VALIDATION OF INTERPOLATIVE INTERFACES

FOR ROTORCRAFT APPLICATIONS

By

Adam Cofer

Dr. Kidambi Sreenivas
Research Professor
(Chair)

Dr. Robert S. Webster
Associate Research Professor
(Committee Member)

Dr. Steve L. Karman, Jr.
Professor
(Committee Member)

VALIDATION OF INTERPOLATIVE INTERFACES

FOR ROTORCRAFT APPLICATIONS

By

Adam Cofer

A Thesis Submitted to the Faculty of the University
of Tennessee at Chattanooga in Partial Fulfillment
of the Requirements of the Degree of
Master of Science in Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

December 2013

ABSTRACT

The study uses computational methods to simulate fluid flow on the NASA ROBIN helicopter model and on a simplified rotor geometry previously tested at Mississippi State. The ROBIN model and the rotor are run using an unstructured grid. Results from the Tenasi flow solver are compared against both simulated and wind tunnel data. Tenasi is an unstructured, Reynolds Averaged Navier-Stokes (RANS) solver developed at the SimCenter: National Center for Computational Engineering, located at the University of Tennessee at Chattanooga.

Steady-state results for the isolated ROBIN fuselage and unsteady results for both fuselage and rotor systems are computed. In the unsteady case, relative grid motion of both the rotor disk relative to the fuselage and the cyclic pitching motion of each blade as the rotor turns must be simulated. Thus, each moving component is meshed in its own subdomain, and a nonmatching and sliding interface method is used to compute fluxes across the subdomain boundaries. Testing of the implementation of this method in Tenasi is the primary purpose of the study.

DEDICATION

To my wife, Tiffany, without whose encouragement and patience I would not have been able to complete this work.

ACKNOWLEDGEMENTS

First, I would like to thank Dr. Kidambi Sreenivas for his support and direction on this project. I would like to thank my committee members, Dr. Robert Webster and Dr. Steve Karman, and also Dr. Daniel Hyams, for their interest, advice, time, and software. Bruce Hilbert has provided invaluable assistance in grid generation on the ROBIN body and through support in testing of the code on the Georgia Tech teetering-rotor case.

TABLE OF CONTENTS

APPENDIX

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

AR, rotor aspect ratio, $R\,/\,c$

c, rotor chord in m

c_p, specific heat in J/(kg K)

C_T, thrust coefficient, $T/\left(\rho_\infty(\Omega\,R)^2\pi\,R^2\right)$

C_p, pressure coefficient, $(p-p_\infty)/q_\infty$ (see note on U_∞ below)

E, total energy per unit volume in J/m^3

H, total enthalpy per unit mass given by $H=\dfrac{E+P}{\rho}$ in J/kg

H/R, height of rotor hub above fuselage centerline, nondimensionalized

k, thermal conductivity in W/(m K)

L, representative length scale of system (e.g. chord length or rotor radius)

M_∞, freestream Mach number

P, static pressure in Pa

Pr, Prandtl number, $\dfrac{\nu}{\alpha}=\dfrac{\eta/\rho}{k/(\rho\,c_p)}$

Pr_t, turbulent Prandtl number (ratio between momentum eddy diffusivity and heat transfer eddy diffusivity)

q_∞, freestream dynamic pressure, $\dfrac{1}{2}\rho_\infty U_\infty^2$ in Pa (see note on U_∞ below)

r, radial blade station (distance along blade from axis)

R, rotor radius in m

Re, Reynolds number, $U_\infty L/v$

u, v, w; components of velocity parallel to x, y, and z axis, respectively, in m/s

U_∞, reference velocity in m/s, usually equal to U_{tip}. NOTE: for some hovercraft simulations, U_∞ is local rotational velocity of the blade section $(U_{tip} * r/R)$

U_{tip}, rotational velocity of rotor blade tip, also referred to as V_∞

$\vec{V}$, flow velocity vector, $u\hat{i}+v\hat{j}+w\hat{k}$

V_x, V_y, V_z, relative grid motion velocity

V_∞, rotor tip speed in m/s or ft/s

X_B, distance along fuselage in the X direction in meters

X_N, distance from fuselage nose to rotor hub in X direction in meters

γ, ratio of specific heats c_P / c_V (specific heat at constant pressure over sp. heat at constant volume)

θ_0, θ_{1c}, θ_{1s}; collective, lateral, and longitudinal blade pitch in degrees

μ, advance ratio U_∞ / V_∞, or dynamic viscosity in Pa·s

μ_t, turbulent or eddy viscosity in Pa·s

v, kinematic viscosity, μ/ρ in m^2/s

ρ, fluid density, kg/m^3

ρ_∞, freestream density, kg/m^3

ρu, ρv, and ρw, Cartesian momentum components in the x, y, and z directions, respectively, in $kg/m^2 \cdot s$

σ, rotor solidity, $N_{blades} / \pi AR$

ψ, blade azimuth (0 deg at rear of rotor disk), degrees

Ω, rotor rotation rate in rad/s

$\hat{i}$, $\hat{j}$, $\hat{k}$, unit vectors in the *x*, *y*, and *z* directions, respectively

τ_{xx} , τ_{xy} , etc., shear stresses in Pascals

CHAPTER I

INTRODUCTION

Background and Previous Work

At NASA Langley Research Center, Freeman and Mineck (1979) took a large amount of experimental data over a ROBIN (ROtor-Body INteraction) fuselage with and without a rotor system. The ROBIN is an analytically defined notional generic helicopter body, described later. The wind-tunnel experiment was conducted to expand the data available to validate analytic models of the flow field around a helicopter fuselage. Test result data were presented without analysis. At the Georgia Institute of Technology, Brand, McMahon, and Komerath (1989) conducted wind-tunnel tests of a simple cylindrical fuselage with a two-bladed teetering rotor system. The Georgia Tech test was intended to provide data for verification of Computational Fluid Dynamics methods for modeling of rotorcraft in flight. At Mississippi State University, Webster (1994) undertook to simulate a helicopter rotor (isolated) in forward flight. A turbomachinery code using the Reynolds-averaged, Navier- Stokes equations with thin-layer approximation called TURBO was modified and used. The modification consisted of cyclic blade surface motion function implementation with grid deformation/distortion using weighting functions and the grid motion techniques used to simulate the rotor assembly rotation. In Schweitzer (1999), an unstuctured simulation of the ROBIN fuselage without a rotor was run using the Parallel Unstructured Maritime Aerodynamics (PUMA) solver from Virginia Polytechnical Institute and State University.

In Mineck (1999), an unstructured-grid, Navier-Stokes solver was used to predict the surface pressure distribution, the off-body flow field, the surface flow pattern, and integrated lift and drag coefficients on the ROBIN configuration without a rotor at four angles of attack. Another wind tunnel test for the ROBIN was run by Mineck and Gorton (2000), and steady and periodic (unsteady) pressure measurements on the fuselage were taken, with peaks corresponding to blade passage. The model was tested at four advance ratios and three thrust coefficients. Data for the unsteady cases are provided, but no analysis was performed. That data is the primary reference for ROBIN simulations in this study.

Park and Kwon (2004) conducted an unstructured-mesh simulation of an isolated helicopter rotor with the three-dimensional Euler (inviscid) flow equations using a sliding-plane method for flow across a single subdomain boundary parallel to and below the rotor plane. Mesh deformation for cyclic blade pitching motion in forward flight was handled using a spring analogy and cell-edge collapsing.

At Mississippi State University, several methods were employed for relative grid movement. According to Blades and Marcum (2005),

> ...an unstructured method for node-centered schemes was developed by Sreenivas et al. for tilt-rotor simulations. This method, referred to as the UVI method, employs local grid reconnection to enable relative grid motion... The UVI method, in principle, is an unstructured implementation of the localized grid distortion and clicking method introduced by Janus and Whitfield (1990). The local-reconnection process reconnects the distorted grid lines at the interface, and the inner grid is essentially clicked into place.

