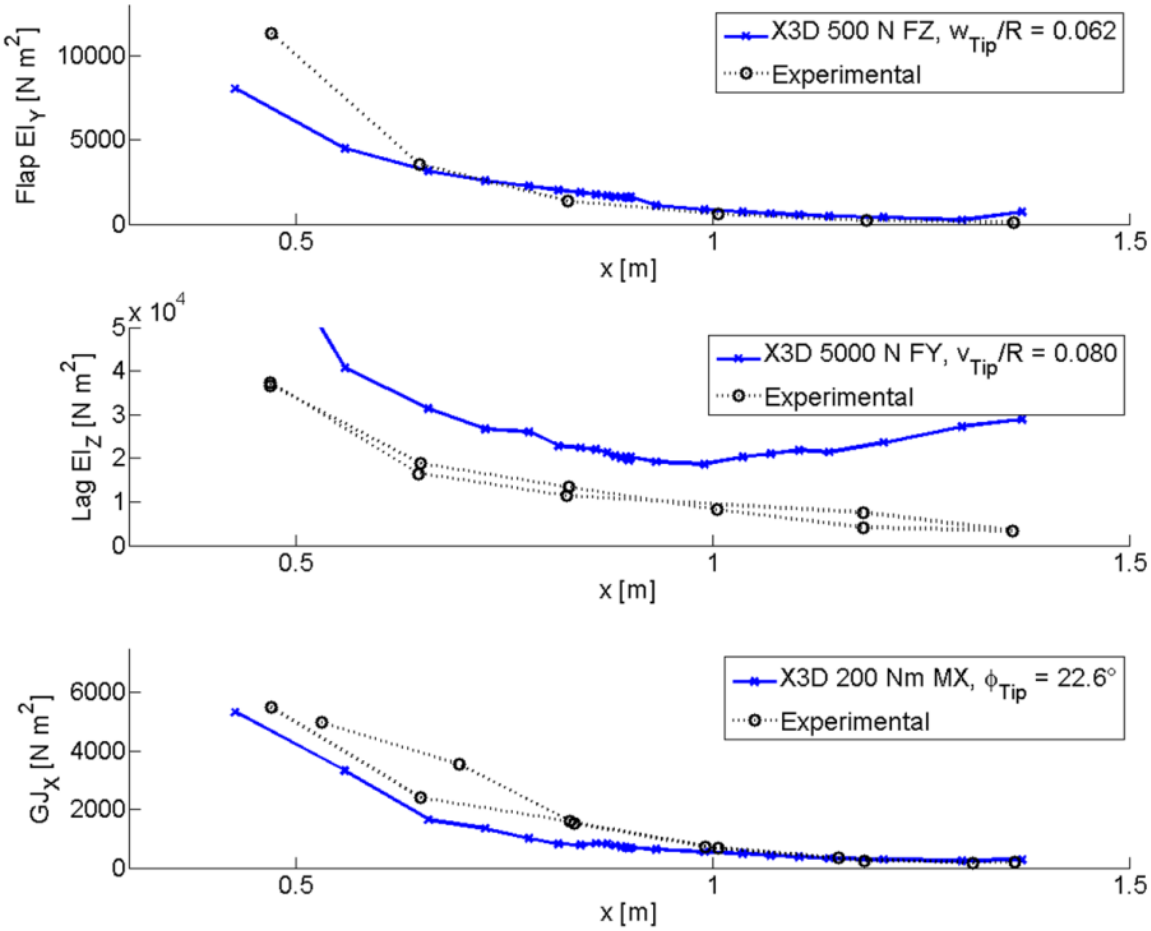


Figure 6.5 TRAM blade sectional properties, flap and chordwise bending and torsional stiffness, original materials

The error in chordwise stiffness in the original blade model could be due to deficiencies in the structural model or the material properties. The 3-D CAD for the blade was built using the manufacturer's cross-sectional drawings at eight stations. At these sections, confidence in the accuracy of the structural analysis model is high. In between, there may be differences between the blade as it was actually built and the internal structure of the CAD model. The fact that the chordwise stiffness errors are no better where the available sections are closely spaced suggests that material discrepancies may be the source of the error.

To examine the effect of material properties, sectional properties were evaluated again after updating the material values to more realistic estimates taken from the CMH-17 composite material hand book [142]. These results, seen in Figure 6.6, show an improvement the flap bending and torsional stiffness. There is also some improvement in the chordwise bending stiffness, especially in the outboard region, but the 3-D model is still approximately twice as stiff in lag as the test article. The errors in chordwise stiffness of the blade, as with errors in chordwise stiffness of the flexbeam, are suspected to be due to incorrect material properties. The blade with the updated material properties was used to create the final fan plot shown in Figure 5.13.

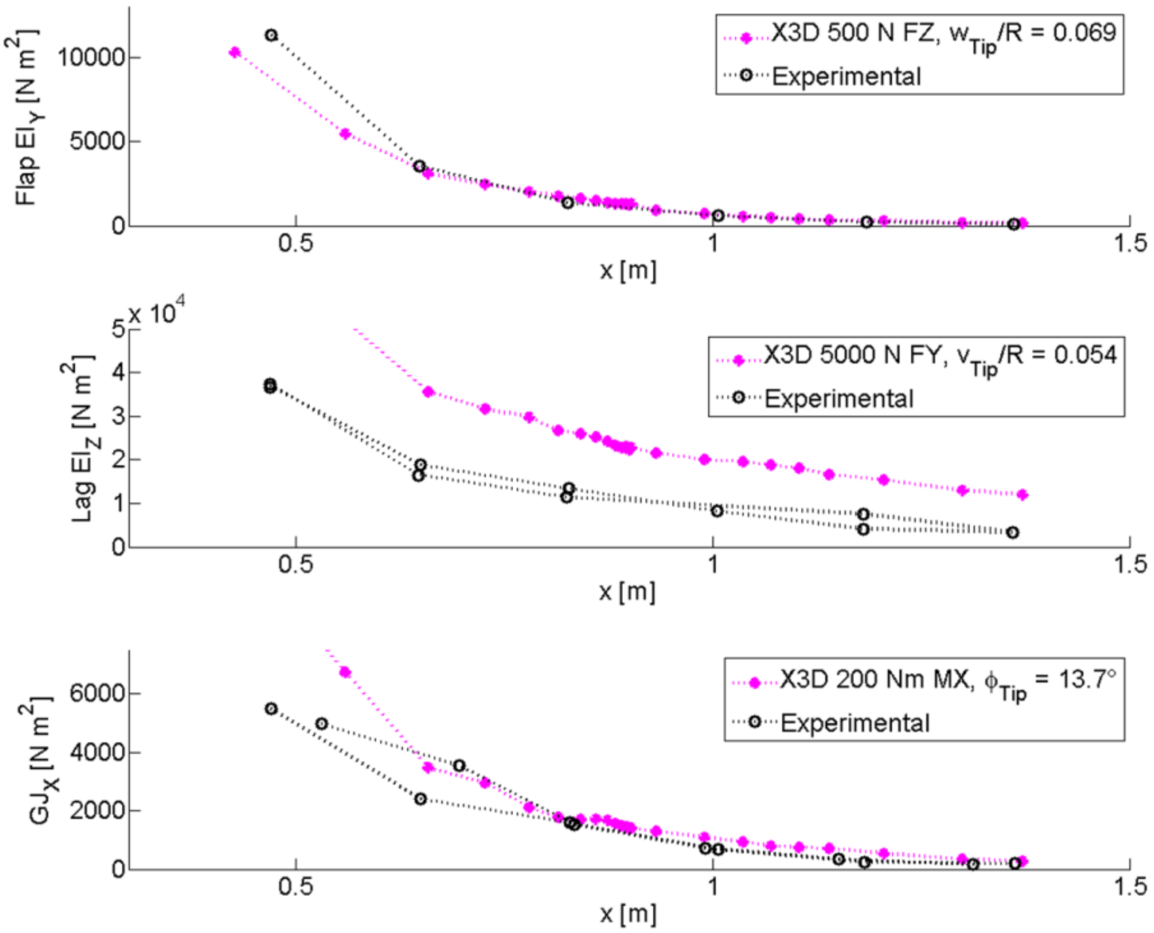


Figure 6.6 TRAM blade sectional properties, flap and chordwise bending and torsional stiffness, updated materials

6.2 Effect of Model Refinement on Performance Predictions

6.2.1 Hover Performance

This section examines predictions of hover performance as the level of modeling is refined. First, rigid blade hover predictions are shown, and a comparison is provided between uniform inflow and BEMT aerodynamics. Then flexibility is added to the model. Finally, the impact of adding the fairing to the aerodynamic model is discussed.

Figure 6.8 presents pure aerodynamic hover performance predictions from X3D for the TRAM rotor using a rigid blade. Results are plotted using both a uniform inflow and a BEMT model, and experimental data for both the TRAM rotor and an earlier JVX rotor test are provided for comparison. The uniform inflow model under-predicts power at high thrust and over-predicts power at moderate thrust. The figure of merit is over-predicted by uniform inflow by approximately 12% at its peak. BEMT improves predictions, particularly for power at moderate thrust, but still under-predicts performance at high thrust.

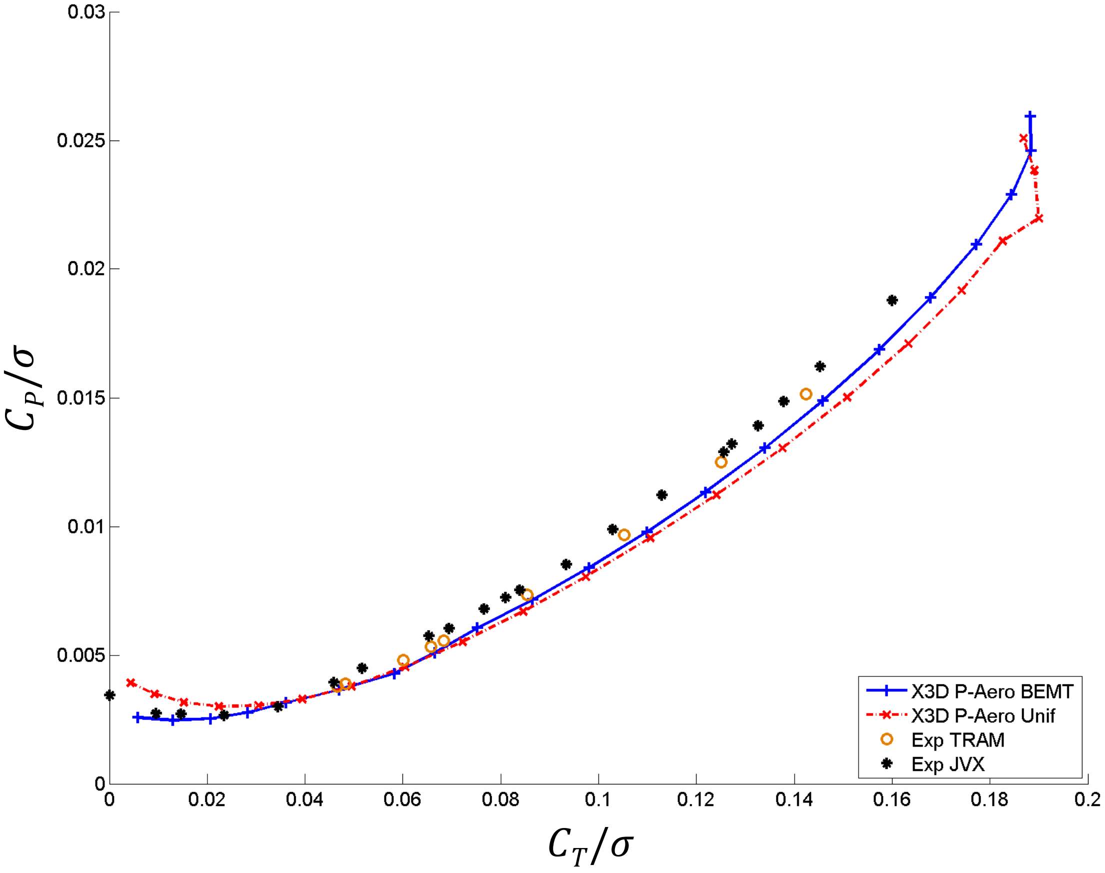


Figure 6.7 Hover performance (power coefficient) for the TRAM rotor; pure aerodynamics (rigid blade)

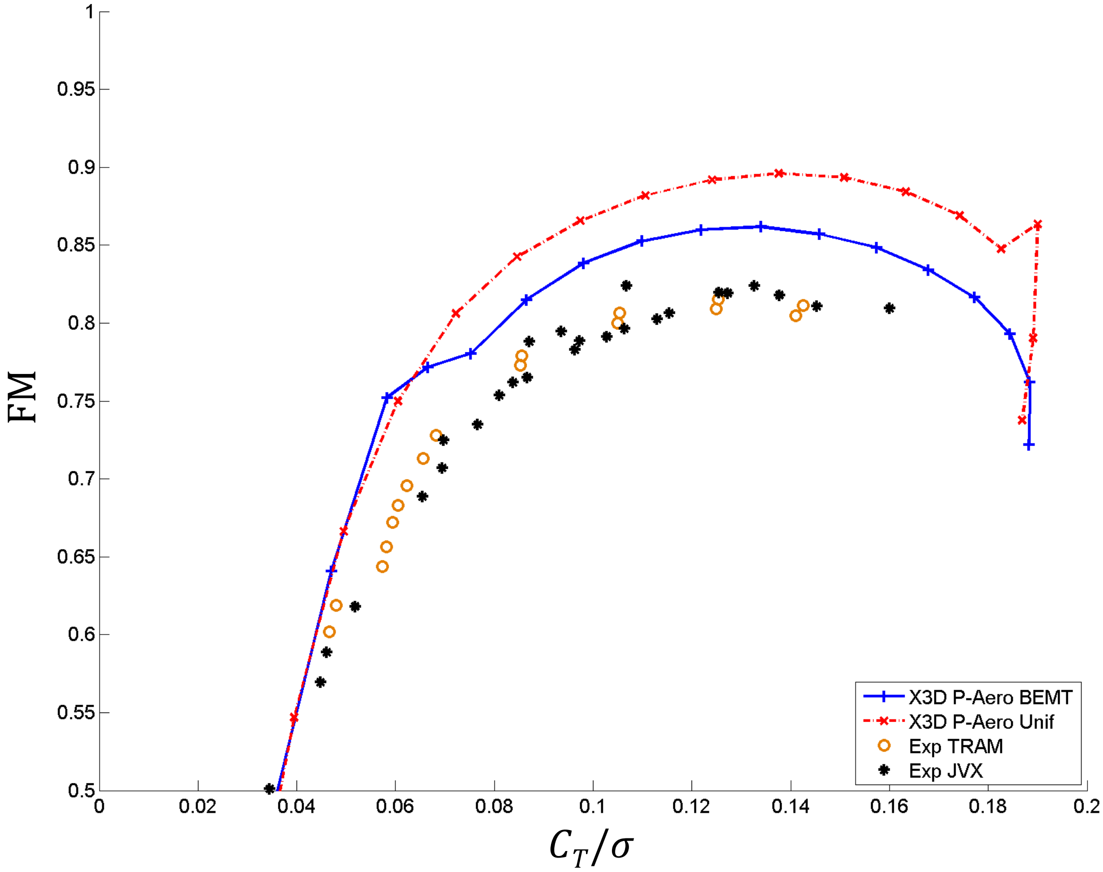


Figure 6.8 Hover performance (figure of merit) for the TRAM rotor; pure aerodynamics (rigid blade)

The effect of adding blade flexibility is considered next. Figure 6.9 shows power required using the BEMT model with and without blade flexibility. As can be seen, adding elastic deformation increases the power required. This result is expected, as blade coning will reduce the efficiency of the rotor. Although it does improve the prediction, the analysis still under-predicts the power required, and thus still over-predicts the figure of merit (Figure 6.10).