In Blades and Marcum (2005), a sliding interface method is developed for simulations involving relative grid motion that requires no grid deformation, remeshing, or hole cutting. This is the method currently used in Tenasi, and it will be described later in Chapter 1. Nam, Park, and Kwon (2006) – following directly on Park and Kwon (2004), also at Korea Advanced Institute of Science and Technology – ran unstructured-mesh simulations of both the ROBIN and Georgia Tech rotorcraft with rotors in forward flight in the three-dimensional Euler flow equations. Again, periodic blade flapping due to pressure fluctuations was modeled via mesh deformation. An upper and lower rotating zone were used for the rotor and fuselage, respectively, with a single sliding plane interface between them. Trim sensitivity calculations were performed on a coarse grid, which was then refined via mesh adaptation to provide results that reasonably matched wind-tunnel data.

Mitchell (2007) uses a constructive geometry approach to calculate the overlap of two surface elements arbitrarily oriented in three-dimensional Cartesian space. This information allows edges to be created in the unstructured solver that allow for a flux to be calculated across the discontinuous interface. The finite-volume solver uses these edges to construct median dual volumes, as seen in Figure 1. Calculations are performed at the true center of gravity of each control volume. Surface elements are refined to allow for median dual volume construction. Data structures for refined surface element and control volume connectivity are maintained in dynamic grid situations. While this reconstructed control volume method is theoretically fully conservative and thus might provide more accurate results, the extrusion method of Blades and Marcum for sliding interfaces is used in the Tenasi solver at the time of this writing and for this study.

Steijl and Barakos (2009) have conducted simulations of both the Georgia Tech and ROBIN simplified rotorcraft geometries in forward flight, using a sliding-plane method on structured grids. Results from their study are compared with this simulation as well as the experimental data from Freeman and Mineck (1979) and Brand et al. (1989).

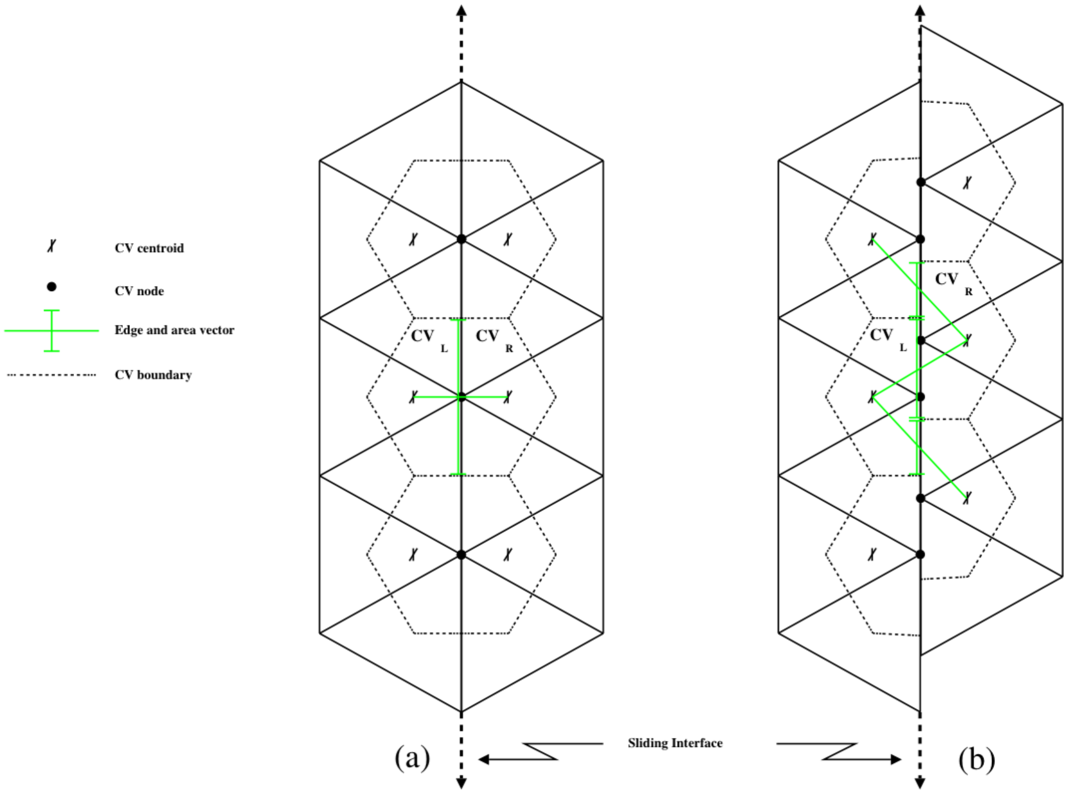


Figure 1 Sliding interface schematic for two-dimensional field elements, reconstructed control volume method

The Unstructured Solver: Tenasi

Tenasi, developed at the UTC SimCenter, is an unstructured, finite-volume, three-dimensional flow solver for a system of coupled partial differential equations that describe fluid flow, known as the Reynolds Averaged Navier-Stokes (RANS) equations. There is currently no analytical solution for them. The full set of equations and methods used to solve them numerically are listed and described in Chapter III. The Tenasi software allows parallel computation over decomposed domains of the mesh, with communication of geometry information and volume surface flux computations occurring across domain boundaries via the Message Passing Interface (MPI) standard. It employs arbitrary Mach preconditioning to enable resolution of essentially incompressible (low Mach number) flows as well as high speed (Mach number > 1) flows (Sreenivas et al., 2005).

Sliding Interfaces

For a rotorcraft simulation, the blades must obviously rotate relative to the body. This presents a difficulty when the flow field is discretized – the mesh blocks near boundary surfaces must move with them in order to maintain conservation of fluid properties that are quantified in the mesh itself. This can be accomplished using either vector arithmetic in Cartesian coordinates, or by recasting the equations in a relative, rotating frame. The problem, then, is what to do at the block interfaces. One approach is similar to that used at boundaries of subdomains created from a large mesh to distribute computational load across cores of a supercomputer – i.e. using “ghost nodes”. From Brand, McMahon, and Komerath (1989):

> “Rotational motion is accomplished by rigidly rotating a subdomain representing the moving component. At the subdomain interface boundary, the faces along the interfaces are extruded into the adjacent subdomain to create new volume elements and provide a one-cell overlap. These new volume elements close the control volumes for the nodes on the interface surface and allow a flux to be computed across the subdomain interface. An interface flux is computed independently for each subdomain. The values of the solution variables and other quantities for the nodes created by the extrusion process are found by interpolation.”

Tenasi currently employs a cell extrusion method for calculating fluid flow across sliding grid interfaces, described in Blades and Marcum (2005):

> The method is implemented into a parallel, node-centered finite volume, unstructured viscous flow solver. The rotational motion is accomplished by rigidly rotating the subdomain representing the moving component. At the subdomain interface boundary, the faces along the interface are extruded into the adjacent subdomain to create new volume elements and forming a one-cell overlap (*See Figure 2*)... the

extrusion is done so that the interpolation will maintain information as localized as possible.

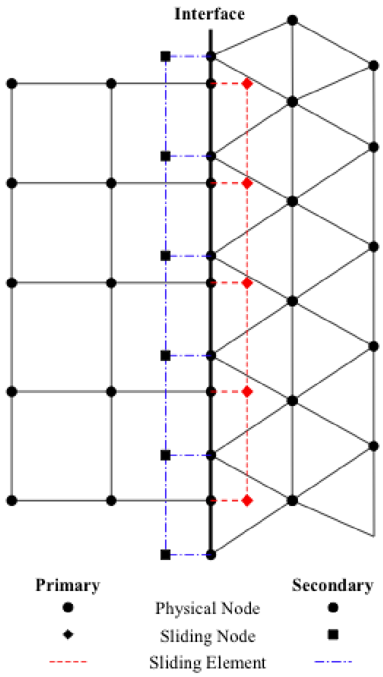


Figure 2 Two-dimensional overlapping sliding interface

To avoid confusion, the entities referred to above as "subdomains" we will call "blocks", since the term "subdomain" is also used to refer to field mesh decomposition units for distributed parallel processing. Since the rotor blocks are rotated in the unsteady simulation, each block and its boundary surface must be axisymmetric in order to maintain volume closure of the computational mesh at the sliding boundaries. Thus, each block boundary surface was created by rotating an arc through a full circle about the rotor axis. Each block boundary was then copied in place, with one copy of the surface assigned to the rotating block, and the other assigned to the enclosing block. It should be noted that exact copies of the meshed block boundary surfaces are not necessary for the extrusion sliding interface method, and unaligned multiblock surface meshes will produce valid solver results as long as they are geometrically co-located within acceptable tolerances. In fact, mirroring around the Y-axis origin was necessary in Pointwise in order to create co-located surface domains, since this software attempts to prevent the creation of duplicate elements. The blocks for each rotating component of the ROBIN model are illustrated in Figure 3.

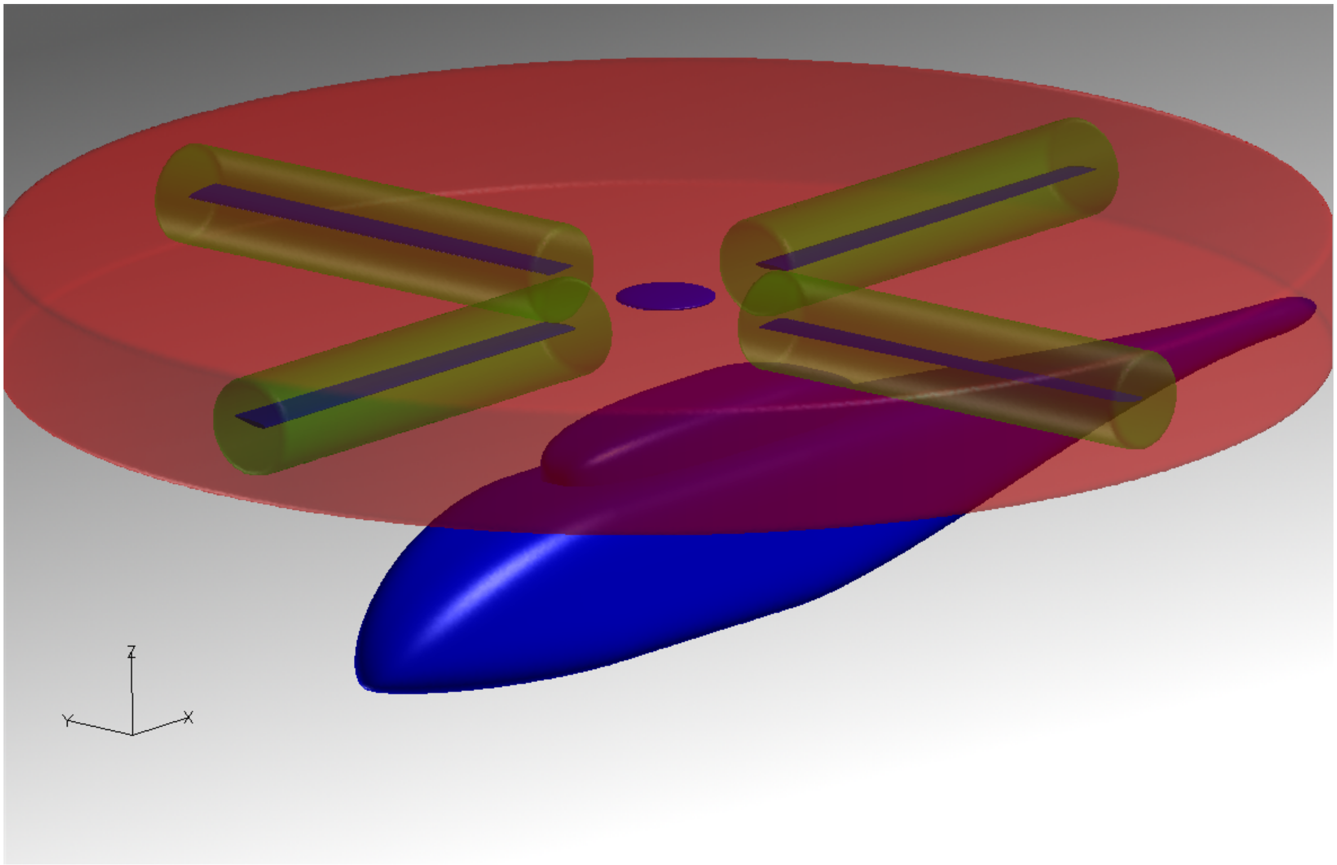


Figure 3 Blocks for rotation of ROBIN rotor (red) and cyclic pitching of blades (yellow)

In the unsteady simulation, the blocks containing solid bodies (solid boundary surfaces) are rotated relative to each other. Rotation of the entire rotor block (red, in the figure above) relative to the body and farfield can be accomplished using relative frame rotation (Ghosh, 1996) in addition to mesh movement for the cyclic pitching of blades, or through absolute frame rotation, i.e. mesh movement with submotion specified for the blade-enclosing blocks. A comparison of these two methods to verify their equivalence (or rather, lack thereof in the current implementation when block submotion) is performed on the isolated rotor test case. The cyclic pitching of the blade blocks (yellow, above) relative to the rotor block is achieved using rigid mesh movement, i.e. coordinate manipulation of the mesh nodes in memory.

Extrusion

The new volume elements that are extruded from the sliding surfaces into each adjacent subdomain have their extrusion distance determined relative to the cell size of the neighboring subdomain near its surface. In this way, the method avoids creating extruded volumes that extend past several neighboring cells in the extrusion direction - see Figure 4. This allows for data closer to the subdomain surface to be used in constructing surface fluxes for control volumes around the subdomain surface nodes and, therefore, more accurate results for trans-surface flow. Figures 2 and 4 illustrating the extrusion are from Blades and Marcum (2005). In these simulations, the sliding interface does not intersect solid boundaries. However, the method allows for solid boundaries by extruding interface edges to quadrilateral elements that will then have appropriate boundary conditions applied. These faces will close control volumes for the interface nodes and their solution flux evaluations.

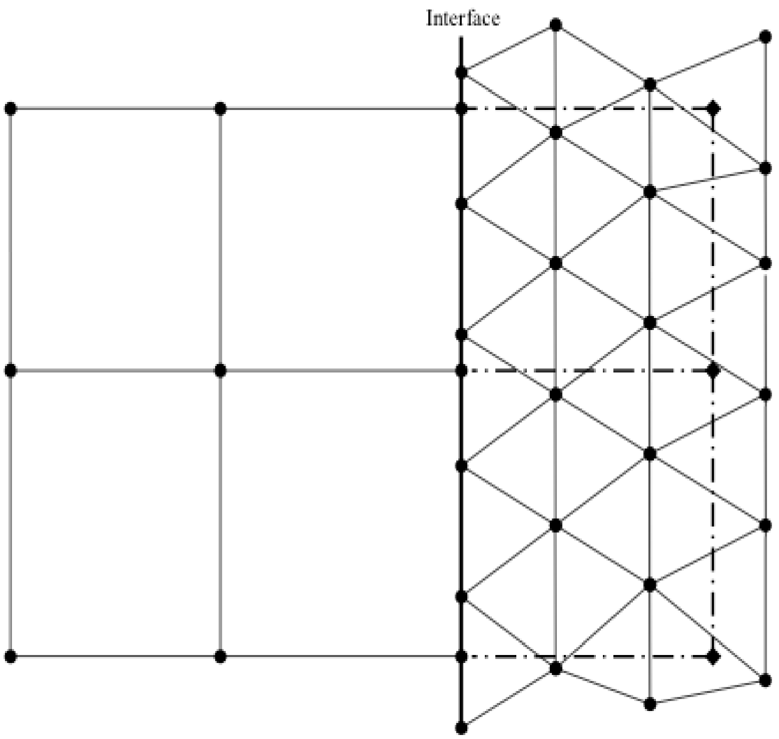


Figure 4 Suboptimal extrusion distance (nodes near surface are ignored)

Variable Value Interpolation

Once the sliding nodes are extruded, and the subdomains are rotated, the containing host element is found through a nearest-neighbor search using the previous host element as an initial guess (Löhner and Ambrosiano 1990). The variable values $\vec{Q}$, $\Delta\vec{Q}$, etc. are then interpolated from the nodes in the host element.

$$\hat{u}=\sum_{j=1}^{n}\phi_j u_j \tag{1}$$

Above, $\hat{u}$ is the interpolated quantity, ϕ_j are the finite element shape functions or weighting functions for the host element, u_j is the value of the quantity at the nodes of the host element, and j is an index over the nodes of the host element. For a tetrahedral host element, ϕ_j are isoparametric shape functions, while for hexahedra, pyramids, and prisms, where the shape functions would require solution of a cubic polynomial, ϕ_j are inverse distance weighting functions. Extruded node values are updated in every timestep, regardless of update frequency for other nodes, e.g. nodes on any parallel block boundaries. This ensures tight coupling of blocks across interfaces.