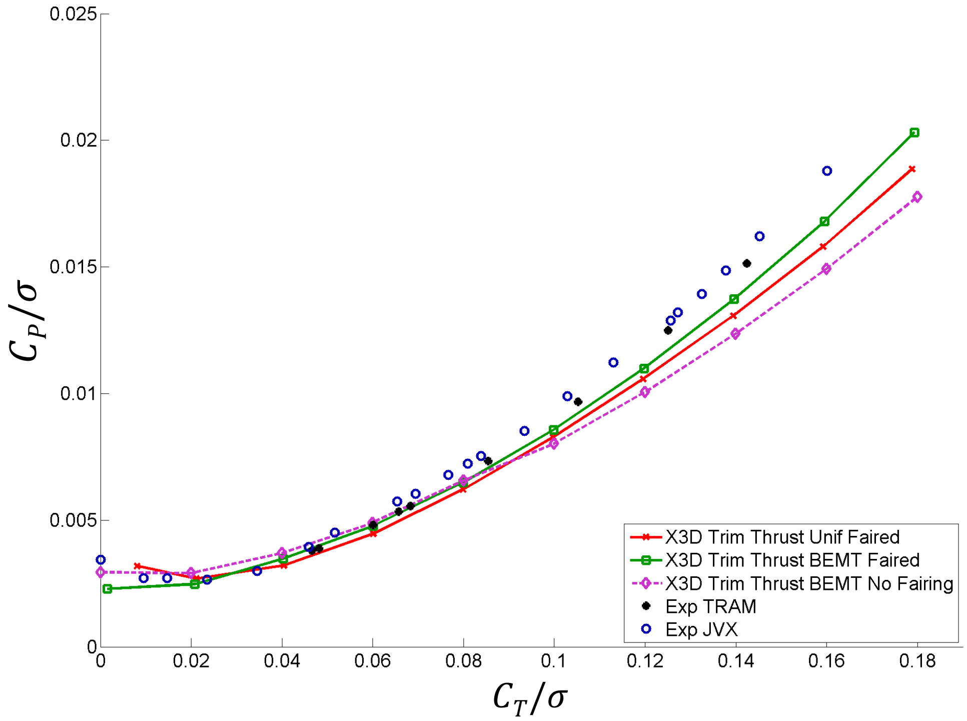


Figure 6.9 Hover performance (power) for the TRAM rotor; flexible blade

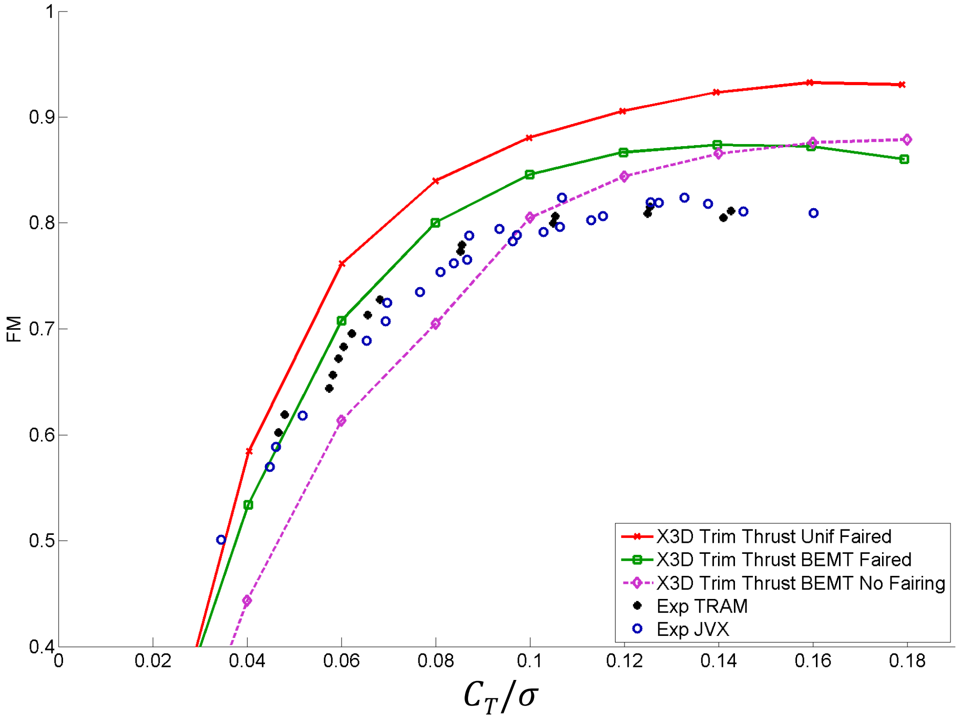


Figure 6.10 Hover performance (figure of merit) for the TRAM rotor; flexible blade

Finally, an aerodynamic model for the fairing which covers in the root of the TRAM blade is applied. As can be seen in Figure 6.11, the fairing adds drag to the prediction. This is because there was originally no aerodynamic model for the root components. The added drag brings hover predictions much closer to the experimental measurements, although there is still an under-prediction of required power, and an over-prediction of the figure of merit (Figure 6.12). As discussed in Section 5.2, this is attributed to the difference in airfoils used between in the model and the actual TRAM test article.

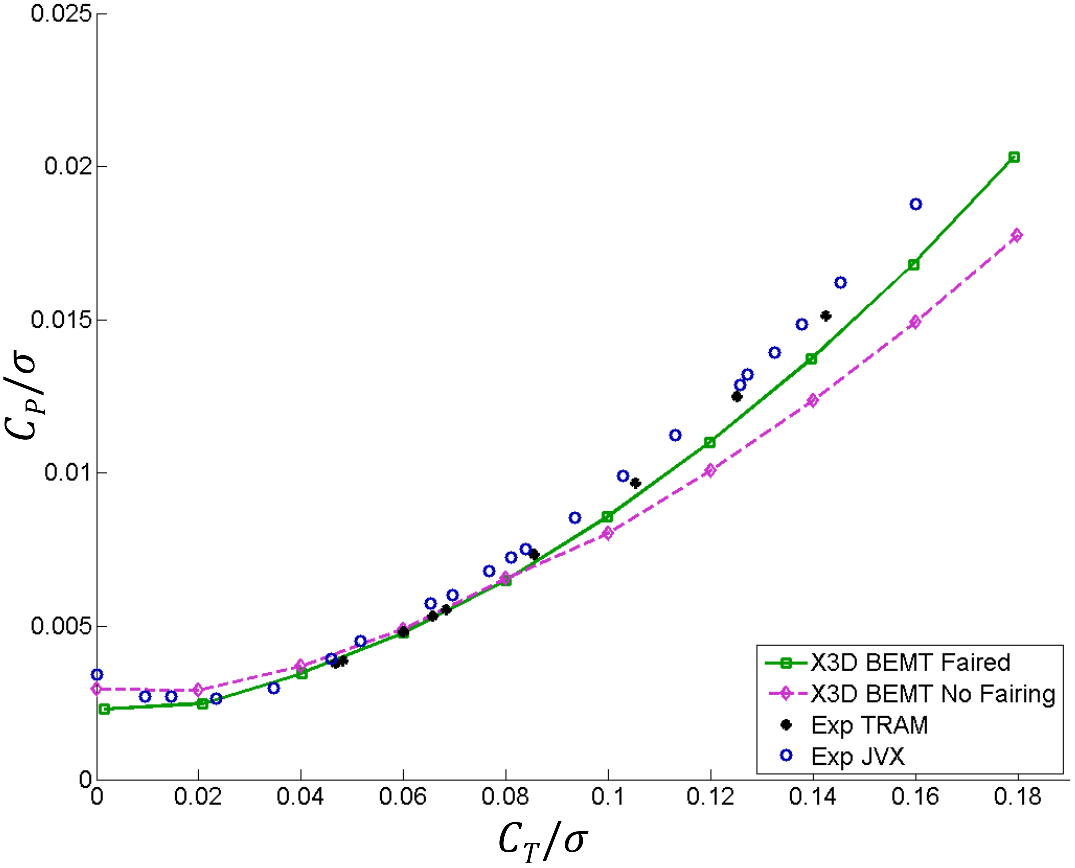


Figure 6.11 Hover performance (power) for the TRAM rotor; X3D coupled aero/structures, BEMT with and without a fairing model

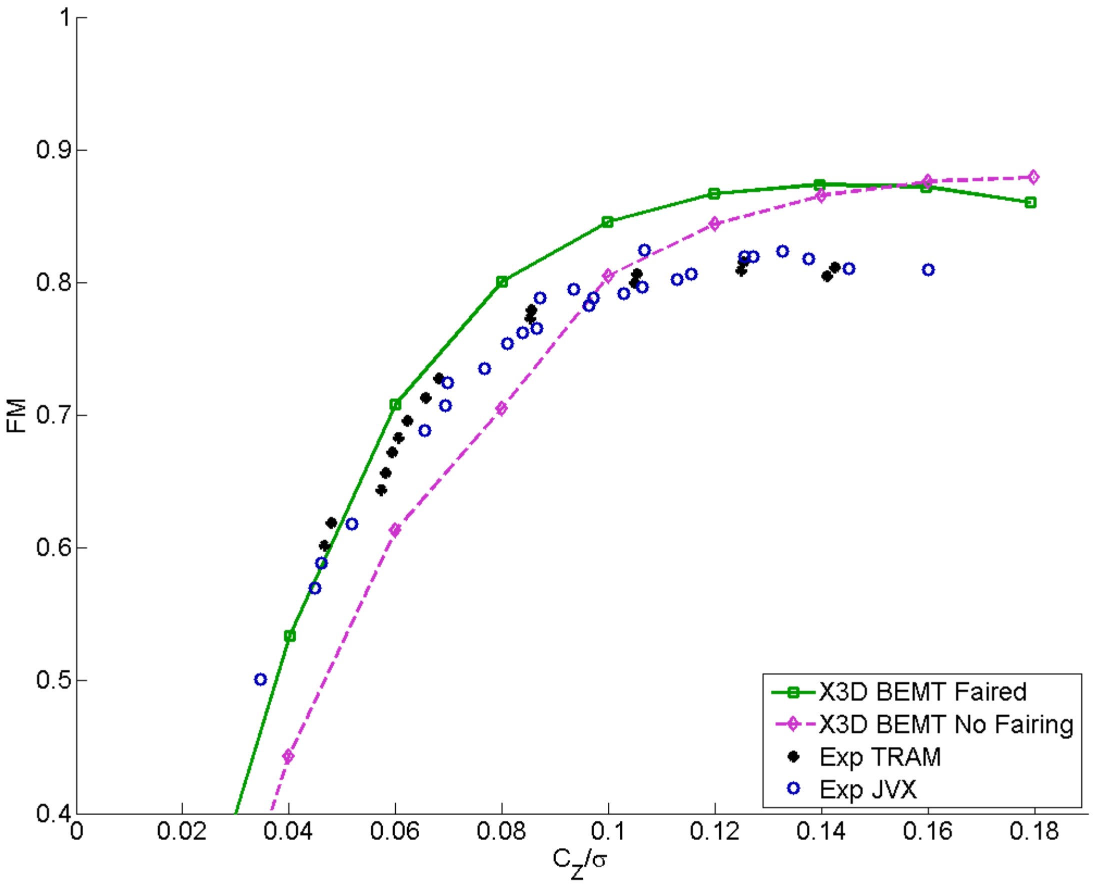


Figure 6.12 Hover performance (figure of merit) for the TRAM rotor; X3D coupled aero/structures, BEMT with and without a fairing model

6.2.2 Edgewise Flight Performance

This section examines how predictions of edgewise flight performance vary as the level of modeling is refined. It presents performance predictions using low order aerodynamic models, free-wake with a single tip vortex, and free-wake with an additional prescribed root vortex.

Figure 6.13 shows performance results, in terms of power, for the TRAM rotor in edgewise flight at two thrust conditions. Predictions are made using linear inflow aerodynamics, with and without the fairing. At both thrust levels, adding the fairing improves prediction of the power required. There is still, however, an under-prediction of power, which is worst at high thrust and high shaft tilt angles into the wind.

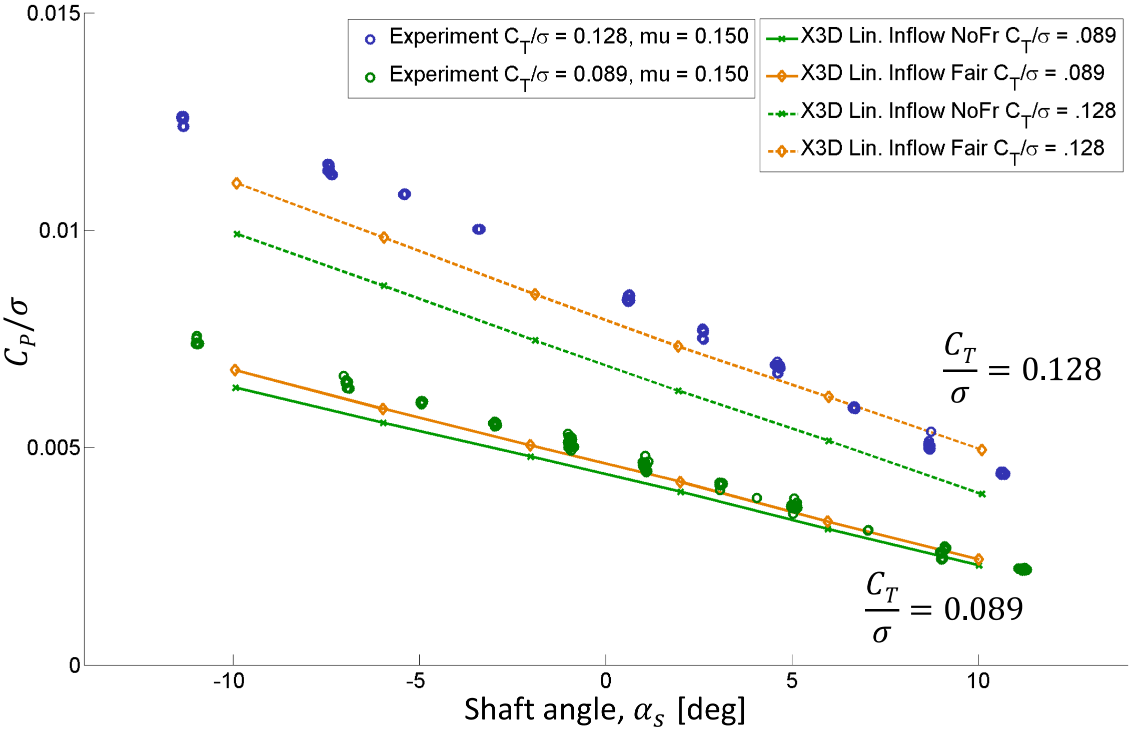


Figure 6.13 Performance prediction (power) for the TRAM rotor in edgewise flight, X3D vs experiment, linear inflow model with (orange) and without (green) the fairing aerodynamics

The propulsive force (Figure 6.14) is over-predicted for both thrusts, and is worse at high thrust. The fairing has less impact on the propulsive force than on power, but the effect it does have is detrimental, exacerbating the over-prediction.

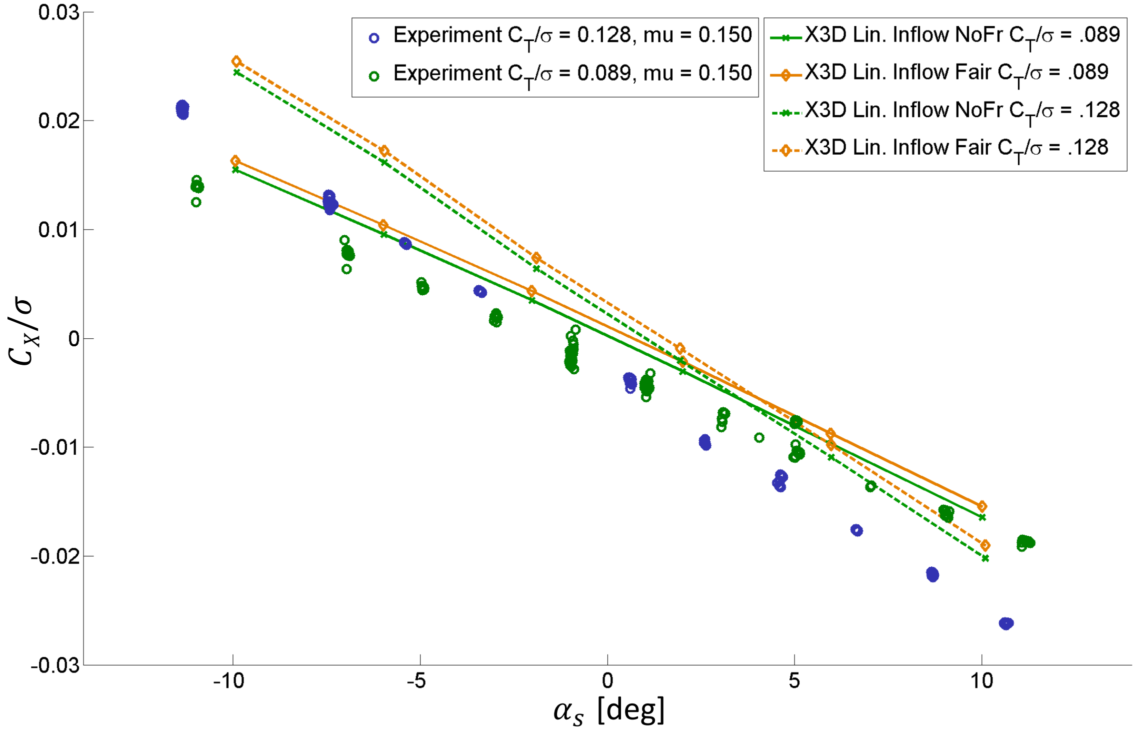


Figure 6.14 Performance prediction (propulsive force) for the TRAM rotor in edgewise flight, X3D vs experiment, linear inflow model with (orange) and without (green) the fairing aerodynamics

The effects of adding a wake model are considered next. All results shown include a fairing, and three aerodynamics models are presented: the same linear inflow model used in Figures 6.11 and 6.13, a free-wake model using a single rolled-up tip vortex, and a wake with an additional prescribed rolled-up root vortex. The geometry of the two wake models for the high thrust case at −2° shaft tilt are shown in Figures 6.15 to 6.18; the upper figures show the wake with only rolled up tip vortices, while the lower figures add a prescribed trajectory inboard vortex.

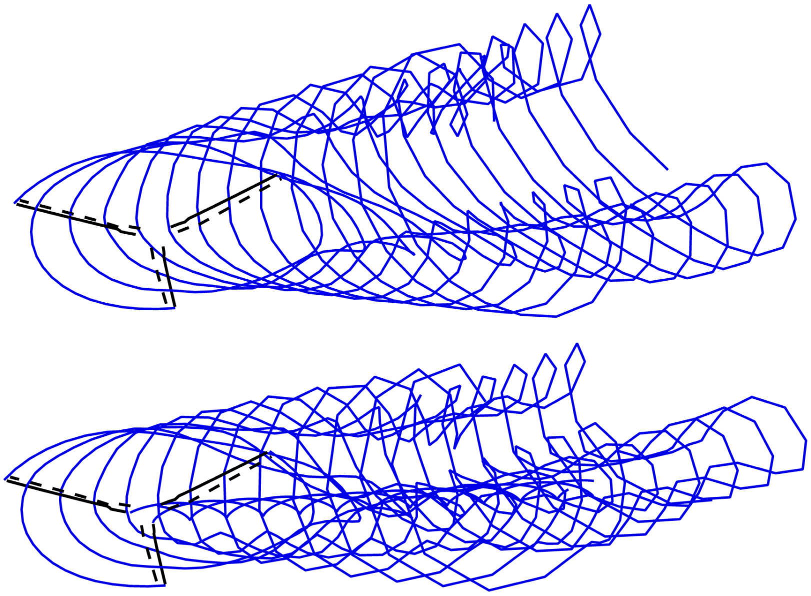


Figure 6.15 Visualization of the wake for the TRAM rotor at high thrust condition, with a free tip vortex (top) and a free tip vortex plus a prescribed root vortex (bottom), $\mu = 0.15$

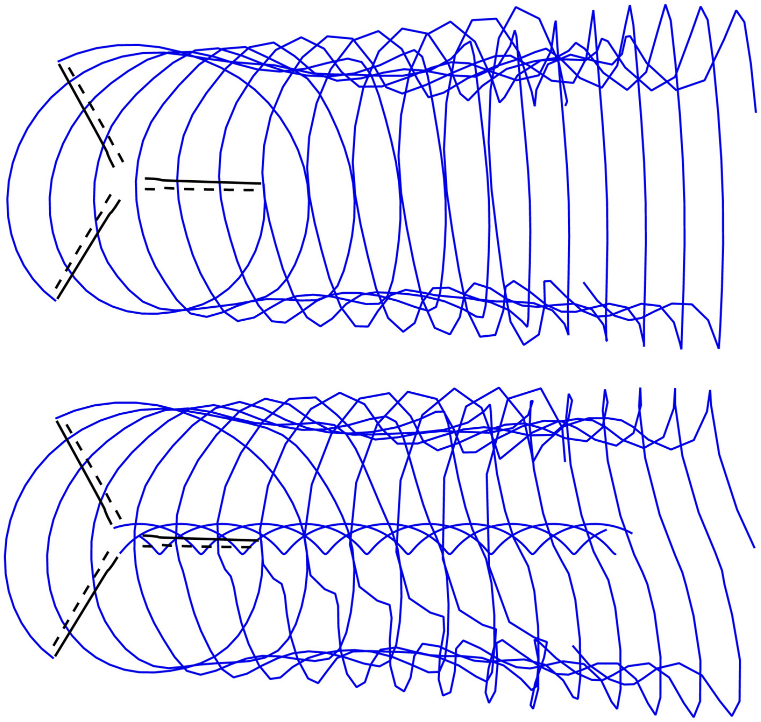


Figure 6.16 Visualization (top view) of the wake at high thrust condition, with a free tip vortex (top) and a free tip vortex plus a prescribed root vortex (bottom), $\mu = 0.15$

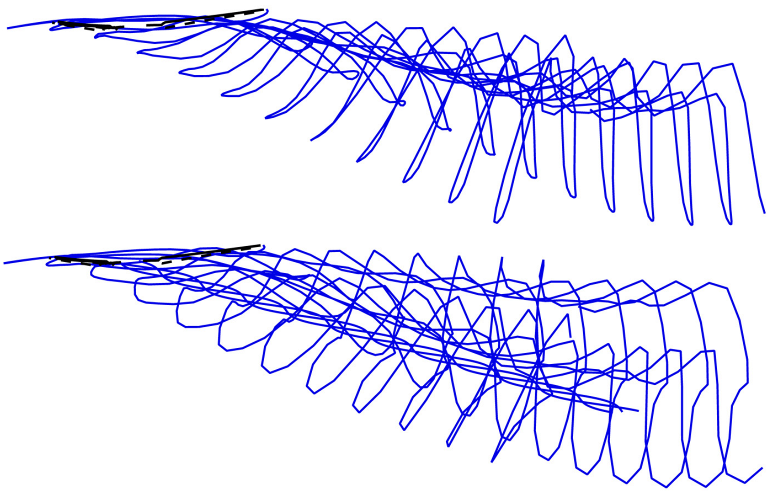


Figure 6.17 Visualization (side view) of the wake at high thrust condition, with a free tip vortex (top) and a free tip vortex plus a prescribed root vortex (bottom), $\boldsymbol{\mu = 0.15}$

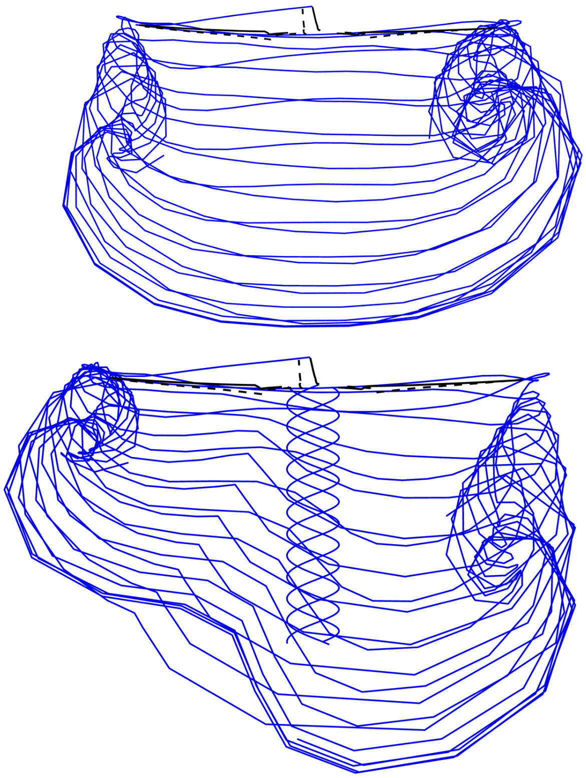


Figure 6.18 Visualization (rear view) of the wake at high thrust condition, with a free tip vortex (top) and a free tip vortex plus a prescribed root vortex (bottom), $\boldsymbol{\mu = 0.15}$

At high thrust, adding any of the wake models improves predictions of the power (Figure 6.19). At this thrust condition, the free-wake model with just the tip vortex offers a better prediction of power at negative shaft tilt angles, whereas the model with the root vortex offers more accurate results at positive shaft tilt angles. At low thrust, the addition of the wake models over-predict the power, and the simple linear inflow model actually performs better.

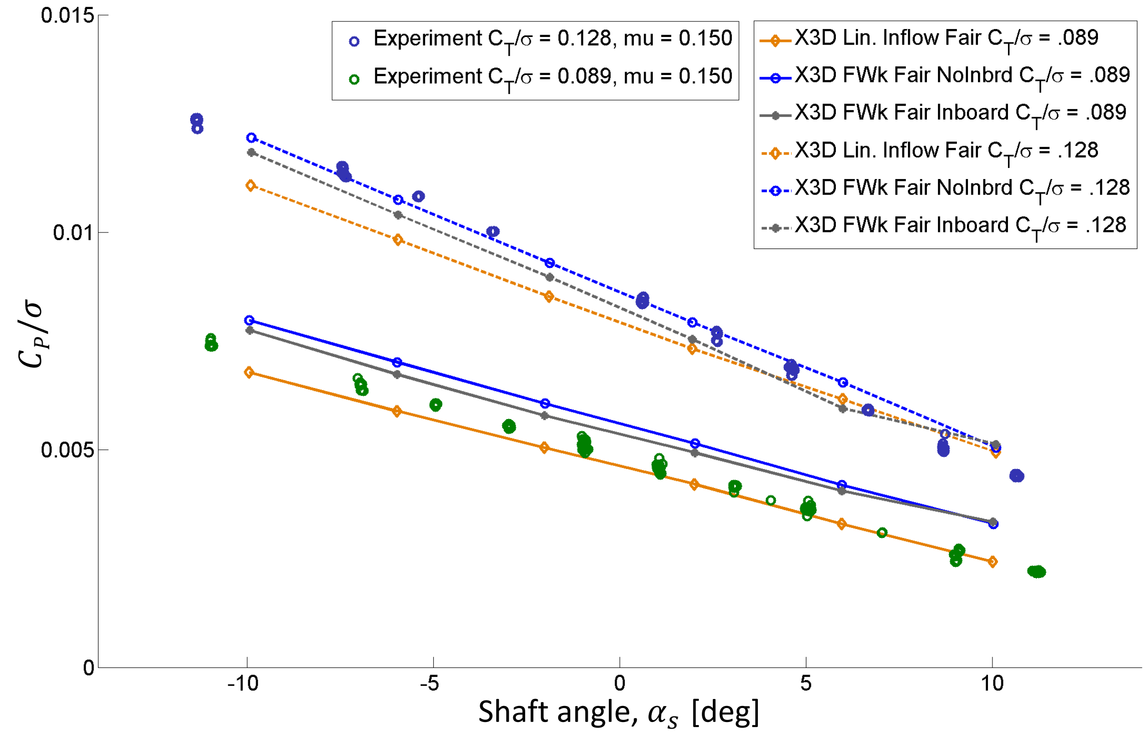


Figure 6.19 Performance prediction (power) for the TRAM rotor in edgewise flight, linear inflow model (orange), free-wake with no root vortex (blue), and free-wake with an inboard root vortex (grey)

Adding a wake model has a greater impact on propulsive force (Figure 6.20). At both thrust levels the wake model reduces the over-prediction of propulsive force at all shaft angles. As with the power results, the wake models with and without the inboard vortex have different levels of success at different shaft tilt angles. Unlike with power, however, the simpler wake with no inboard vortex performs better at high positive shaft tilt angles, and the model with the inboard vortex is best at negative shaft tilt angles. With careful tuning of the wake parameters it may be possible to get a better prediction of performance. However, the focus of this work was to move forward to using full CFD aerodynamics for an integrated 3-D analysis, the results of which were shown in Figure 5.20 and Figure 5.21.

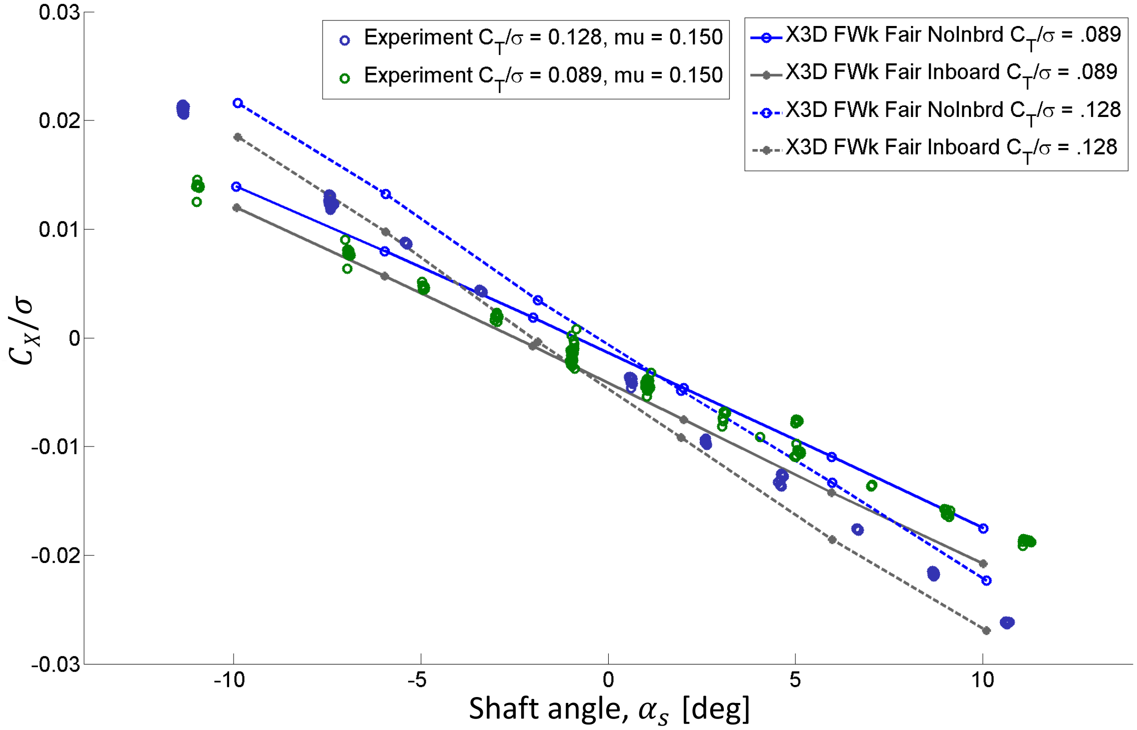


Figure 6.20 Performance prediction (propulsive force) for the TRAM rotor in edgewise flight, free-wake with no root vortex (blue), and free-wake with an inboard root vortex (grey)

Chapter 7: Conclusions

7.1 Summary

The goal of this research was to lay the ground work for using X3D for advanced rotor design. First, a methodology was developed for creating multibody, 3-D FEM structural analysis models for rotor analysis. This included establishing the process of developing 3-D CAD models, breaking them down into constituent parts, and choosing how to model each; developing best practices for meshing rotor components using 27-node solid hexahedral elements; and combining component meshes, material definitions, and joint definitions into a final model for analysis. To integrate the models with X3D, standards were established for defining mesh and joint input files, and pre-processing tools were developed to generate compatible input files. Additionally, techniques for extracting useful data for comparison to experimental measurements were developed.

Next, simple models were created to verify and validate that the methodology was compatible with X3D and capable of yielding accurate structural dynamics predictions. Deformations of isotropic beams, composite beams, and composite plates were examined, with attention paid particularly to mesh convergence. X3D prediction of beam-like sectional properties was also validated against theoretical calculations. Notional rotor geometries were tested to verify proper integration with X3D and demonstrate the ability to model joints and dampers.

Finally, a detailed, CAD-based, 3-D structural analysis model of the TRAM proprotor was developed. This proved the new methodology capable of modeling a real, advanced rotor with multiple load paths and an involved blade structure. Studies of the structural dynamics of the rotor were carried out by varying blade parameters, including materials and joint properties, demonstrating the potential of X3D as a detailed design tool. Analysis was performed using both low order aerodynamics models and CFD was used to validate X3D predictions for the TRAM model, and performance, airloads, blade loads, and dynamic 3-D stress were examined. This analysis revealed accurate predictions for airloads at two flight conditions and good capture of the trends in structural loads at low thrust. Not only was this the first ever demonstration of integrated 3-D aeromechanics analysis of a real rotor, but it was also the first coupled CFD/CSD analysis of a proprotor. Additionally, a model for simulating gimbaled rotor kinematics with a single blade was developed and implemented for rotor analysis.