CHAPTER II

MODEL CONSTRUCTION AND METHODOLOGY

Isolated Rotor Geometry

The isolated rotor that was tested had blades with a NACA 0012 cross-section, untwisted and untapered. The rotor diameter was 7.5 feet or 2.286 m. This gives a circumference of *π*d/2* = 23.562 feet = 7.1817 m. The blades were offset from center (giving a "root cutout") of 5 inches or 5.5 % of the diameter (0.127 m). The aspect ratio of each blade is 6, giving a blade chord of 0.625 feet or 0.1905 m. The collective pitch for the rotor blades in the grid as constructed was 8 degrees. The farfield was generated as a cylinder having a diamter equal to 3 times that of the rotor. The hub is a cylindrical surface meant to approximate the rotor test assembly. The full rotor geometry can be seen in Figure 5.

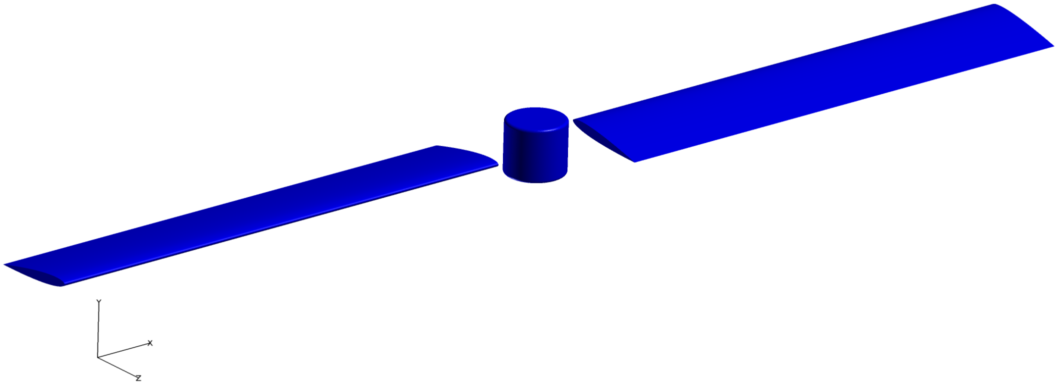


Figure 5 Isolated rotor geometry

Definition of the ROBIN Body

The coordinates of the ROBIN body are defined by super-ellipse equations. For a given nondimensional body longitudinal station, the nondimensional coordinates of the cross section are obtained from the analytic functions for the model height (H), width (W), camber (Z0), and elliptical power (N). Each function has the same form; only the eight coefficients (C1 to C8) differ. The body is divided into four regions and the pylon is divided into two regions. Separate coefficients are used for each of these six regions in the four functions. The form of these functions is defined as follows:

$$\begin{bmatrix} H(x/l) \\ W(x/l) \\ Z_0(x/l) \\ N(x/l) \end{bmatrix} = C_6 + C_7\left(C_1 + C_2\left(\frac{x/l + C_3}{C_4}\right)^{C_5}\right)^{1/C_8} \tag{2}$$

The Cartesian coordinates at a given body station *x/l* are defined in terms of polar coordinates. The nondimensional radial coordinate for the cross section is defined as follows:

$$r = \left(\frac{\left(\frac{H}{2}\frac{W}{2}\right)^N}{\left(\frac{H}{2}\sin\phi\right)^N + \left(\frac{W}{2}\cos\phi\right)^N}\right)^{1/N} \tag{3}$$

From the radial coordinate, the nondimensional coordinates *y/l* and *z/l* on the cross section can be obtained from equation 4 by varying ϕ from 0 to 2π. It should be noted, here, that the listing of these formulas in Mineck and Gorton (2000) is incorrect, having the sine function in *y/l* and cosine in *z/l*. This results in a subtle difference in the body shape that was only noticed on direct comparison of the 3-D model in software with the body as generated by previous codes (Schweitzer 1999). In fact, in the study presented here, many simulations of the unsteady case (having a rotor) were run with the wrong body, using different parameters in an attempt to isolate potential sources of disagreement in results.

$$\begin{aligned} y/l &= r\cos\phi \\ z/l &= r\sin\phi + Z_0 \end{aligned} \tag{4}$$

A listing of the code used to generate the model for this study can be found in the appendices. The values of the coefficients are listed in tables below. For further details, see Mineck and Gorton (2000).

Table 1 Coefficients to define body shape

Function	**0.0 < x/l < 0.4**							
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-.4	.4	1.8	0.0	.25	1.8
W	1.0	-1.0	-.4	.4	2.0	0.0	.25	2.0
Z_0	1.0	-1.0	-.4	.4	1.8	-.08	.08	1.8
N	2.0	3.0	0.0	.4	1.0	0.0	1.0	1.0
	0.4 < x/l < 0.8							
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
W	.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Z_0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.8 < x/l < 1.9							
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-.8	1.1	1.5	.05	.2	.6
W	1.0	-1.0	-.8	1.1	1.5	.05	.2	.6
Z_0	1.0	-1.0	-.8	1.1	1.5	.04	-.04	.6
N	5.0	-3.0	-.8	1.1	1.0	0.0	0.0	0.0
	1.9 < x/l < 2.0							
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-1.9	.1	2.0	0.0	.05	2.0
W	1.0	-1.0	-1.9	.1	2.0	0.0	.05	2.0
Z_0	.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 2 Coefficients to define pylon shape

Function	**0.4 < x/l < 0.8**							
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-.8	.4	3.0	0.0	.145	3.0
W	1.0	-1.0	-.8	.4	3.0	0.0	.166	3.0
Z_0	.125	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.8 < x/l < 1.018							
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
H	1.0	-1.0	-.8	.218	2.0	0.0	.145	2.0
W	1.0	-1.0	-.8	.218	2.0	0.0	.166	2.0
Z_0	1.0	-1.0	-.8	1.1	1.5	.065	.06	.6
N	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

ROBIN Rotor Geometry

The ROBIN model for this study uses a four-bladed rotor. The full rotor suspension, propulsion, and trim mechanisms as used in the NASA Langley experiments are not modeled. However, a simple ellipsoidal hub is included for approximation of flow obstruction through the rotor center.

Each of the four rotor blades is identical. The cross-section is a standard NACA 0012 airfoil extruded for the length of the blade and given a linear -8° twist starting from the blade root. The blade tips, for a collective pitch of 0º, are set at 0° angle of attack with respect to the direction of travel or tip path plane (with no flapping). The blades are untapered and rectangular in planform. The root cutout is at 24% of the radius from the center, as shown with blades and hub in Figure 6. The rotor radius R as constructed is 33.876 inches, giving a blade chord c of 2.61 inches, and a blade length (with cutout) of 25.75 inches. The body length is $2R$ = 78.7 inches. The rotor aspect ratio, $AR = R/c$ is approximately 13. The rotor geometry as modeled includes a 2 inch offset starboard of fuselage centerline. For forward flight, the entire rotor disk block was tilted forward 3 degrees from the hover condition, where the rotor axis of rotation and the Z axis are parallel, and the mesh block having the rotor disk and ROBIN body as interior surfaces was reinitialized. Note that, due to blade twist, collective pitch θ_0 is measured at 0.75 R along the blade by common convention.