7.2 Outlook

As with all analysis tools, the present modeling methodology and solver are intended to be used for the design of future rotor systems. By utilizing 3-D solid finite element structural analysis in a multibody dynamics framework, the present work offers capabilities not found elsewhere in current state-of-the-art rotorcraft aeromechanics analyses. These include the ability to model blades with structural discontinuities arising from advanced geometry, material and construction discontinuities, or ballistic damage, and to model the kinematic couplings of advanced hubs, with multiple load paths, from first principles. It predicts detailed internal stresses and strains from CFD, giving the designer greater insight into factors of safety up front in the design cycle, not only in the blade but also in 3-D components at the hub that are the key drivers of weight and maintenance requirements. A 3-D analysis also has the natural potential to be able to model morphing blades, enabling analysis of future technologies to increase rotor efficiency and performance.

Certainly, X3D faces limitations and complications, but it is believed these are all resolvable. Developing full CAD-based 3-D rotor models takes more effort up front than a legacy beam model. While this is true, modern rotor designs are being developed using CAD tools from early stages, and these models can be leveraged by the aeromechanics community for 3-D analysis. Beam models also take effort to develop and even greater effort is needed to fit beams to all parts of a rotor, typically requiring experimental measurements. This work also saw difficulties in obtaining accurate composite material properties. However, this would be no less of a problem when using an advanced beam-based analysis which relies on 2-D sectional analysis tools to generate sectional properties. Additionally, the analysis of the TRAM proprotor revealed difficulties in matching the first lag frequency, which had to be addressed by varying material and joint properties. Taken from the point of view of a design task, however, this does not demonstrate a limitation of 3-D, but instead an increased capability. It allows designers to develop a model with as much or as little detail as required: the full hub could be modeled, providing the designer the opportunity to analyze the contribution of each individual component, or a simplified single root joint could be employed, as with the single load path structural analysis model. And although 3-D analysis is computationally more expensive than legacy 1-D beam-based methods, it is predicted that advances in parallel high performance computing will alleviate the time and cost of integrated 3-D analysis. To be clear, 3-D CSD is not intended to replace existing engineering level rotor analysis methods, any more than CFD replaces lifting line aerodynamics, but to offer a high fidelity option with added scope and capabilities through computation based detailed design of advanced rotor systems.

7.3 Key Conclusions

From this research, a few specific conclusions can be drawn. In regards to the 3-D modeling methodology:

- Integrated 3-D analysis is capable of predicting aeromechanics of an advanced rotor configuration from first principles. This was demonstrated throughout the dissertation, from the accurate prediction of first flap and lag frequencies for the TRAM proprotor structural model, to first principles resolution of non-classical effects in the flow field, to internal stresses and strains from aerodynamic loading.
- Part ordering is important when establishing the topology of the multibody system in order to reduce bandwidth of the problem, and thereby minimize memory use and computational time within the solver. A three-bladed structural analysis model of the TRAM rotor required 71% more memory than the well-ordered version.
- Although capable of generating the required 27-noded hexahedral meshes, Cubit can experience errors with placement of internal nodes in higher order elements, as observed when meshing thin concave geometry with multiple hexes.
- The 3-D analysis was capable of accurately predicting deformations and internal stresses of isotropic and composite beams and plates. Mesh resolution through the thickness of composites affects the output of discontinuous stresses at ply

interfaces: the solver reports the stress at an interface as an average of both sides. Finer meshes resolve the step change in stress more accurately.

- Axial mesh size had an effect on the prediction of shear stress in the walls of a box beam undergoing vertical tip loading, with very coarse meshes leading to spurious oscillatory patterns in the stress distribution.
- An effective modulus method is capable of creating homogenized laminate material properties, enabling multiple plies to be modeled using a single element, and thereby reducing problem size. This method was validated against experimentally measured deflections of composite coupled box beams.
- Post-processing tools developed for evaluating beam-like sectional stiffness properties (EI, GJ) of a 3-D rotor blade structure was shown accurate by verifying against model problems with analytical solutions.

From integrated 3-D analysis of the TRAM rotor:

- The in-plane stiffness of the inboard centering bearing has a major impact on the first lag frequency for the TRAM model; if the in-plane degree of freedom is kept rigid the first lag frequency will be over-predicted.
- Material properties can be difficult to obtain for legacy rotors, but obviously have a strong effect on structural dynamics. Satisfactory prediction of flap bending stiffness and torsional stiffness was demonstrated for the TRAM rotor, but the chordwise bending moment was not accurately captured, which led to a negative impact on higher rotor modes.
- Hover analysis, carried out with a lower order aerodynamic model, showed an under-prediction of power (by approximately 6% at C_T/σ of 0.14) and a

corresponding over-prediction of figure of merit compared to measured values, possibly due to the use of different airfoils in the analysis than the actual model (which used proprietary airfoils).

- The more difficult edgewise flight analysis, carried out by coupled coupling X3D to full RANS CFD, showed good power predictions. However, propulsive force was not well predicted. The propulsive force was more accurately predicted by free-wake models, something that remains not fully understood.
- Accurate modeling of the airloads on the root fairing was found to be important for predictions of both hover performance and edgewise flight steady normal force distribution.
- Airloads prediction showed marked improvement as the level of aerodynamic model fidelity was increased, with Helios RANS CFD accurately predicting the first four harmonics of sectional normal force at low thrust, and the first, second, and fourth harmonics at high thrust. The third harmonic at high thrust was over-predicted.
- Non-classical effects were observed in the RANS flow field, including stall delay, complex tip vortex rollup, a large center body wake, and a blade vortex-induced dynamic stall at the blade tip under high thrust. All of these phenomena are almost impossible to capture precisely with lifting-line models.
- Structural blade loads trends were reasonably well predicted by X3D when coupled to Helios at moderate thrust. At high thrust, however, a dominant 3/rev signal appeared in measurements of flap and lag bending which was not captured by the analysis.

- The proposed gimbaled rotor model, which simulates gimbal motion with a single blade by suppressing 0/rev and 3/rev gimbal flapping in the trim solution, did not improve predictions of airloads or blade loads versus a simpler articulated-in-flap model. Accurate modeling of the 3-D flow field with CFD appears more important than accurate modeling of the gimbal motion, at least for the rotor studied here.
- The dynamic stresses at the root of the blade show complex unsteady behavior, with maximum stresses higher than anticipated. There are large unsteady stresses at the blade root, demonstrating the impact of oscillatory airloads on the rotor structure.

7.4 Recommendations for Future Work

The research performed and presented in this dissertation is the first step on the path to adoption of integrated 3-D aeromechanics analysis for rotorcraft design. Much work is still needed to insert a new tool like X3D into the regular industry workflow.

First, and most obviously, further validation is needed. The TRAM model developed for this research offers a good starting point for future researchers. Some tasks that can be performed with it include studies of the full three-bladed rotor model with CFD aerodynamics, and deeper studies of mesh size convergence for various components in the TRAM rotor. Additionally, further investigation of the material properties may yield improved correlation with test results. Beyond TRAM, further validation studies are needed. This includes validation against experimental tests of simpler model scale rotors that are being conducted as part of an ongoing research project at the University of Maryland, for which the properties are well documented. There would be even greater value in validating X3D against modern advanced rotor designs. Ideally, this would require partners in industry willing to share information about rotor design and test results. More realistically, it may require development of advanced model scale test articles, data from which would be open to the public. Mesh size and quality metrics must also be established through validation with analytical or test data in order to create guidelines for good meshes which can reliably generate accurate solutions.

Effort is also required to develop new features to improve the usability and efficiency of 3-D analysis. The most important algorithmic task is the implementation of domain decomposition schemes for parallelization of the structural solver. While X3D is currently capable of parallel analysis, the part meshes must be manually decomposed. Not only is this time consuming, but there are also no guidelines for decomposing blade meshes for maximum efficiency. Schemes for decomposing generic meshes of complex rotor geometries in an automated fashion must be developed to make X3D a practical design tool.

Currently, X3D is only capable of analyzing individual systems which are rotating or fixed, it cannot combine rotating and non-rotating subsystems into a single analysis. A rotating/non-rotating interface must be developed in order to allow analysis of entire aircraft. This interface must allow for multiple sub-systems with varying orientations, such as a tiltrotor with two rotating proprotors, mounted on two tilting pylons, attached at the tip of fixed wings. Without such an analysis, whirl-flutter studies are not currently possibly.

A level-II CFD/CSD interface, which passes information about deformations and aerodynamic forces at the node level instead of relying on existing beam-based mesh motion schemes, is desired for more robust aero/structural integration. While the framework exists for such a scheme in X3D, modifications will need to be made to CFD software tools to enable compatibility with 3-D structural dynamics.

Material modeling in X3D would also benefit from further development. Currently, an optional material homogenization technique is performed as a pre-processing step, enabling composite plies to be modeled with fewer elements. Integrating the material homogenization calculations into the X3D solver would reduce the risk of error in transferring homogenized material properties to the X3D input files. More importantly, a scheme needs to be developed to allow X3D to re-orient composite ply properties based on un-deformed mesh shapes. For example, in a twisted blade X3D cannot currently rotate the constituent material properties to follow the twist, it must be carried out by the user manually by defining separate material zones in the mesh. Automating this process would enable easier and more accurate analysis of parametric variations.

Finally, the analysis in X3D should be expanded to accommodate advanced blade design features, such as active morphing, blowing, or device actuation. The potential for such mechanisms to improve performance, reduce vibrations, or mitigate control loads has been discussed and studied using experimental models, and attempts have been made to model them using beam-based analysis. The use of full 3-D structural dynamics offers the potential to easily integrate such features, accounting for effects such as chordwise bending that are not well modeled by legacy analyses.

Designing the lighter, higher performing, vertical takeoff aircraft of the future will require new tools, and integrated 3-D aeromechanics analysis has the potential to enable such next-generation rotorcraft designs. It will, of course, take time to expand the capabilities of the analysis and achieve the level of trust necessary to see wide spread employment of these techniques. By identifying the potential and the limitations of those tools now, this research seeks to accelerate the pace of development of integrated 3-D aeromechanics, and hasten its eventual adoption by the rotor design community.

Appendix I

This appendix contains `CCBeam_17072501.jou`, a sample meshing journal file for the composite box beam shown in Figure 3.6. It is intended to illustrate the steps of the meshing process as discussed in Section 2.5.3. Note that because of the simple geometry, clean-up is not demonstrated. This mesh is attached to joints created by the joint input file provided in Appendix III and is called by the X3D structural analysis model input file found in Appendix IV.

```
#############################################################################
#############################################################################
##### Notes

# CCBeam_17072501jou, created on 07/25/2017 from "CCBeam_1701602.jou"

# Generates an updated mesh for the Chandra Chopra box beam with variable
# elements/ply to test mesh resolution effects on stress predictions

# Beam consists of two vertical webs capped by two horizontal caps
# (caps overhang spars to be flush with outside of beam)

# This journal will create the geometry and mesh


### Notes from "Structural Response of Composite Beams and Blades
### with Elastic Couplings," Chandra, Chopra, 1992 [UMD hard copy]

#   E1 = 20.59E6 psi
#   E2 =  1.42E6 psi
#  G12 =  0.89E6 psi
# nu12 =  0.42

# Length           = 36 in.
# Clamped Length   = 6 in.
# Effective Length = 30 in.
# Wall thickness   = 0.03 in.
# Number of Layers = 6
# Ply thickness    = 0.005 in.
# Outer width      = 0.953 in.
# Outer depth      = 0.537 in. (handwritten over crossed out 0.37 in.)



#############################################################################
```

```
#############################################################################
##### Initialization

### Reset Cubit to clear old geometry
reset


#####################################
##### Dimensions

### Overall dimensions
$ {Hgt_O =  0.537*25.4}  # Outisde height of the beam [mm]   # Hgt_O = 30 #
Outisde height of the beam [mm] EXAG
$ {Wid_O =  0.953*25.4}  # Outside width of the beam [mm]    # Wid_O = 45 #
Outside width of the beam [mm] EXAG
$ {Lng   = 30.000*25.4}  # Length of the beam [mm]


### Member thicknesses
$ {th_cap = 0.030*25.4} # Thickness, box caps [mm]    # th_cap = 5 # EXAG
$ {th_web = 0.030*25.4} # Thickness, box webs [mm]

### Inside channel dimensions (calculation)
$ {Hgt_I = Hgt_O - th_cap - th_cap} # Inside height of the beam [mm]
$ {Wid_I = Wid_I - th_web - th_web} # Inside width  of the beam [mm]



#####################################
##### Mesh

### Set mesh size
$ {N_Plies = 6}  # Number of plies in the caps/webs
$ {N_ElPly = 3}  # Number of elements per ply  (1, 2, or 3)
$ {N_ElLng = 20} # Number of elements along the length of the beam (Originally
10, also tested 30, 60)

### Calculate total elements through thickness
$ {N_ElThk = N_Plies * N_ElPly}   # Number of elements through cap/web
thickness

### Set tolerance for automated element selection
$ {tol_x = Lng/N_ElLng/4}
$ {tol_z = th_cap/N_ElThk/4}
$ {tol_y = th_web/N_ElThk/4}



#############################################################################
#############################################################################
##### Create Geometry

### Create three rectangular solids, then shift into position

## Top spar cap
```

```
brick x {Lng} y {Wid_O} z {th_cap}
volume 1 rename "Cap_Top"
move volume 1 x {Lng/2} z {(Hgt_O - th_cap)/2}

## Bottom spar cap
brick x {Lng} y {Wid_O} z {th_cap}
volume 2 rename "Cap_Bot"
move volume 2 x {Lng/2} z {-(Hgt_O - th_cap)/2}

## Positive-Y spar web
brick x {Lng} y {th_web} z {Hgt_I}
volume 3 rename "Web_PosY"
move volume 3 x {Lng/2} y {(Wid_O - th_web)/2} z {(th_cap - th_cap)/2}

## Negative-Y spar web
brick x {Lng} y {th_web} z {Hgt_I}
volume 4 rename "Web_NegY"
move volume 4 x {Lng/2} y {-(Wid_O - th_web)/2} z {(th_cap - th_cap)/2}



#############################################################################
#############################################################################
##### Decompose Geometry

### Use inside walls of boxbeam to separate corners
webcut volume 1 2  with plane surface 15
webcut volume 5 6  with plane surface 23


### Split corners on their diagonals (ONLY IF N_ElThk > 1)
${If(N_ElThk > 1)}
  webcut volume 1  with plane vertex 38  vertex  3  vertex  2  # Upper Fwd
(+Y)
  webcut volume 5  with plane vertex  4  vertex 27  vertex  1  # Upper Aft (-
Y)
  webcut volume 2  with plane vertex 47  vertex 16  vertex 13  # Lower Fwd
(+Y)
  webcut volume 6  with plane vertex 15  vertex 61  vertex 14  # Lower Aft (-
Y)
${EndIf}



### Imprint and merge geometry
imprint all
merge all

### Set colors
color Volume 8 lightseagreen
color Volume 6 blueviolet
color Volume 11 lawngreen
```

```
##############################################################################
##############################################################################
##### Mesh Mesh Geometry

##### Mesh root surfaces

### Set intervals along inside of walls/caps
curve 93 33 111 47 95 35 109 45  interval 2

### If N_ElThk > 1, corners are split into triangles; use this section
${If(N_ElThk > 1)}
  curve        52 27  81 39 43 97  31 68 113 119 126 131  interval {N_ElThk}
# Set intervals through wall thickness
  mesh Surface 16 22 52 62 66 70 72 76 79 81 84 88
# Mesh root surfaces

### If N_ElThk == 1, corners are square; use this section
${Else}
  curve        52 27 81 39 43 97 31 68  interval {N_ElThk}  # Set intervals
through wall thickness
  mesh Surface 16 22 52 62 29 39 52 58                      # Mesh root
surfaces
${EndIf}



##### Mesh beam volumes
curve 51 2 8 65 20 14 30 28  interval {N_ElLng}      # Set intervals along
length of beam
mesh volume all


##############################################################################
##############################################################################
##### Define Blocks, Nodesets, Sidesets

#####################################
##### Define Blocks
# Note: Each ply is its own block: 6 plies = 24 blocks (6 top, bottom, fwd,
aft)
# Block 201-206  =  Top cap
# Block 207-212  =  Bottom cap
# Block 213-218  =  Forward (+ve Y) cap
# Block 219-224  =  Aft (-ve Y) cap
# For each member, loop over number of layers, assign blocks based on Z or Y
coordinates




### If N_ElThk > 1, assign blocks using loops below

${If(N_ElThk > 1)}
```

```
### Assign first block ID
$ {Blck_ID = 201} # Assign first block number


### Loop over elements in Top Spar Cap (Volumes 7 9 10)
$ {jj = 1}         # Initialize loop index
$ {Loop(N_ElThk)}  # Loop over elements through member

  ### Calculate the coordinate of the center of the element layer
  $ {Elem_Z = (Hgt_O/2 - (jj-1/2)*th_cap/N_ElThk)}

  ### Add elements with proper coordinates to block
  block {Blck_ID} hex in volume 7 9 10 with (z_coord < {Elem_Z + tol_z}) and
(z_coord > {Elem_Z - tol_z})

  ### Check modulus of the hex index: if the number of elements is evenly
divisible by the ply per element, advance the block ID
  $ {if(jj%N_ElPly == 0)}
    $ {Blck_ID++} # Advance block ID
  ${EndIf}

  $ {jj++}  # Advance layer counter jj
$ {EndLoop}



### Loop over elements in Bottom Spar Cap (Volumes 2, 6, 8)
$ {jj = 1}         # Initialize loop index
$ {Loop(N_ElThk)}  # Loop over elements through member

  ### Calculate the coordinate of the center of the element layer
  $ {Elem_Z = -(Hgt_O/2 - (jj-1/2)*th_cap/N_ElThk)}

  ### Add elements with proper coordinates to block
  block {Blck_ID} hex in volume 2 6 8 with (z_coord < {Elem_Z + tol_z}) and
(z_coord > {Elem_Z - tol_z})

  ### Check modulus of the hex index: if the number of elements is evenly
divisible by the ply per element, advance the block ID
  $ {if(jj%N_ElPly == 0)}
    $ {Blck_ID++} # Advance block ID
  ${EndIf}

  $ {jj++}  # Advance layer counter jj
$ {EndLoop}




### Loop over elements in Forward (Positive-Y) Web (Volumes 1 3 11)
$ {jj = 1}         # Initialize loop index
$ {Loop(N_ElThk)}  # Loop over elements through member

  ### Calculate the coordinate of the center of the element layer
  $ {Elem_Y = (Wid_O/2 - (jj-1/2)*th_web/N_ElThk)}

  ### Add elements with proper coordinates to block
  block {Blck_ID} hex in volume 1 3 11 with (y_coord < {Elem_Y + tol_y}) and
(y_coord > {Elem_Y - tol_y})

  ### Check modulus of the hex index: if the number of elements is evenly
divisible by the ply per element, advance the block ID
  $ {if(jj%N_ElPly == 0)}
    $ {Blck_ID++} # Advance block ID
  ${EndIf}

  $ {jj++}  # Advance layer counter jj
$ {EndLoop}




### Loop over elements in Aft (Negative-Y) Web (Volumes 4 5 12)
$ {jj = 1}         # Initialize loop index
$ {Loop(N_ElThk)}  # Loop over elements through member

  ### Calculate the coordinate of the center of the element layer
  $ {Elem_Y = -(Wid_O/2 - (jj-1/2)*th_web/N_ElThk)}

  ### Add elements with proper coordinates to block
  block {Blck_ID} hex in volume 4 5 12 with (y_coord < {Elem_Y + tol_y}) and
(y_coord > {Elem_Y - tol_y})

  ### Check modulus of the hex index: if the number of elements is evenly
divisible by the ply per element, advance the block ID
  $ {if(jj%N_ElPly == 0)}
    $ {Blck_ID++} # Advance block ID
  ${EndIf}

  $ {jj++}  # Advance layer counter jj
$ {EndLoop}



${Else}  # If N_ElThk == 1, just add corners to cap
  block 201 volume 7 5 1  # Top Cap
  block 202 volume 8 6 2  # Bot Cap
  block 203 volume 3      # Fwd Web
  block 204 volume 4      # Aft Web

${EndIf}  # End "If N_ElThk > 1"






### Assign block materials
# Note: Assumes 6 plies
# If fewer, names won't match but no error

create material name 'Cap_Top_1'
create material name 'Cap_Top_2'
create material name 'Cap_Top_3'
create material name 'Cap_Top_4'
create material name 'Cap_Top_5'
create material name 'Cap_Top_6'
create material name 'Cap_Bot_1'
create material name 'Cap_Bot_2'
create material name 'Cap_Bot_3'
create material name 'Cap_Bot_4'
create material name 'Cap_Bot_5'
create material name 'Cap_Bot_6'
create material name 'Cap_Fwd_1'
create material name 'Cap_Fwd_2'
create material name 'Cap_Fwd_3'
create material name 'Cap_Fwd_4'
create material name 'Cap_Fwd_5'
create material name 'Cap_Fwd_6'
create material name 'Cap_Aft_1'
create material name 'Cap_Aft_2'
create material name 'Cap_Aft_3'
create material name 'Cap_Aft_4'
create material name 'Cap_Aft_5'
create material name 'Cap_Aft_6'

block 201 material 'Cap_Top_1'
block 202 material 'Cap_Top_2'
block 203 material 'Cap_Top_3'
block 204 material 'Cap_Top_4'
block 205 material 'Cap_Top_5'
block 206 material 'Cap_Top_6'
block 207 material 'Cap_Bot_1'
block 208 material 'Cap_Bot_2'
block 209 material 'Cap_Bot_3'
block 210 material 'Cap_Bot_4'
block 211 material 'Cap_Bot_5'
block 212 material 'Cap_Bot_6'
block 213 material 'Cap_Fwd_1'
block 214 material 'Cap_Fwd_2'
block 215 material 'Cap_Fwd_3'
block 216 material 'Cap_Fwd_4'
block 217 material 'Cap_Fwd_5'
block 218 material 'Cap_Fwd_6'
block 219 material 'Cap_Aft_1'
block 220 material 'Cap_Aft_2'
block 221 material 'Cap_Aft_3'
block 222 material 'Cap_Aft_4'
block 223 material 'Cap_Aft_5'
block 224 material 'Cap_Aft_6'

### Set element type
block all element type hex27
```

```
### Color blocks
color block 203 darkslateblue
color block 213 magenta
color block 216 orangered
color block 217 green
color block 214 gold
color block 207 blue



### Set element type
block all element type hex27


#####################################
##### Sidesets

#################
#### Series SS300 (Standard SSets + Boundary + Interfaces)
### Boundary surface sideset    # [300 => BOUNDARY]
# No boundary sideset

### Wet surface sideset   # [301 => AERO INTERFACE]
# No wet surface sideset


#################
#### Series SS500 (Stress/Strain Cross Section Markers)
# No CS sidesets


#################
#### Series SS600 (Surfaces for Loading)
# No 600 series sidesets


#################
#### Series SS900 (Aero Segments)
# No 900 series sidesets






#####################################
##### Nodesets

#################
#### Series NS400 (Standard Nodesets + Boundary)

### Boundary Nodes -> Root surface
# No NS400 boundary nodeset
```

```
### Joint Connections
nodeset 401 node in surface with x_coord == 0      # Root surface for clamp
joint
nodeset 402 node in surface with x_coord == {Lng}  # Tip surface for loading
joint


### Other NSets
# Creating nodesets of lines through member thickness at midpoint of top cap
and LE web
nodeset 451 node in volume 7 with (x_coord <= {Lng/2 + tol_x}) and (x_coord >=
{Lng/2 - tol_x}) and (y_coord <= {tol_y}) and (y_coord >= {-tol_y})
nodeset 452 node in volume 3 with (x_coord <= {Lng/2 + tol_x}) and (x_coord >=
{Lng/2 - tol_x}) and (z_coord <= {tol_z}) and (z_coord >= {-tol_z})



#################
#### Series NS500 (Stress/Strain Cross Section Markers)
# No 500 series nodesets


#################
#### Series NS600 (Surfaces for Loading)
# No 600 series NSets




#################
#### Series NS700 (Deformation Cross-sections CN)

### Set coordinates for LE, TE, US, BS
$ {y_LE =  Wid_O/2}    # Y-Coordinate of LE thickness line
$ {z_LE =  0}               # Z-Coordinate of LE thickness line

$ {y_TE =  -Wid_O/2}  # Y-Coordinate of TE thickness line
$ {z_TE =  0}               # Z-Coordinate of TE thickness line$

$ {y_t_US =  0}                # Y-Coordinate of top thickness line
$ {z_t_US =  Hgt_O/2}    # Z-Coordinate of top thickness line

$ {y_t_BS =  0}               # Y-Coordinate of bottom thickness line
$ {z_t_BS =  -Hgt_O/2}  # Z-Coordinate of bottom thickness line


### NS 701: Leading Edge of cross-sections
$ {yy = y_LE} {zz = z_LE}
nodeset  701 node with (y_coord < {yy + tol_y}) and (y_coord > {yy - tol_y})
and (z_coord < {zz + tol_z}) and (z_coord > {zz - tol_z})

### NS 702: Trailing Edge of cross-sections
$ {yy = y_TE} {zz = z_TE}
```

```
nodeset  702 node with (y_coord < {yy + tol_y}) and (y_coord > {yy - tol_y})
and (z_coord < {zz + tol_z}) and (z_coord > {zz - tol_z})

### NS 703: Upper Surface of cross-sections
$ {yy = y_t_US} {zz = z_t_US}
nodeset  703 node with (y_coord < {yy + tol_y}) and (y_coord > {yy - tol_y})
and (z_coord < {zz + tol_z}) and (z_coord > {zz - tol_z})

### NS 704: Bottom Surface of cross-sections
$ {yy = y_t_BS} {zz = z_t_BS}
nodeset  704  node with (y_coord < {yy + tol_y}) and (y_coord > {yy - tol_y})
and (z_coord < {zz + tol_z}) and (z_coord > {zz - tol_z})


#################
#### Series NS800 (Structural Loads Cross-sections SC)
# Going to use CNLE/TE/US/BS at every radial station
nodeset 801 node in nodeset 701
nodeset 802 node in nodeset 702
nodeset 803 node in nodeset 703
nodeset 804 node in nodeset 704



#################
#### Series NS900 (Aero Cross-sections AE)
# No 900 series NSets



#################
#### Series NS600 (Surfaces for Loading)
# No 600 series NSets




##############################################################################
##############################################################################
##### Finalize and Export

##### Export
### Export box beam w/ 2 elem webs/caps, 10 elem long, varying thickness
# export Ideas "CCBeam_1x1x20_17072501.unv"  #  240 elements   2952 nodes
# export Ideas "CCBeam_2x1x20_17072501.unv"  #  800 elements   7872 nodes
# export Ideas "CCBeam_3x1x20_17072501.unv"  # 1280 elements   1488 nodes
# export Ideas "CCBeam_6x1x20_17072501.unv"  # 2720 elements  23616 nodes
# export Ideas "CCBeam_6x2x20_17072501.unv"  # 5600 elements  47232 nodes
# export Ideas "CCBeam_6x3x20_17072501.unv"  # 8480 elements  70848 nodes
```

Appendix II

This appendix contains a sample joint input for the notional bearingless rotor model introduced in Figure 2.7. This joint input file “`BLess_Joint_In_16042101.m`” is referenced in Section 2.6.2.2.1: Joint Definition Input File Format.

```
%%%%% INPUT FILE FOR JOINT DEFINITION FUNCTION "JointDef.m"
% BLess_Joint_In_16042101.m
% This input file is for a bearingless rotor blade + torque tube + flexure
% It has been updated to the latest standard and is intended as an example for
release
% William Staruk
% 04/21/2016

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Joint Output Details
% This section contains inputs that are common to all joints

%%% Joint file output location and name
% Use directory in which this joint input file is found [typical]
JFDir = [pwd,'/'];

% % Enter other directory in which to write the joints [optional]
% JFDir = 'C:/Users/WStaruk/Documents/WStaruk Documents/UMD/Research/Papers
and Presentations/Papers/2016-03 CMARS Final Report/Figures/Sample Meshes
Joints/BLess SAM/';


%%% Define Master Header
% The master header will be printed to the header of every joint file
% Each joint will also get a specific header line, printed in JHeadLine

% Get time/date processor is being run
timestamp = now;

% Set master header
header{ 1} = ['! ------    Joint file created on ',datestr(timestamp),'    ---
---'];
header{ 2} = '! Bearingless rotor with tube spar blade, flexbeam, tqtube,
PLink';
header{ 3} = '!   Updated version of "BLess_SqrDamp"';
header{ 4} = '! Designed to demonstrated latest joint standards with old
mesh';
header{ 5} = '! Intended for release';
header{ 6} = '! William Staruk, 04/21/2016';
header{ 7} = '!';
header{ 8} = '!';
header{ 9} = '!';
```

```
header{10} = '! --------------------------------------------------------------
---------';

% Set line of header to overwrite with custom header line for each joint
JHeadLine = 8;


%%% Specify Version Tag
% Use this section to append a version tag to each joint file
% This tag can be deleted when running the solver if desired
ver_tag = '_16042101';


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Topological Information
% Contains information on the parts within the SAM


%%% Total number of bodies to be joined
nBod_Total = 4; % Blade, Flexure, Torque Tube, Pitch Link

%%% Number of joints to create
nJoint = 6; % 2 blade bolts, lag damper, hub joint, PLink top, PLink bottom

%%% Number of nodes per element
HexType = 27; % NOTE: Only Hex27 meshes supported

%%% Mesh dimensionality
nDimensions = 3; % NOTE: Only 3-D meshes supported


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Flex Part Meshes

%%% Set parent directory in which to find flex part meshes files
% Use directory in which this joint input file is found [typical]
MFDir = [pwd,'/'];

% % Enter other directory in which to write the joints [optional]
% MFDir = 'C:/Users/WStaruk/Documents/WStaruk Documents/UMD/Research/Papers
and Presentations/Papers/2016-03 CMARS Final Report/Figures/Sample Meshes
Joints/BLess SAM/';


%%% Names of part mesh data files, names of flex parts, and global part IDs
% Include sub-directories, do not put .dat extension

% Flex Part 1: Blade
MFNames{1}    = 'Blade_BLess_16040501'; % Mesh file name
FlexNames{1}  = 'Blade';                % Shorthand name of the part for quick
reference
SAR_PartID(1) = 1;                      % The part ID in the global
(component) SAR

% Flex Part 2: Flexbeam
```

```
MFNames{2}    = 'Flex_BLess_16042101'; % Mesh file name
FlexNames{2}  = 'Flexbeam';            % Shorthand name of the part for quick
reference
SAR_PartID(2) = 4;                     % The part ID in the global (component)
SAR

% Flex Part 3: Torque Tube
MFNames{3}    = 'TqTube_BLess_16042101'; % Mesh file name
FlexNames{3}  = 'Torque Tube';           % Shorthand name of the part for
quick reference
SAR_PartID(3) = 5;                       % The part ID in the global
(component) SAR

% Flex Part 4: Pitch Link
MFNames{4}    = 'PLink_15031601'; % Mesh file name
FlexNames{4}  = 'Pitch Link';     % Shorthand name of the part for quick
reference
SAR_PartID(4) = 6;                % The part ID in the global (component) SAR


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Joint Definitions
% Define joints using nodeset names


%%%% NOTE: FLEX PART NUMBERS BELOW MATCH INPUT INDEX ABOVE, NOT SAR_PartID

%%% Joint 1: LE (-Y) blade bolt connecting Blade, TqTube, and Flexure
J_ID = 1;

JFNames{J_ID} = 'J_BladeBoltsLE';  % Output name for this joint's ".j.dat"
file
J_Desc{J_ID}  = 'LE Blade Bolts';  % Brief description of the part for
outputting to joint file
JHead{J_ID,1} = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

nJBods(J_ID)  = 3;             % Number of bodies (flex parts) connected by
joint J_ID
J_Type(J_ID)  = 1;             % Joint Type (1 = std, 0 = boundary)
J_Dof{J_ID}   = [0 0 0 0 0 0]; % Joint DoFs [x y z thx thy thz]

J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of nodesets in
joint 1
J_NSets{J_ID}{1} = {1,'NS401'};          % (Part #, nodeset names) (Blade)
J_NSets{J_ID}{2} = {2,'NS401'};          % (Part #, nodeset names) (Flexure)
J_NSets{J_ID}{3} = {3,'NS401'};          % (Part #, nodeset names) (TqTube)


%%% Joint 2: TE blade bolt connecting Blade and TqTube
J_ID = 2;

JFNames{J_ID} = 'J_BladeBoltsTE';  % Output name for this joint's ".j.dat"
file
```

```
J_Desc{J_ID}  = 'TE Blade Bolts';  % Brief description of the part for
outputting to joint file
JHead{J_ID,1} = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

nJBods(J_ID)  = 3;             % Number of bodies (flex parts) connected by
joint J_ID
J_Type(J_ID)  = 1;             % Joint Type (1 = std, 0 = boundary)
J_Dof{J_ID}   = [0 0 0 0 0 0]; % Joint DoFs [x y z thx thy thz]

J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of nodesets in
joint 1
J_NSets{J_ID}{1} = {1,'NS402'};          % (Part #, nodeset names) (Blade)
J_NSets{J_ID}{2} = {2,'NS402'};          % (Part #, nodeset names) (Flexure)
J_NSets{J_ID}{3} = {3,'NS402'};          % (Part #, nodeset names) (TqTube)


%%% Joint 3: Lag Damper between TqTube and Flexure
J_ID = 3;

JFNames{J_ID} = 'J_LagDamp';       % Output name for this joint's ".j.dat"
file
J_Desc{J_ID}  = 'Lag Damp Joint';  % Brief description of the part for
outputting to joint file
JHead{J_ID,1} = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

nJBods(J_ID)  = 2;             % Number of bodies (flex parts) connected by
joint J_ID
J_Type(J_ID)  = 1;             % Joint Type (1 = std, 0 = boundary)
J_Dof{J_ID}   = [1 1 0 1 1 1]; % Joint DoFs [x y z thx thy thz]

J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of nodesets in
joint 1
J_NSets{J_ID}{1} = {2,'NS403'};          % (Part #, nodeset names) (Flexure)
J_NSets{J_ID}{2} = {3,'NS403'};          % (Part #, nodeset names) (TqTube)


%%% Joint 4: Hub bolts fixing the flexure (BOUNDARY)
J_ID = 4;

JFNames{J_ID} = 'J_Hub';        % Output name for this joint's ".j.dat" file
J_Desc{J_ID}  = 'Hub Joint';  % Brief description of the part for outputting
to joint file
JHead{J_ID,1} = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

nJBods(J_ID)  = 1;             % Number of bodies (flex parts) connected by
joint J_ID
J_Type(J_ID)  = 0;             % Joint Type (1 = std, 0 = boundary)
J_Dof{J_ID}   = [0 0 0 0 0 0]; % Joint DoFs [x y z thx thy thz]

J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of nodesets in
joint 1
J_NSets{J_ID}{1} = {2,'NS404'};          % (Part #, nodeset names) (Flexure)
```

```
%%% Joint 5: Torque tube connection to the pitchlink
J_ID = 5;

JFNames{J_ID} = 'J_PlinkTop'; % Output name for this joint's ".j.dat" file
J_Desc{J_ID}  = 'PLink Top';  % Brief description of the part for outputting
to joint file
JHead{J_ID,1} = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

nJBods(J_ID)  = 2;             % Number of bodies (flex parts) connected by
joint J_ID
J_Type(J_ID)  = 1;             % Joint Type (1 = std, 0 = boundary)
J_Dof{J_ID}   = [0 0 0 1 1 1]; % Joint DoFs [x y z thx thy thz]

J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of nodesets in
joint 1
J_NSets{J_ID}{1} = {3,'NS404'};          % (Part #, nodeset names) (TqTube)
J_NSets{J_ID}{2} = {4,'NS401'};          % (Part #, nodeset names) (PLink)


%%% Joint 6: Pitch link connection to swash plate (BOUNDARY)
J_ID = 6;

JFNames{J_ID} = 'J_PlinkBot';    % Output name for this joint's ".j.dat" file
J_Desc{J_ID}  = 'PLink Bottom';  % Brief description of the part for
outputting to joint file
JHead{J_ID,1} = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

nJBods(J_ID)  = 1;             % Number of bodies (flex parts) connected by
joint J_ID
J_Type(J_ID)  = 0;             % Joint Type (1 = std, 0 = boundary)
J_Dof{J_ID}   = [0 0 1 1 1 1]; % Joint DoFs [x y z thx thy thz]

J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of nodesets in
joint 1
J_NSets{J_ID}{1} = {4,'NS402'};          % (Part #, nodeset names) (PLink)

%%%% Append the version tag to each to each joint file output name
for jj = 1:nJoint
    JFNames{jj} = [JFNames{jj},ver_tag];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Error Checks

%%%%% Check to make sure correct number of inputs were provided

%%% Ensure the correct number of meshes were provided
if length(MFNames) ~= nBod_Total
    fprintf(2,'ERROR: INCORRECT NUMBER OF MESH FILES PROVIDED\r\n');
    return
```

```
end

%%% Ensure the correct number of joints were provided
% Check J_Type
if length(J_Type) ~= nJoint
    fprintf(2,'ERROR: NOT ALL J_Type DEFINED\r\n');
    return
end
% Check J_Desc
if length(J_Desc) ~= nJoint
    fprintf(2,'ERROR: NOT ENOUGH JOINT DESCRIPTIONS PROVIDED\r\n');
    return
end
% Check J_Nsets and nJBods
if length(nJBods) ~= nJoint || length(J_NSets) ~= nJoint
    fprintf(2,'ERROR: INSUFFICIENT NUMBER OF JOINTS DEFINED\r\n');
    return
end
% Check J_DoF
if length(J_Dof) ~= nJoint
    fprintf(2,'ERROR: MISSING JOINT DoFs\r\n');
    return
end
```