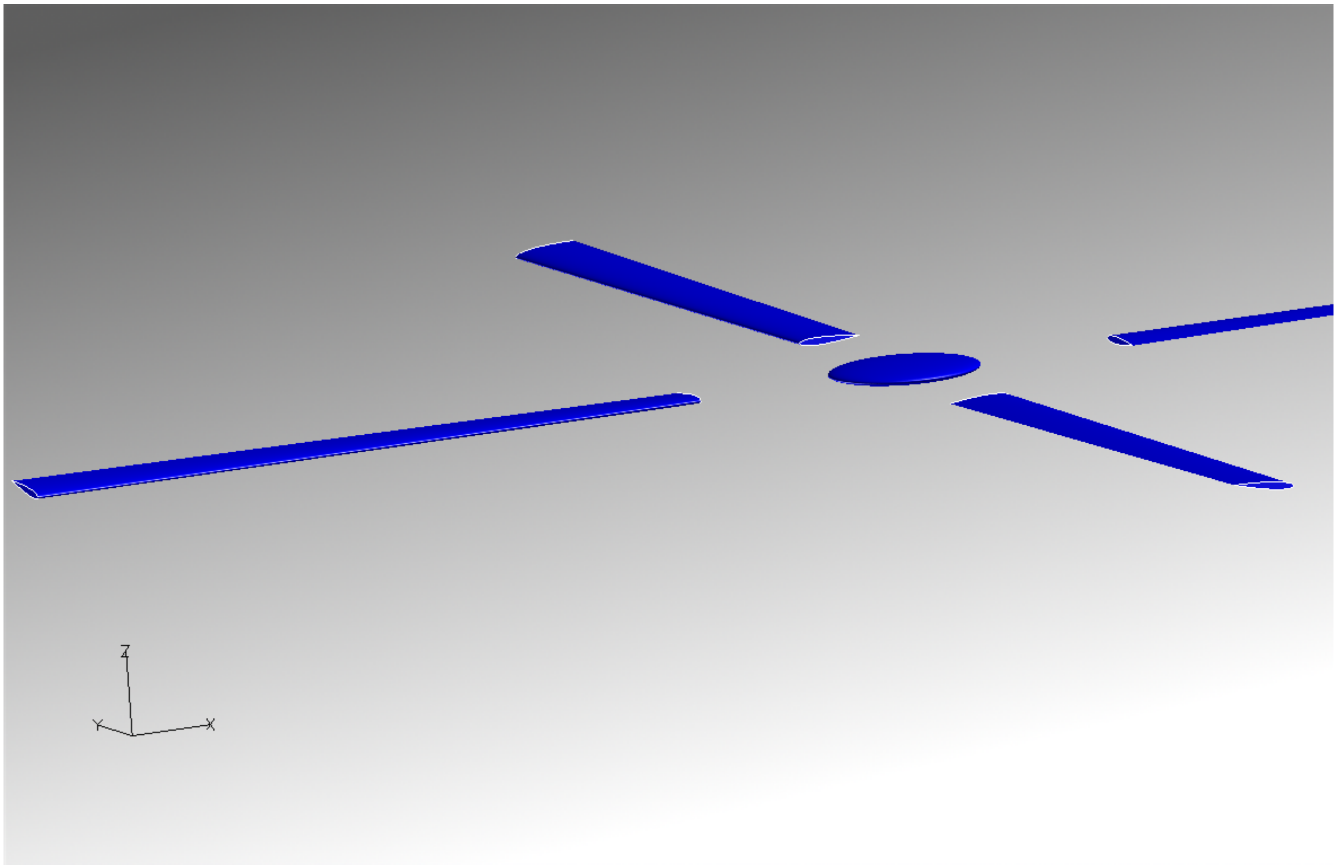


Figure 6 ROBIN rotor geometry

Grid Generation

Meshes which discretize field space for the computational solution of the governing equations were generated using Pointwise® software by Pointwise, Inc. Pointwise supports the generation of structured, unstructured, and hybrid meshes. “The grid in each block can either be structured, unstructured, or hybrid. A structured grid consists entirely of quadrilateral (2D) or hexahedral (3D) cells that have been arranged in an ordered IxJxK array. An unstructured grid consists of triangles (2D) or tetrahedral, pyramid, and prism cells (3D) having no implicit order." (Pointwise 2010)

All flapping motions were modeled as cyclic pitching according to the coupling equation given as a truncated Fourier series $\theta=\theta_0-\theta_{1c}\cos\varphi-\theta_{1s}\sin\varphi-\ldots$ (where θ is total blade pitch angle, and θ_0, θ_{1c}, and θ_{1s} are collective, lateral, and longitudinal cyclic pitch respectively) (Webster 1994, p. 31). In addition, rotation of the blades and hub as a rotor must be modeled. To that end, in accordance with the sliding interface methodology described above, each moving surface had cylindrical interfaces defined that would allow movement around an axis of rotation. All sliding interface surfaces were generated as cylinders with rounded corners – one for cyclic pitching of each of the blades and one for rotation of the rotor itself within the farfield. These can be seen for the isolated rotor in Figure 7.

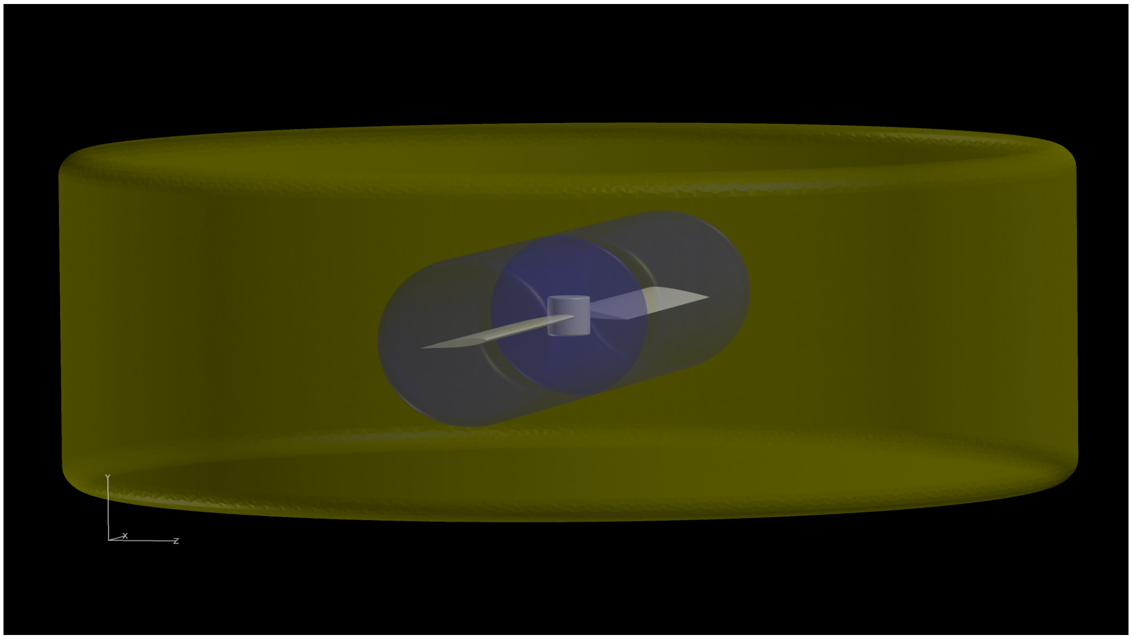


Figure 7 Rotation sliding interfaces for isolated rotor

Surface and Volume Meshing

Each surface was modeled according to specifications (see above) and imported into Pointwise as a “database”. These surfaces were quilted to form water-tight bodies – having no gaps between surface edges bounding the volume – and outlined with connectors. Surfaces bounded by these connectors have a “domain” mesh created on their topology, either through trans-finite interpolation (TFI) in the case of structured grids, or via Delaunay triangulation in the case of unstructured grids (see Figure 8). These surfaces are then associated to form adjacent faces of three-dimensional blocks. In an unstructured mesh, points allocated to the interior of these blocks are connected to form tetrahedra (or prisms and/or pyramids extruding from quadrilateral surface elements, in the case of some structured/unstructured hybrid meshes).