Appendix III

This appendix contains `CCBeam_17072501_JointIn.m`, a sample joint input file for a beam undergoing static tip loading. It follows the convention defined in Section 2.6.2.2.1. Two joints are created, one attached to root nodes and one attached to the tip. The mesh the joints are associated with is the box beam generated by the journal file in Appendix I, and they are called by the X3D structural analysis model input file found in Appendix IV.

```
%%%%% INPUT FILE FOR JOINT DEFINITION FUNCTION "JointDef.m"
% This input file is for a Chandra-Chopra composite box beam undergoing
%   static tip deflection
%
% William Staruk
% 07/25/2017



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Joint Output Details
% This section contains inputs that are common to all joints

%%% Joint file output location and name
% Use directory in which this joint input file is found [typical]
JFDir = [pwd,'/'];

% % Enter other directory in which to write the joints [optional]
% JFDir = 'C:/Users/WStaruk/Documents/WStaruk Documents/UMD/Research/Other/Box
Beam SAM/CC Beam SAM Update 17010601/For Eliz 170725/';



%%% Define Master Header
% The master header will be printed to the header of every joint file
% Each joint will also get a specific header line, printed in JHeadLine

% Get time/date processor is being run
timestamp = now;

header{ 1} = ['! ------    Joint file created on ',datestr(timestamp),'    ---
---'];
header{ 2} = '! Chandra-Chopra box beam for static deflection';
header{ 3} = '!   Uses mesh "CCBeam_17072501_6x1/2/3x20"';
header{ 4} = '!     1/2/3/6/12/18 elements across thickness';
```

```
header{ 5} = '!     2 elements across inside of caps, walls';
header{ 6} = '!     20 elements long';
header{ 7} = '! William Staruk, 7/25/2017';
header{ 8} = '!';
header{ 9} = '!';
header{10} = '! --------------------------------------------------------------
---------';

JHeadLine = 9; % Line of header to overwrite with custom header line for each
joint



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Version Tag
% Use this section to append a version tag to each joint file
% This tag can be deleted when running the solver

ver_tag = '_17072501';  % Joints created on 97/25/2017, iteration 01



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Topological Information
% Contains information on the parts within the SAM

%%% Total number of bodies to be joined
nBod_Total = 1;  % 1 beam

%%% Number of joints to create
nJoint = 2;      % 1 root clamp, 1 tip loading

%%% Number of nodes per element
HexType = 27; % NOTE: Only hex27 meshes supported

%%% Mesh dimensionality
nDimensions = 3; % NOTE: Only 3-D meshes supported



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Flex Part Meshes

%%% Set parent directory in which to find flex part meshes files
% Use directory in which this joint input file is found [typical]
MFDir = [pwd,'/'];

% % Enter other directory in which to write the joints [optional]
% MFDir = 'C:/Users/WStaruk/Documents/WStaruk Documents/UMD/Research/Other/Box
Beam SAM/CC Beam SAM Update 17010601/For Eliz 170725/';
```

```
%%% Names of part mesh data files, names of flex parts, and global part IDs
% Include sub-directories, do not put .dat extension

% Flex Part 1: Beam
MFNames{1}    = 'CCBeam_1x1x20_17072501'; % Tube beam mesh
FlexNames{1}  = 'CCBeam_1x1x20';          % Short name of part for reference
SAR_PartID(1) = 1;                        % Part ID (Not FlexID)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Joint Definitions
% Define joints using nodeset names


%%%% NOTE: FLEX PART IDS BELOW MATCH INPUT INDEX ABOVE, NOT SAR_PartID


%%% Joint 1: Blade bolts
J_ID = 1;
JFNames{J_ID}    = 'CCBeam_1x1x20_Root'; % Output name for ".j.dat" file
J_Desc{J_ID}     = 'Root Clamp CC 6x1x30'; % Brief description of the part
nJBods(J_ID)     = 1;                     % # bodies (flex) connected by joint
J_Type(J_ID)     = 0;                     % Joint Type: Boundary Joint
J_Dof{J_ID}      = [0 0 0 0 0 0];         % Joint DoF: Fixed
J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of NSets in joint
J_NSets{J_ID}{1} = {1,'NS401'};           % {Part #, nodeset names} (Beam)
JHead{J_ID,1}    = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint

J_ID = 2;
JFNames{J_ID}    = 'CCBeam_1x1x20_Tip';  % Output name for ".j.dat" file
J_Desc{J_ID}     = 'Tip Load CC 6x1x30'; % Brief description of the part
nJBods(J_ID)     = 1;                     % # bodies (flex) connected by joint
J_Type(J_ID)     = 1;                     % Joint Type: Standard Joint
J_Dof{J_ID}      = [1 1 1 1 1 1];         % Joint DoF: Free motion
J_NSets{J_ID}    = cell(nJBods(J_ID),1); % Initialize list of NSets in joint
J_NSets{J_ID}{1} = {1,'NS402'};                % {Part #, nodeset names} (Beam)
JHead{J_ID,1}    = sprintf('!   Joint %d: %s', J_ID, J_Desc{J_ID}); % Custom
header line for this specific joint



%%%% Append the version tag to each to each joint file output name
for jj = 1:nJoint
    JFNames{jj} = [JFNames{jj},ver_tag];
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%% Error Checks
```

```
%%%%% Check to make sure correct number of inputs were provided

%%% Ensure the correct number of meshes were provided
if length(MFNames) ~= nBod_Total
    fprintf(2,'ERROR: INCORRECT NUMBER OF MESH FILES PROVIDED\r\n');
    return
end

%%% Ensure the correct number of joints were provided
% Check J_Type
if length(J_Type) ~= nJoint
    fprintf(2,'ERROR: NOT ALL J_Type DEFINED\r\n');
    return
end
% Check J_Desc
if length(J_Desc) ~= nJoint
    fprintf(2,'ERROR: NOT ENOUGH JOINT DESCRIPTIONS PROVIDED\r\n');
    return
end
% Check J_Nsets and nJBods
if length(nJBods) ~= nJoint || length(J_NSets) ~= nJoint
    fprintf(2,'ERROR: INSUFFICIENT NUMBER OF JOINTS DEFINED\r\n');
    return
end
% Check J_DoF
if length(J_Dof) ~= nJoint
    fprintf(2,'ERROR: MISSING JOINT DoFs\r\n');
    return
end
```

Appendix IV

This appendix contains a sample `SAM.input` structural analysis model X3D input file, as described in Section 2.6.5. This particular file is for a composite box beam undergoing a vertical tip bending load. The beam mesh referenced was generated using the journal file provided in Appendix I, and is attached to root cantilever and tip loading joints created with the joint input file provided in Appendix III.

```
!================================================!
!                                                !
!                    SAM.input                   !
!                                                !
!================================================!

! SAM of a composite box beam for static tip loading


&NLDEF  model='SAM',  action='INIT',  class='SUBSYSTEM', &END
&NLVAL
        STYPE = 'ROTOR',
        IDS = 1,                            ! Subsystem ID
        IDR = 1,                            ! Rotor ID
        NCOMPONENT = 1,                     ! Number of components
&END
&NLDEF  action='ENDINITSUBSYSTEM',  &END


&NLDEF  model='SAM',  action='INIT',  class='COMPONENT', &END
&NLVAL
        IDS = 1, IDR = 1, IDB = 0, IDC = 1,
        NPART = 3,
&END
&NLDEF  action='ENDINITCOMPONENT',  &END



&NLDEF  model='SAM',  action='INIT',  class='FLEX3D',  &END
&NLVAL
      NAME = 'CC Beam 6x1x20',              ! Part name
        IDP = 1, TYPE = 'F', IDF = 1,       ! Part ID, Type, Type ID
        NCONNPARTS = 2,                     ! Number of connecting parts
         CONNPARTS = 2,3,                   ! List of connecting part IDs
&END

&NLDEF  class='JOINT',  &END
&NLVAL
```

```
      NAME = 'Root Clamp',
        IDP = 2, TYPE = 'J', IDJ = 1,
        NCONNPARTS = 1,
         CONNPARTS = 1,
&END

&NLDEF  class='JOINT',  &END
&NLVAL
      NAME = 'Tip Load',
        IDP = 3, TYPE = 'J', IDJ = 2,
        NCONNPARTS = 1,
         CONNPARTS = 1,
&END

&NLDEF  action='ENDINITPART',  &END



!--------------------------------------------------

! Rule: Must input flex part data first.
!       Flex part norder decides whether joint
!       connection nodes are to be reordered.

&NLDEF  model='SAM',  action='DATA',  class='FLEX3D', &END
&NLVAL
      IDC = 1, IDP = 1,
      GRIDFILE = './../../../grids/CCBeam_6x1x20_17072501.dat',
      FILEFORM = 0,
      GRSCALE  = 0.001,
      IGRAXIS  = 0,
      IFGR = 0,
      RXP = 0.,0.,0.,
      IH27NORDER = 1,
           H27NORDER = 2,10,3,11,4,12,1,9,18,19,20,17,6,14,7,15,8,16,5,
                       13,27,25,26,22,24,21,23,
      IH27FORDER = 1,
           H27FORDER = 6,5,1,3,2,4,
      NNODALF = 0,
      NNODALM = 0,
      DFC = 0.5,
&END

! Rule: Material of class must follow class;
!       as class reads in grid file where material
!       zones are defined.

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 101, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 0.,
          PORDER = 1,3,0,
&END
```

```
&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 102, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 103, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 104, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 105, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 106, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 90.,
          PORDER = 1,3,0,
&END



&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 107, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 108, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 109, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 110, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 111, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 112, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE =  0., 0., 90.,
          PORDER = 1,3,0,
&END




&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 113, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 114, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 115, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 116, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 117, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 118, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 90.,
          PORDER = 1,3,0,
&END




&NLDEF  class='MAT3D',  &END
&NLVAL
```

```
        IMAT = 119, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 120, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 121, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 122, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 123, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 0.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
        IMAT = 124, TYPE = 'ORTHTISO', RHO = 1444.,
          E1     = 141.9654e9, E2   = 9.7909e9,
          G12    =   6.1363e9, nu12 = 0.42,
          PANGLE = 90., 0., 90.,
          PORDER = 1,3,0,
&END

&NLDEF  class='MAT3D',  &END
&NLVAL
```

```
&END


&NLDEF  class='JOINT',  &END
&NLVAL
       IDC = 1, IDP = 2,
       GRIDFILE = './../../../grids/CCBeam_6x1x20_Root_17072501.j.dat',
       JFORM = 'EULER',
       JTYPE = 'GENERIC6',
        TJDOF = 0,0,0,0,0,0,
        PJDOF = 0,0,0,0,0,0,
        FJDOF = 0,0,0,0,0,0,
      RXP = 0., 0., 0.,
      CXP = 1.,0.,0.,0.,1.,0.,0.,0.,1.
&END


&NLDEF  class='JOINT',  &END
&NLVAL
      IDC = 1, IDP = 3,
      GRIDFILE = './../../../grids/CCBeam_6x1x20_Tip_17072501.j.dat',
      JFORM = 'EULER',
      JTYPE = 'GENERIC6',
        TJDOF = 1,1,1,1,1,1,
        PJDOF = 0,0,0,0,0,0,
        FJDOF = 0,0,1,0,0,0,
          NHJF  = 1,
          HJF   = 0.,   0., 4.4482,   0.,    0.,    0.,
        KJDOF =  0.,    0.,    0.,    0.,    0.,    0.,
                 0.,    0.,    0.,    0.,    0.,    0.,
                 0.,    0.,    0.,    0.,    0.,    0.,
                 0.,    0.,    0.,    0.,    0.,    0.,
                 0.,    0.,    0.,    0.,    0.,    0.,
                 0.,    0.,    0.,    0.,    0.,    0.,
      RXP =  .762, 0., 0.,
      CXP = 1., 0., 0., 0., 1., 0., 0., 0., 1.
&END

&NLDEF  action='END',  &END
```

References

[1] A. Datta and W. Johnson, "An Assessment of the State-of-the-Art in Multidisciplinary Aeromechanical Analyses," in *AHS Specialist's Conference on Aeromechanics*, San Francisco, CA, Januray, 2008.

[2] W. Johnson, "A History of Rotorcraft Comprehensive Analyses," in *American Helicopter Society 69th Annual Forum*, Phoenix, AZ, May, 2013.