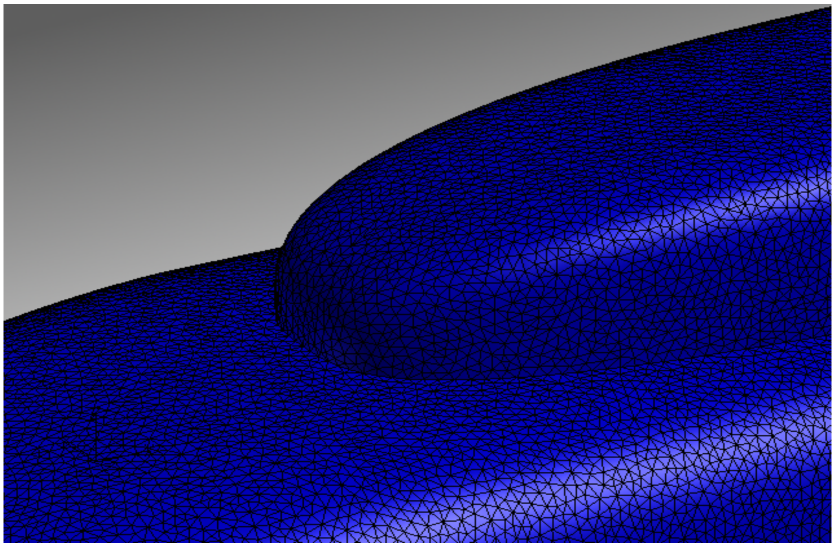


Figure 8 ROBIN surface mesh

Isolated Rotor Case Mesh Generation

A single blade was extruded from a NACA 0012 outline used as a spline surface. A sliding interface for rotation of the rotor assembly in the freestream and sliding interfaces for pitching rotation of each blade was constructed for the isolated rotor. The interfaces were axisymmetric cylinders with rounded edges to facilitate computation of boundary cell face fluxes in the normal direction. Tighter control of point spacing on the blade surfaces necessitated the diagonalization of a structured (quadrilateral) mesh with regular spacing. This allowed more points to be clustered in areas of higher expected gradient - near the leading and trailing edges of the blades - for better numerical resolution of flow phenomena (see Figure 9). The volume between triangulated surfaces was initialized with an unstructured tetrahedral mesh (see Figure 10). The total number of nodes on each blade surface domain was 121,594, while each cyclic pitching block interface had 56,179 nodes, and the rotor block interfaces had 26,595. The total count of nodes for all blocks in the mesh was 1,993,605 before viscous layers were added (discussed below).

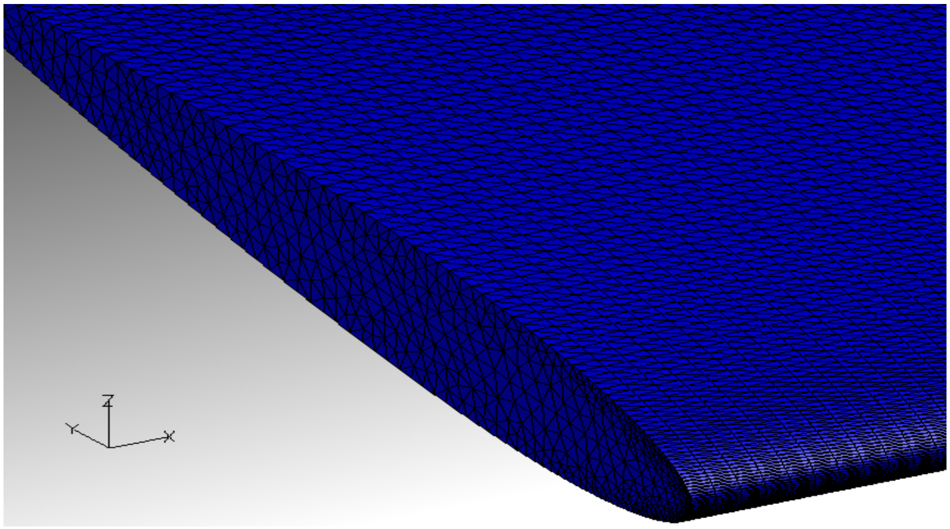


Figure 9 Blade surface mesh

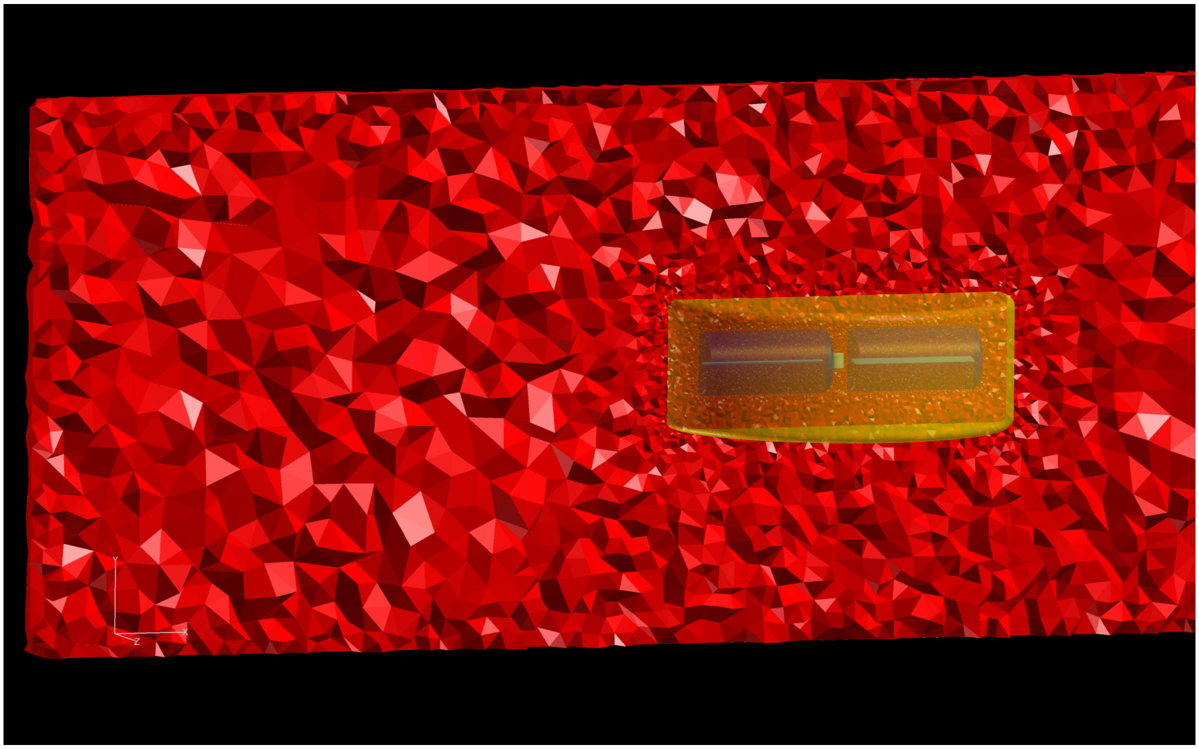


Figure 10 Full grid for isolated rotor case

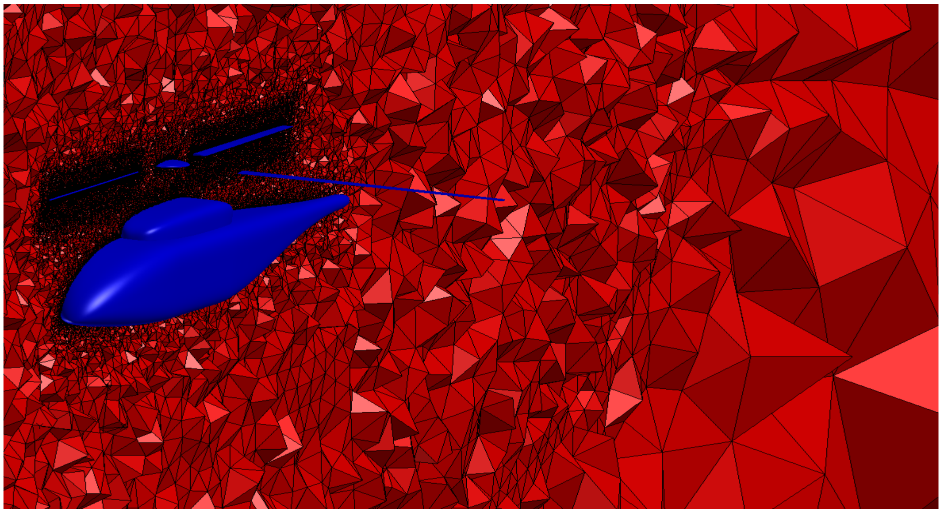


Figure 11 “Crinkle cut” through tetrahedral volume mesh arounnd rotor

ROBIN Case Mesh Generation

A Delaunay-triangulated mesh was created on the ROBIN body surfaces (see above). The ROBIN body surface mesh had 73,675 nodes. For ROBIN body only cases in steady-state, the body surface mesh was divided on the symmetry plane and a block was created from the body surface to a spherical “farfield” boundary with a radius of 26 m (the body length being 2 m). The entire mesh for the steady-state case had 1,194,304 nodes. For the cases having a rotor, each blade surface was constructed and meshed similar to the procedure for the isolated rotor case above. Sliding interfaces for the cyclic pitching blocks around the blades and the rotor revolution block were constructed as cylinders with rounded corners, and blocks using solid and sliding interface boundaries were defined and initialized with an unstructured tetrahedral mesh (see Figure 11). Each blade surface domain was meshed with 132,768 nodes. The cyclic pitching block sliding interfaces each had 16,406 nodes, and the rotor sliding interface had 328,175. The total number of nodes for all volume blocks in the mesh with rotor was 5,879,513.