[3] A. Datta, M. Nixon and I. Chopra, "Review of Rotor Loads Prediction with the Emergence of Rotorcraft CFD," *Journal of the American Helicopter Society,* vol. 52, no. 4, pp. 287-317, October 2007.

[4] W. Johnson, "Milestones in Rotorcraft Aeromechanics," *Alexander A. Nikolsky Honorary Lecture, Journal of the American Helicopter Society,* vol. 56, no. 3, pp. 1-24, July, 2011.

[5] C. W. Acree, "Effects of V-22 Blade Modifications on Whirl Flutter and Loads," *Journal of the American Helicopter Society,* vol. 50, no. 3, pp. 269-278, July 2005.

[6] W. Johnson and A. Datta, "Requirements for Next Generation Comprehensive Analysis of Rotorcraft," in *AHS Specialist's Conference on Aeromechanics*, San Francisco, CA, January, 2008.

[7] A. Datta and W. Johnson, "Three-Dimensional Finite Element Formulation and Scalable Domain Decomposition for High-Fidelity Rotor Dynamic Analysis," *Journal of the American Helicopter Society,* vol. 56, no. 21, p. 22003, 2011.

[8] A. Datta, "X3D - A 3D Solid Finite Element Multibody Dynamic Analysis for Rotorcraft," in *American Helicopter Society Technical Meeting on Aeromechanics Design for Vertical Lift*, San Francisco, CA, January 20-22, 2016.

[9] J. G. Leishman, Principles of Helicopter Aerodynamics, New York: Cambridge University Press, 2006.

[10] J. G. Leishman, "Validation of Approximate Indicial Aerodynamic Functions for Two-Dimensional Subsonic Flow," *Journal of Aircraft,* vol. 25, no. 10, pp. 914-922, 1988.

[11] T. S. Beddoes, "Practical Computation of Unsteady Lift," *Vertica,* vol. 8, no. 1, pp. 55-71, 1984.

[12] T. S. Beddoes, "Representation of Airfoil Behaviour," *Vertica,* vol. 7, no. 2, pp. 183-197, 1983.

[13] J. G. Leishman and T. S. Beddoes, "A Semi-Empirical Model for Dynamic Stall," *Journal of the American Helicopter Society,* vol. 34, no. 3, pp. 3-17, 1989.

[14] J. G. Leishman, "Modeling Sweep Effects on Dynamic Stall," *Journal of the American Helicopter Society,* vol. 34, no. 3, pp. 18-29, 1989.

[15] W. Johnson, "The Response and Airloading of Helicopter Rotor Blades Due to Dynamic Stall," ASRL TR 130-1, Cambridge, MA, 1970.

[16] R. Gormont, "A Mathematical Model of Unsteady Aerodynamics and Radial Flow for Application to Helicopter Rotors," USAAMRDL TR 72-67, Fort Eustis, VA, 1973.

[17] "Differential Equation Modeling of Dynamic Stall," *Recherche Aérospatiale,* vol. 5, pp. 59-72, 1989.

[18] V. Truong, "A 2-D Dynamic Stall Model Based on a Hopf Bifurcation," in *Nineteenth European Rotorcraft Forum*, Marseille, France, 1993.

[19] W. Johnson, "Rotorcraft Aerodynamics Models for a Comprehensive Analysis," in *AHS International 54th Annual Forum Proceedings*, San Francisco, CA, May 20-22, 1998.

[20] R. Piziali and F. DuWaldt, "Computation of Rotary Wing Harmonic Airloads and Comparison with Experimental Results," in *American Helicopter Society 18th Annual National Forum*, Washington, DC, 1962.

[21] R. Piziali, "A Method for Predicting the Aerodynamic Loads and Dynamic Response of Rotor Blades," *Journal of Sound and Vibration,* vol. 4, no. 3, pp. 445-489, 1966.

[22] A. Landgrebe, "An Analytical Method for Predicting Rotor Wake Geometry," *Journal of the American Helicopter Society,* vol. 14, no. 4, pp. 20-32, 1969.

[23] A. Landgrebe, "The Wake Geometry of a Hovering Rotor and its Influence on Rotor Performance," *Journal of the American Helicopter Society,* vol. 17, no. 4, pp. 3-15, 1972.

[24] M. P. Scully, "A Method of Computing Helicopter Vortex Wake Distortion," ASRL TR 138-1, MIT, 1967.

[25] M. P. Scull, "Computation of Helicopter Rotor Wake Geometry and Its Influence on Rotor Harmonic Airloads," ASRL TR 178-1, MIT, 1975.

[26] J. Kocurek and L. Berkovitz, "Velocity Coupling : A New Concept for Hover and Axial Flow Wake Analysis and Design," AGARD CP-334, 1982.

[27] T. Egolf and A. Landgrebe, "Helicopter Rotor Wake Geometry and its influence in Forward Flight, Vol. 1 - Generalized Wake Geometry and Wake Effects in Rotor Airloads and Performance," NASA CR-3726, 1983.

[28] T. S. Beddoes, "A Wake Model for High Resolution Airloads," in *Proceedings of the 2nd International Conference on Basic Rotorcraft Research*, Triange Park, NC, 1985.

[29] D. Wachspress, T. Quackenbush and A. Boschitsch, "First-Principles Free-Vortex Wake Analysis For Helicopters and Tiltrotors," in *American Helicopter*

Society International 59th Annual Forum, Phoenix, AZ, May 6-8, 2003.

[30] A. Bagai and J. G. Leishman, "Rotor Free-Wake Modeling Using a Pseudo-Implicit Technique - Including Comparisons with Experimental Data," *Journal of the American Helicopter Society,* vol. 40, no. 13, pp. 29-41, July, 1995.

[31] W. Johnson, "A General Free Wake Geometry Calculation For Wings and Rotors," in *American Helicopter Society 51st Annual Forum*, Fort Worth, TX, May 9-11, 1995.

[32] W. Johnson, "Influence of Wake Models on Calculated Tiltrotor Aerodynamics," in *AHS Aerodynamics, Acoustics, and Test and Evaluation Technical Specialists' Meeting, San Francisco*, San Francisco, CA, January 23-25, 2002.

[33] D. Clark and A. Leiper, "The Free Wake Analysis - A Method for Prediction of Helicopter Rotor Hovering Performance," *Journal of the American Helicopter Society,* vol. 15, no. 1, pp. 3-11, 1970.

[34] S. G. Sadler, "A Method for Predicting Helicopter Wake Geometry, Wake Induced Inflow and Wake Effects on Blade Airloads," in *American Helicopter Society 27th Annual National Forum*, Washington, DC, May 1971.

[35] M. Bhagwat and J. G. Leishman, "Stability, Consistency and Convergence of Numerical Algorithms for Time-Marching Free-Vortex Wake Analyses," *Journal of the American Helicopter Society,* vol. 46, no. 1, pp. 59-71, 2001.

[36] M. Bhagwat and J. G. Leishman, "Rotor Aerodynamics During Maneuvering Flight Using a Time-Accurate Free-Vortex Wake," *Journal of American Helicopter Society,* vol. 48, no. 3, pp. 143-158, 2003.

[37] B. M. Govindarajan and J. G. Leishman, "Prediction of Rotor and Rotor/Airframe Configurational Effects on Brownout Dust Clouds," *Journal of Aircraft,* Vols. (Yet to appear, doi: 10.2514/1.C033447), 2015.

[38] K. W. Mangler and H. B. Squire, "Calculation of the Induced Velocity Field of a Rotor," Royal Aircraft Establishment, Report Aero 2247, September 1948.

[39] M. Joglekar and R. Loewy, "An Actuator-Disk Analysis of Helicopter Wake Geometry and the Corresponding Blade Response," USAAVLABS, Technical Report 69-66, December 1970.

[40] D. Pitt and D. Peters, "Rotor Dynamic Inflow Derivatives and Time Constants from Various Inflow Models," in *9th European Rotorcraft Forum*, Stresa, Italy, September 13-15, 1983.

[41] D. Peters and C. J. He, "Finite State Induced Flow Models Part II: Three-Dimensional Rotor Disk," *Journal of Aircraft,* vol. 32, no. 2, pp. 323-333, 1995.

[42] W. Johnson, "Milestones in Rotorcraft Aeromechanics," NASA TP-2011-215971, Moffett Field, CA, 2011.

[43] D. A. Peter, "How Dynamic Inflow Survives in the Competitive World of Rotorcraft Aerodynamics," *Journal of the American Helicopter Society,* vol. 53, no. 1, pp. 1-15, January 2009.

[44] H. Glauert, "The Theory of the Autogyro," *The Journal of the Royal Aeronautical Society,* vol. 31, no. 198, pp. 483-508, June, 1927.

[45] J. Houbolt and G. Brooks, "Differential Equations of Motion for Combined Flapwise Bending, Chordwise Bending, and Torsion of Twisted Nonuniform Rotor Blades," NACA TN 3905, 1957.

[46] D. Hodges and E. Dowell, "Nonlinear Equations of Motion for the Elastic Bending and Torsion of Twisted Nonuniform Rotor Blades," NASA TN D-7818, 1974.

[47] D. Hodges, R. Ormiston and D. Peters, "On the Nonlinear Deformation Geometry of Euler–Bernoulli Beams," NASA TP 1566, 1980.

[48] R. Kvaternik and R. Kirshna, "Nonlinear Curvature Expressions for Combined Flapwise Bending, Chordwise Bending, Torsion, and Extension of Twisted Rotor Blades," NASA TM X-73997, 1976.

[49] A. Rosen and P. Friedmann, "Nonlinear Equations of Equilibrium for Elastic Helicopter or Wind Turbine Blades Undergoing Moderate Deformation," NASA CR-159478, 1978.

[50] W. Johnson, "Aeroelastic Analysis for Rotorcraft in Flight or in a Wind Tunnel," NASA TN D-8515, 1977.

[51] F. Straub and P. Friedmann, "A Galerkin Type Finite Element Method for Rotary-Wing Aeroelasticity in Hover and Forward Flight," in *Sixth European Rotorcraft and Powered Lift Aircraft Forum*, Bristol, UK, September 16-19, 1980.

[52] I. Chopra and N. Sivaneri, "Aeroelastic Stability of Rotor Blades Using Finite Element Analysis," NASA CR 166389, 1982.

[53] A. Bauchau and C. Hong, "Nonlinear Composite Beam Theory," *Journal of Applied Mechanics,* vol. 55, no. 1, pp. 156-163, March 1988.

[54] D. Hodges, "A Mixed Variational Foundation Based on Exact Intrinsic Equations for Dynamics of Moving Beams," *International Journal of Solids and Structures,* vol. 26, no. 11, pp. 1253-1273, December 1990.

[55] E. Smith and I. Chopra, "Aeroelastic Response, Loads, and Stability of a Composite Rotor in Forward Flight," *AIAA Journal,* vol. 31, no. 7, pp. 1265-1273, July 1993.

[56] O. Bauchau and N. Kang, "A Multibody Formulation for Helicopter Structural Dynamic Analysis," *Journal of the American Helicopter Society,* vol. 38, no. 2, pp. 3-14, April 1993.

[57] W. Johnson, "Rotorcraft Dynamics Models for a Comprehensive Analysis," in *AHS International 54th Annual Forum Proceedings*, San Francisco, CA, May 20–22, 1998.

[58] L. G. G., P. Masarati, P. Mantegazza and M. W. Nixon, "Multi-Body Analysis of the 1/5 Scale Wind Tunnel Model of the V-22 Tiltrotor," in *American Helicopter Society 55th Annual Forum*, Montreal, QC, May 1999.

[59] H. Saberi, M. Khoshlahjeh, R. Ormiston and M. Rutkowski, "Overview of RCAS and Application to Advanced Rotorcraft Problems," in *4th AHS Decennial Specialist's Conference on Aeromechanics*, San Francisco, CA, January 21-23, 2004.

[60] A. Abishek, A. Datta and I. Chopra, "Prediction of UH-60A Structural Loads Using Multibody Analysis and Swashplate Dynamics," *Journal of Aircraft,* vol. 46, no. 2, pp. 474-490, 2009.

[61] C. Cesnik and D. Hodges, "VABS: A New Concept for Composite Rotor Blade Cross-Sectional Modeling," *Journal of the American Helicopter Society,* vol. 42, no. 1, pp. 27-38, January 1997.

[62] G. Jelenić and M. Crisfield, "Geometrically Exact 3D beam Theory: Implementation of a Strain-Invariant Finite Element for Statics and Dynamics," *Computer Methods in Applied Mechanics and Engineering,* vol. 171, no. 1-2, pp. 141-171, March 1999.

[63] O. Bauchau, C. Bottasso and Y. Nikishkov, "Modeling Rotorcraft Dynamics with Finite Element Multibody Procedures," *Mathematical and Computer Modeling,* vol. 33, no. 10-11, pp. 1113-1137, 2001.

[64] D. Hodges, "Geometrically Exact, Intrinsic Theory for Dynamics of Curved and Twisted Anisotropic Beams," *AIAA Journal,* vol. 41, no. 6, pp. 1131-1137, June 2003.

[65] S. Han and O. Bauchau, "Nonlinear Three-Dimensional Beam Theory for Flexible Multibody Dynamics," *Multibody System Dynamics,* vol. 34, no. 3, pp. 211-242, 2015.

[66] Y. Bazilevs, M. C. Hsu, J. Kiendl, R. Wüchner and K. U. Bletzinger, "3-D Simulation of Wind Turbine Rotors at Full Scale. Part Ii: Fluid–Structure Interaction Modeling with Composite Blades," *International Journal for Numerical Methods in Fluids,* vol. 65, no. 1-3, pp. 236-253, January 2011.

[67] C. Bottasso, O. Bauchau and J. Choi, "An Energy Decaying Scheme for Nonlinear Dynamics of Shells," *Computer Methods in Applied Mechanics and Engineering,* vol. 191, no. 27, pp. 3099-3121, 2002.

[68] O. Bauchau, J. Choi and C. Bottasso, "On the Modeling of Shells in Multibody Dynamics," *Multibody System Dynamics,* vol. 8, no. 4, pp. 459-489, 2002.

[69] H. Kang, C. Chang, H. Saberi and R. Ormiston, "Assessment of Beam and Shell Elements for Modeling Rotorcraft Blades," *Journal of Aircraft,* vol. 51, no. 2, pp. 520-531, March-April 2014.

[70] H. Yeo, K. Truong and R. Ormiston, "Assessment of 1-D Versus 3-D Methods for Modeling Rotor Blade Structural Dynamics," in *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, April 2010.

[71] K. Truong, H. Yeo and R. Ormiston, "Investigation of Finite Element Approaches for Rotor Blade Structural Dynamics," in *36th European Rotorcraft Forum*, Paris, France, September 2010.

[72] D. Post, C. Atwood, K. Newmeyer and S. Landsber, "The Computational Research and Engineering Acquisition Tools and Environments (CREATE) Program," *Computing in Science & Engineering,* vol. 18, no. 1, pp. 10-13, 2016.

[73] A. Datta and W. Johnson, "Three-dimensional Finite Element Formulation and

Scalable Domain Decomposition for High Fidelity Rotor Dynamic Analysis," in *American Helicopter Society 65th Annual Forum*, Grapevine, TX, May 27-29. 2009.

[74] A. Datta and W. Johnson, "A Multibody Formulation For Three Dimensional Brick Finite Element Based Parallel and Scalable Rotor Dynamic Analysis," in *American Helicopter Society 66th Annual Forum*, Phoenix, AZ, May 11-13, 2010.

[75] A. Datta and W. Johnson, "Integrated Aeromechanics with Three-Dimensional Solid-Multibody Structures," in *American Helicopter Society 70th Annual Forum Proceedings*, Montreal, QC, May 20-22, 2014.

[76] W. Staruk, I. Chopra and A. Datta, "Three-Dimensional CAD-Based Structural Modeling for Next Generation Rotor Dynamic Analysis," in *American Helicopter Society 70th Annual Forum Proceedings*, Montreal, QC, May 22-24, 2014.

[77] W. Staruk, E. Ward and I. Chopra, "CAD-Based 3-D Structural Dynamic Modeling of the Tilt Rotor Aeroacoustic Model (TRAM) Proprotor," in *American Helicopter Society 71st Annual Forum Proceedings*, Virginia Beach, VA, May 5-7, 2015.

[78] W. Staruk, I. Chopra and A. Datta, "Coupled Aerodynamics and 3-D Structural Dynamics of the Tilt Rotor Aeroacoustic Model (TRAM) Proprotor," in *AHS International Technical Meeting on Aeromechanics Design for Vertical Lift*, San Francisco, CA, January 20-22, 2016.

[79] W. Staruk, I. Chopra, A. Datta and B. Jayaraman, "Validation of Aeromechanics Predictions for a Full 3-D Structural Analysis Model of the Tilt Rotor Aeroacoustic Model (TRAM) Proprotor," in *42nd European Rotorcraft Forum*, Lille, France, September 6-8, 2016.