Viscous Layering

In a fluid flow model that accounts for viscosity, a “no-slip” condition is imposed at the solid boundary surfaces. This condition specifies that fluid velocity magnitude relative to the surface is zero, and that within a “boundary layer” extending away from that surface, internal friction and shear stresses will induce a velocity gradient (see Figure 12 and equation 5).

$$\tau=\mu\frac{\partial u}{\partial y} \tag{5}$$

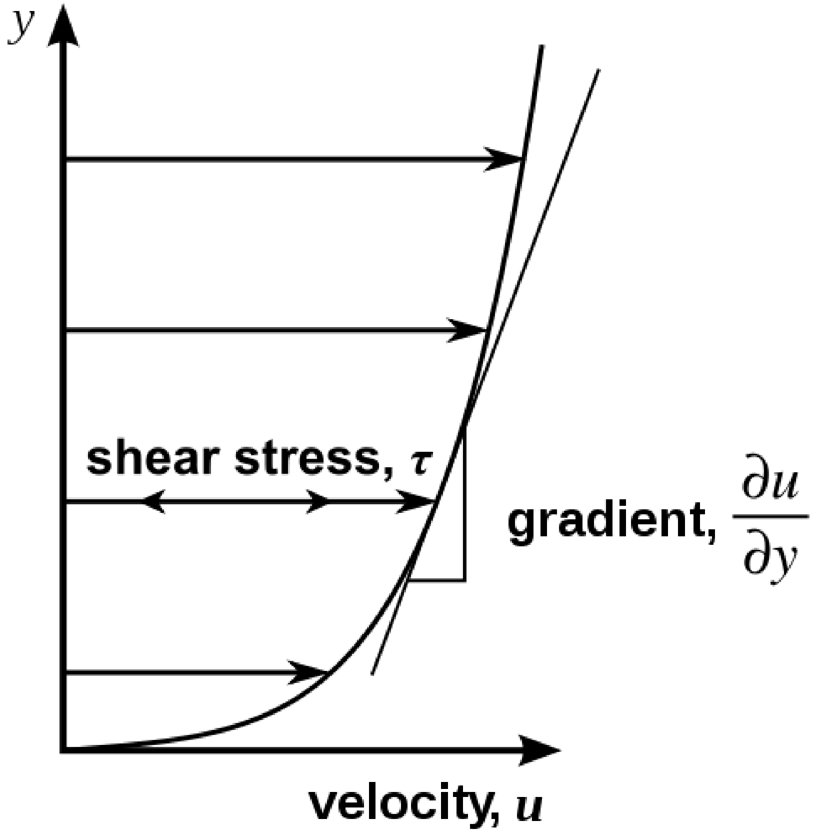


Figure 12 Viscous shear stress (Image source: http://en.wikipedia.org/wiki/File:Laminar_shear_flow.svg)

Because of this gradient, mesh layers that will resolve viscous effects must have very small spacing in the direction perpendicular to the boundary surface. These layers are very difficult to generate in an unstructured mesh using current tools. At the SimCenter, Parallel Viscous Layer Insertion (PVLI) has been developed by Dr. Steve Karman et al. in order to insert these layers on specified surfaces (Karman 2007). The existing mesh is treated as a visco-elastic material using the linear-elastic equations of structural mechanics, and the innermost layer of it is moved away from the surface. New layers are then inserted from the mesh towards the surface, with the perpendicular length specified according to a geometric progression factor to be specified by the user. The results for the ROBIN model can be seen in Figures 13 and 14, and for the isolated rotor case in Figure 15. In both cases, the expected Reynolds number for transonic flow recommended an initial wall or y-direction spacing on the order of 1e-05, using a community rule of thumb. The growth rate of the spacing in the y+ direction was 1.2, and P_VLI was able to insert between 12 and 40 layers on all viscous surfaces. After viscous layer insertion, the isolated rotor mesh had 6,875,660 nodes. The viscous ROBIN mesh had 15,513,143.