[80] W. Staruk and A. Datta, "Fundamental Understanding, Prediction, and Validation

of Tiltrotor Dynamic Loads in Transition Flight Using RANS/FEA," in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Science and Technology Forum and Exposition 2017*, Grapevine, TX, January 9-13, 2017.

[81] W. Staruk, I. Chopra and A. Datta, "Loads Prediction for a Gimbaled Tiltrotor in Conversion Flight Using CAD-Based 3-D Structural Analysis Models," in *American Helicopter Society 73rd Annual Forum Proceedings*, Fort Worth, TX, May 9-11, 2017.

[82] E. Ward, I. Chopra and A. Datta, "RPM Driven Extension-Torsion Coupled Self-Twisting Rotor Blades," in *AHS 72nd Annual Forum*, West Palm Beach, FL, May 17-19, 2016.

[83] E. Ward, I. Chopra and A. Datta, "Design of Self-Twisting Rotor Blades for High-Speed Compound Rotorcraft," in *25th AIAA/AHS Adaptive Structures Conference, AIAA Science and Technology Forum and Exposition 2017*, Grapevine, TX, January 9-13, 2017.

[84] E. Ward, I. Chopra and A. Datta, "RPM Driven Extension-Torsion Coupled Self-Twisting Rotor Blades," in *American Helicopter Society 73rd Annual Forum*, Fort Worth, TX, May 2017.

[85] I. Fejtek and L. Roberts, "Navier-Stokes Computation of Wing/Rotor Interaction for a Tilt Rotor in Hover," *AIAA Journal,* vol. 30, no. 11, pp. 2595-2603, November 1992.

[86] R. Meakin, "Moving Body Overset Grid Methods for Complete Aircraft Tiltrotor Simulations," in *11th Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, AIAA Paper 3350*, Orlando, FL, 1993.

[87] H. Tadghighi, G. Rajagopalan and C. Burley, "Simulation of Tiltrotor Fountain Flow Field Effects Using a Finite Volume Technique - An Aero/Acoustic

Study," in *American Helicopter Society 51st Annual Forum Proceedings*, Fort Worth, TX, May 11, 1995.

[88] D. Poling, H. Rosenstein and G. Rajagopalan, "Use of a Navier-Stokes Code in Understanding Tiltrotor Flowfields in Hover," *Jounral of the American Helicopter Society,* vol. 43, no. 2, pp. 103-109, April 1998.

[89] M. Potsdam and R. Strawn, "CFD Simulations of Tiltrotor Configurations in Hover," *Journal of the American Helicopter Society,* vol. 50, no. 1, pp. 82-94, January 2005.

[90] N. Chaderjian and P. Buning, "High Resolution Navier-Stokes Simulation of Rotor Wakes," in *American Helicopter Society 67th Annual Forum Proceedings*, Virginia Beach, VA, May 3-5, 2011.

[91] A. Wissink, B. Jayaraman, A. Datta, J. Sitaraman, M. Potsdam, S. Kamkar, D. Mavriplis, Z. Yang, R. Jain, J. Lim and R. Strawn, "Capability Enhancement in Version 3 of the Helios High-Fidelity Rotorcraft Simulation Code," in *50th AIAA Aerospace Sciences Meeting*, Nashville, TN, 09-12 January, 2012.

[92] E. M. Lee-Rausch and R. T. Biedron, "Simulation of an Isolated Tiltrotor in Hover with an Unstructured Overset-Grid RANS Solver," in *American Helicopter Society 65th Annual Forum*, Grapevine, TX, May 27-29, 2009.

[93] T. L. Holst and T. H. Pulliam, "Overset Solution Adaptive Grid Approach Applied to Hovering Rotorcraft Flows," in *27th AIAA Applied Aerodynamics Conference*, San Antonio, Texas, 22-25 June 2009.

[94] A. M. Wissink, M. Potsdam, V. Sankaran, J. Sitaraman and D. Mavriplis, "A Dual-Mesh Unstructured Adaptive Cartesian Computational Fluid Dynamics Approach for Hover Prediction," *Journal of the American Helicopter Society,* vol. 61, no. 1, pp. 1-19, January 2016.

[95] W. Johnson, "Calculation of Tilt Rotor Aeroacoustic Model (TRAM DNW)

Performance, Airloads, and Structural Loads," in *American Helicopter Society Aeromechanics Specialists' Meeting*, Atlanta, GA, November 13-15, 2000.

[96] W. E. Hall, "Prop-Rotor Stability at High Advance Ratios," *Journal of the American Helicopter Society,* vol. 11, no. 2, pp. 11-26, 1966.

[97] T. Gaffey, "The Effect of Positive Pitch-flap Coupling (Negative δ3) on Rotor Blade Motion Stability and Flapping," *Journal of the American Helicopter Society,* vol. 14, no. 2, pp. 49-67, 1969.

[98] W. Johnson, "Dynamics of Tilting Proprotor Aircraft in Cruise Flight," NASA TN D-7677, NASA Ames Research Center, 1974.

[99] W. Johnson, B. H. Lau and J. V. Bowles, "Calculated Performance, Stability, and Maneuverability of High-Speed Tilting-Prop-Rotor Aircraft," NASA TM 88349, 1986.

[100] V. Srinivas, I. Chopra and M. W. Nixon, "Aeroelastic Analysis of Advanced Geometry Tiltrotor Aircraft," *Journal of the American Helicopter Society,* vol. 43, no. 3, pp. 212-221, July 1998.

[101] S. M. Barkai and O. Rand, "The Influence of Composite Induced Couplings on Tiltrotor Whirl Flutter Stability," *Journal of the American Helicopter Society,* vol. 43, no. 2, pp. 133-145, April 1998.

[102] M. W. Nixon, D. J. Piatak, L. M. Corso and D. A. Popelka, "Aeroelastic Tailoring for Stability Augmentation and Performance Enhancements of Tiltrotor Aircraft," *Journal of the American Helicopter Society,* vol. 45, no. 4, pp. 270-279, October 2000.

[103] C. W. Acree, R. J. Peyran and W. Johnson, "Improving Tiltrotor Whirl-mode Stability with Rotor Design Variations," in *European Rotorcraft Forum 26*, The Hague, Netherlands, September, 2000.

[104] M. Potsdam and R. Strawn, "CFD Simulations of Tiltrotor Configurations in Hover," in *American Helicopter Society 58th Annual Forum*, Montreal, QC, June 11-13, 2002.

[105] C. W. Acree, "Rotor Design Options for Improving V-22 Whirl-Mode Stability," in *American Helicopter Society 58th Annual Forum Proceedings*, Montreal, QC, June 11-13, 2002.

[106] J. Shen, J. Singleton, D. Piatak, O. Bauchau and P. Masarati, "Multibody Dynamics Simulation and Experimental Investigation of a Model-Scale Tiltrotor," *Jounral of the American Helicopter Society,* vol. 61, no. 2, pp. 022010-1 - 022010-11, 2016.

[107] J. M. Bilger, R. L. Marr and A. Zahedi, "In-Flight Structural Dynamic Characteristics of the XV-15 Tilt Rotor Research Aircraft," *Journal of Aircraft,* vol. 19, no. 11, pp. 1005-1011, November 1982.

[108] J. J. Totah and J. F. I. Madden, "Rotor and Control System Loads Analysis of the XV-15 With the Advanced Technology Blades," NASA TM 102876, April 1991.

[109] W. Johnson, "Calculation of the Aerodynamic Behavior of the Tilt Rotor Aeroacoustic Model (TRAM) in the DNW," in *American Helicopter Society 57th Annual Forum Proceedings*, Washington, DC, May 9-11, 2001.

[110] W. Johnson, "Airloads and Wake Geometry Calculations for an Isolated Tiltrotor Model in a Wind Tunnel," in *27th European Rotorcraft Forum*, Moscow, Russia, September, 2001.

[111] J. Ho and H. Yeo, "Comparison of Calculated and Measured Blade Loads of the Tilt Rotor Aeroacoustic Model," in *AHS 73rd Annual Forum*, Fort Worth, TX, May, 2017.

[112] M. Smith, "Curating Architectural 3D CAD Models," *International Journal of*

Digital Curation, vol. 4, no. 1, pp. 98-106, 2009.

[113] I. C. Braid, "The Synthesis of Solids Bounded by Many Faces," *Communications of the ACM,* vol. 18, no. 4, pp. 209-216, 1975.

[114] D. Laidlaw, W. B. Trumbore and J. Hughes, "Constructive Solid Geometry for Polyhedral Objects," *ACM SIGGRAPH Computer Graphics,* vol. 20, no. 4, pp. 161-170, 1986.

[115] J. J. Shah, "Assessment of Features Technology," *Computer-Aided Design,* vol. 23, no. 5, pp. 331-343, 1991.

[116] J. Cambra, M. Contero and P. Company, "Parametric CAD Modeling: An Analysis of Strategies for Design Reusability," *Computer-Aided Design,* vol. 74, pp. 18-31, 2016.

[117] S. Gerbino, "Tools for the Interoperability Among CAD Systems," in *XIII ADM - XV INGEGRAF International Conference on Tools and Methods Evolution in Engineering Design*, Naples. Italy, 2003.

[118] L. A. Young, E. R. Booth, G. K. Yamauchi, G. Botha and S. Dawson, "Overview of the Testing of a Small-Scale Proprotor," in *American Helicopter Society 55th Annual Forum Proceedings*, Montreal, QC, Canada, May 25-27, 1999.

[119] I. Babuška and S. Manil, "On Locking and Robustness in the Finite Element Method," *SIAM Journal on Numerical Analysis,* vol. 29, no. 5, pp. 1261-1293, 1992.

[120] UC Structural Dynamics Research Lab, [Online]. Available: http://www.sdrl.uc.edu/sdrl/referenceinfo/universalfileformats/file-format-storehouse/universal-file-datasets-summary. [Accessed 22 March 2016].

[121] T. Blacker, S. Owen, M. Staten, R. Quadros, B. Hanks and et al., "CUBIT Geometry and Mesh Generation Toolkit 15.2 User Documentation," Sandia National Laboratories, SAND2016-1649 R, Albuquerque, New Mexico, May,

2016.

[122] T. D. Blacker and M. B. Stephenson, "Paving: A New Approach to Automated Quadrilateral Mesh Generation," *International Journal for Numerical Methods in Engineering,* vol. 32, no. 4, pp. 811-847, 1991.

[123] D. R. White and K. Paul, "Redesign of the Paving Algorithm: Robustness Enhancements Through Element by Element Meshing," in *Proceedings 6th International Meshing Roundtable*, 1997.

[124] P. M. Knupp, "Next-Generation Sweep Tool: A Method for Generating All-Hex Meshes on Two-and-One-Half Dimensional Geometries," in *Proceedings of the Seventh International Meshing Roundtable*, 1998.

[125] L. Mingwu, S. Benzley, G. Sjaardema and T. Tautges, "A Multiple Source and Target Sweeping Method for Generating All Hexahedral Finite Element Meshes," in *Proceedings, 5th International Meshing Roundtable*, Albuquerque, NM, October, 1996.

[126] P. M. Knupp, "Algebraic Mesh Quality Metrics for Unstructured Initial Meshes," *Finite Elements in Analysis and Design,* vol. 39, no. 3, pp. 217-241, 2003.

[127] C. T. Sun and S. Liao, "Three-Dimensional Effective Elastic Constants for Thick Laminates," *Journal of Composite Materials,* vol. 22, no. 7, pp. 629-639, July, 1988.

[128] A. Alexander and J. T. Tzeng, "Three Dimensional Effective Properties of Composite Materials for Finite Element Applications," *Journal of Composite Materials,* vol. 31, no. 5, pp. 466-485, 1997.

[129] N. J. Pagano and F. G. Yuan, "The Significance of Effective Modulus Theory (Homogenization) in Composite Laminate Mechanics," *Composite Science and Technology,* vol. 60, no. 12, pp. 2471-2788, Sept. 2000.

[130] P. C. Chou, J. Carleone and C. M. Hsu, "Elastic Constants of Layered Media," *Journal of Composite Materials,* vol. 6, no. 1, pp. 80-93, January 1972.

[131] S. Jung and B. Lau, "Determination of HART I Blade Structural Properties," NASA/CR-2012-216039, NASA Ames Research Center, 2012.

[132] W. Carnegie, "Static Bending of Pre-twisted Cantilever Blading," *Proceedings of the Institution of Mechanical Engineers,* vol. 171, no. 1, pp. 879-894, 1957.

[133] D. Zupan and M. Saje, "On “A proposed standard set of problems to test finite element accuracy”: the twisted beam," *Finite Elements in Analysis and Design,* vol. 40, pp. 1445-1451, 2004.

[134] R. Chandra, A. D. Stemple and I. Chopra, "Thin-Walled Composite Beams Under Bending, Torsional, and Extensional Loads," *Journal of Aircraft,* vol. 27, no. 7, pp. 619-626, July 1990.

[135] J. Fan and J. Ye, "An Exact Solution for the Statics and Dynamics of Laminated Thick Plates with Orthotropic Layers," *International Journal of Solid Structures,* vol. 26, no. 5/6, pp. 655-662, 1990.

[136] K. T. Sundara Raja Iyengar, K. Chandrashekhara and V. K. Sebastian, "On the Analysis of Thick Rectangular Plates," *Archive of Applied Mechanics,* vol. 43, no. 5, pp. 317-330, 1974.

[137] K. T. Sundara Raja Iyengar and S. K. Pandya, "Analysis of Orthotropic Rectangular Thick Plates," *Fibre Science and Technology,* vol. 17, no. 1, pp. 19-36, 1983.

[138] L. A. Young, "Tilt Rotor Aeroacoustic Model (TRAM): A New Rotorcraft Research Facility," in *AHS International Meeting on Advanced Rotorcraft Technology and Disaster Relief*, Gifu, Japan, 1998.

[139] J. L. Johnson and L. A. Young, "Tilt Rotor Aeroacoustic Model Project," in *Confederation of European Aerospace Societies (CEAS) Forum on*

Aeroacoustics of Rotorcraft and Propellers, Rome, Italy, June 9-11, 1999.

[140] L. A. Young, D. Lillie, M. McCluer and G. K. Yamauchi, "Insights into Airframe Aerodynamics and Rotor-on-Wing Interactions from a 0.25-Scale Tiltrotor Wind Tunnel Model," in *AHS International Aerodynamics, Acoustics, and Test and Evaluation Specialists' Conference*, San Francisco, CA, January 23-25, 2002.

[141] M. S. McCluer and J. L. Johnson, "Full-Span Tiltrotor Aeroacoustic Model (FS TRAM) Overview and Initial Testing," in *American Helicopter Society Aerodynamics, Acoustics, and Test and Evaluation Technical Specialists' Meeting Proceedings*, San Francisco, CA, Jan. 23-25, 2002.

[142] CMH-17 Composite Materials Handbook Volumes 1, 2, and 3, SAE International, 2012.

[143] R. Chen, "A Survey of Nonuniform Inflow Models for Rotorcraft Flight Dynamics and Control Applications," NASA Technical Memorandum 102219, 1989.

[144] C. W. Acree, P. Martin and E. Romander, "Impact of Airfoils on Aerodynamic Optimization of Heavy Lift Rotorcraft," in *AHS Vertical Lift Aircraft Design*, San Francisco, CA, January 18-20, 2006.

[145] C. W. Acree, H. Yeo and J. Sinsay, "Perfromance Optimization of the NASA Large Civil Tiltrotor," in *International Powered Lift Conference*, London, UK, July, 2008.

[146] M. K. Farrell, "Aerodynamic Design of the V-22 Osprey Proprotor," in *45th Annual Forum of teh American Helicopter Society*, Boston, MA, May, 1989.

[147] R. Strawn, "High-Performance Computing for Rotorcraft Modeling and Simulation," *Computing in Science and Engineering,* vol. 12, no. 5, pp. 27-35, Sep-Oct 2010.

[148] J. Sitaraman, M. Potsdam, A. Wissink, B. Jayaraman, D. Mavriplis and H. Saberi, "Rotor Loads Prediction Using Helios: A Multisolver Framework for Rotorcraft Aeromechanics," *Journal of Aircraft,* vol. 50, no. 2, pp. 478-492, March 2013.

[149] H. Yeo and I. Chopra, "Coupled Rotor/Fuselage Vibration Analysis for Teetering Rotor and Test Data Comparison," *Journal of Aircraft,* vol. 38, no. 1, pp. 111-121, 2001.

[150] F. Richez and B. Ortun, "Numerical Investigation of the Flow Separation on a Helicopter Rotor in Dynamic Stall Configuration," in *42nd European Rotorcraft Forum*, Lille, France, Sept. 5-8, 2016.