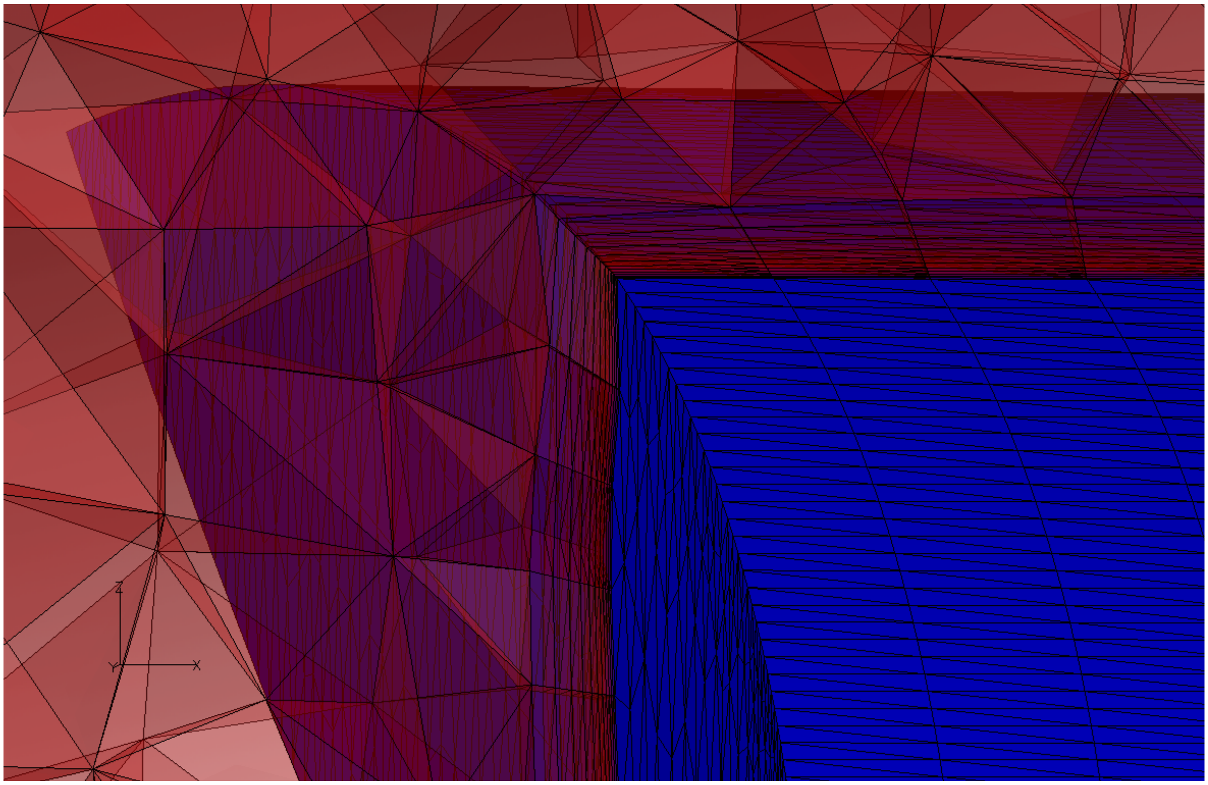


Figure 13 PVLI viscous layer packing around ROBIN blade surface

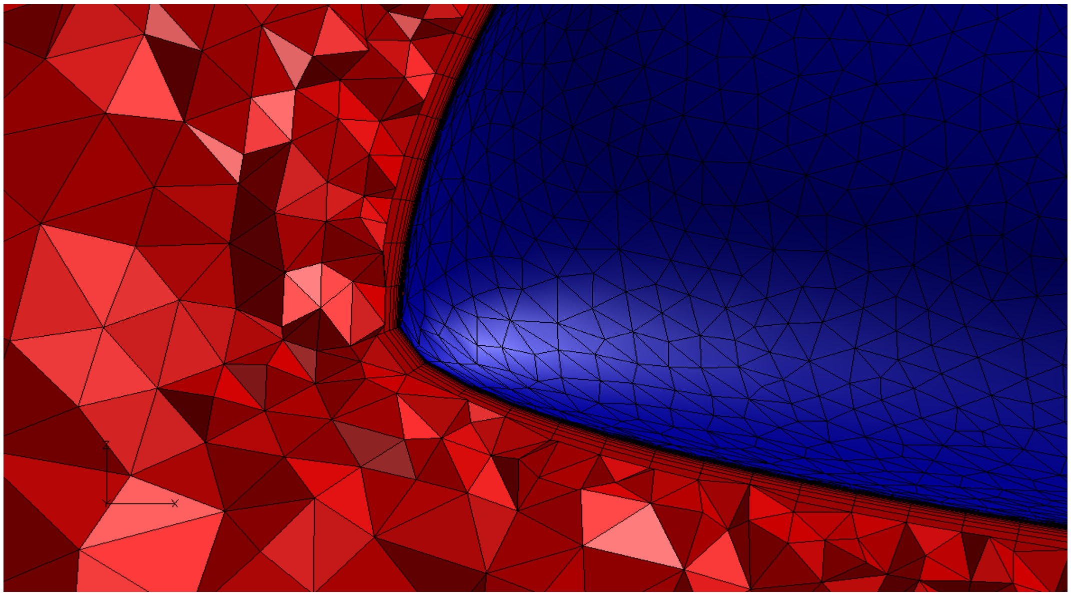


Figure 14 PVLI viscous layer packing around ROBIN body surface

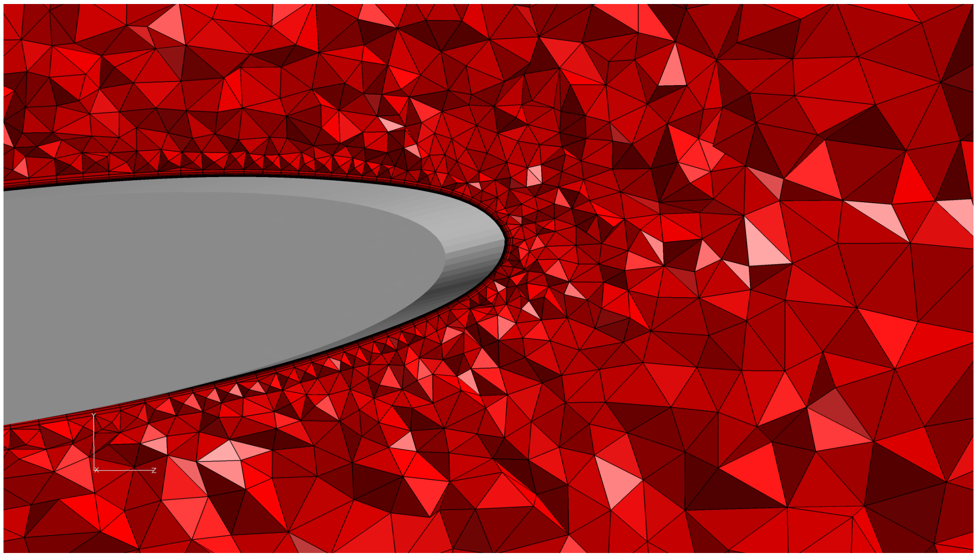


Figure 15 Viscous layers on isolated rotor blade

CHAPTER III

COMPUTATIONAL FLUID DYNAMIC MODEL

Governing Equations

$$\frac{\partial}{\partial t}\oiiint_{\Omega} Q dV + \oiint_{d\Omega} \vec{F}(Q)\cdot\hat{n}\, dA = \frac{M_{\infty}}{Re}\oiint_{d\Omega} \vec{G}(Q)\cdot\hat{n}\, dA \tag{6}$$

$$Q = \left[\rho \;\; \rho u \;\; \rho v \;\; \rho w \;\; \rho e_t\right]^T \tag{7}$$

$$\vec{F} = \begin{bmatrix} \rho(u-V_x) \\ \rho u(u-V_x)+P \\ \rho v(u-V_x) \\ \rho w(u-V_x) \\ \rho h_t(u-V_x)+V_x P \end{bmatrix}\hat{i} + \begin{bmatrix} \rho(v-V_y) \\ \rho u(v-V_y) \\ \rho v(v-V_y)+P \\ \rho w(v-V_y) \\ \rho h_t(v-V_y)+V_y P \end{bmatrix}\hat{j} + \begin{bmatrix} \rho(w-V_z) \\ \rho u(w-V_z) \\ \rho v(w-V_z) \\ \rho w(w-V_z)+P \\ \rho h_t(w-V_z)+V_z P \end{bmatrix}\hat{k} \tag{8}$$

$$\vec{G} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ u\tau_{xx}+v\tau_{xy}+w\tau_{xz}-q_x \end{bmatrix}\hat{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ u\tau_{yx}+v\tau_{yy}+w\tau_{yz}-q_y \end{bmatrix}\hat{j} + \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{zx}+v\tau_{zy}+w\tau_{zz}-q_z \end{bmatrix}\hat{k} \tag{9}$$

$$\tau_{xx} = (\mu+\mu_t)\left(2u_x - \frac{2}{3}\nabla\vec{V}\right) \tag{10}$$

$$\tau_{yy} = (\mu+\mu_t)\left(2v_y - \frac{2}{3}\nabla\vec{V}\right) \tag{11}$$

$$\tau_{zz} = (\mu+\mu_t)\left(2w_z - \frac{2}{3}\nabla\vec{V}\right) \tag{12}$$

$$\tau_{xy}=\tau_{yx}=(\mu+\mu_t)(u_y+v_x) \tag{13}$$

$$\tau_{xz}=\tau_{zx}=(\mu+\mu_t)(u_z+w_x) \tag{14}$$

$$\tau_{yz}=\tau_{zy}=(\mu+\mu_t)(v_z+w_y) \tag{15}$$

$$\vec{q}=-\frac{\frac{\mu}{Pr}+\frac{\mu_t}{Pr_t}}{(\gamma-1)}\nabla T \tag{16}$$

The Prandtl number *Pr* above is between 0.7 and 0.8 for air, while the turbulent Prandtl number Pr_t has an average value of 0.85 (http://en.wikipedia.org/wiki/Prandtl_number). The specific heat ratio *γ* is 1.401 for air at 15ºC and Standard Atmospheric Pressure at sea level.

The equations presented above are the three-dimensional compressible Navier-Stokes equations in conservative integral form and Cartesian coordinates for a control volume *Ω* bounded by surface *dΩ* and moving with velocity $\vec{V}$ (*Tenasi Documentation* 2012). For an ideal gas, the equations are closed with the applicable equation of state (17) and the appropriate boundary conditions, described below.

$$p=\rho R T \tag{17}$$

Computational Methods

Tenasi uses several techniques to allow for numerical solution of a discretized form of the continuous governing equations. In particular, it employs a node-centered finite-volume approach, in which solution variables are stored at each node and associated with a corresponding control volume constructed as a median dual as follows: each edge connected to a node has a midpoint; these are connected to the geometric center of their corresponding incident faces and elements (triangle, quadrilateral, tetrahedron, hexahedron, etc.). Each midpoint and centroid then form the nodes for faces of the median dual volume. The two-dimensional analogue can be observed in Figure 16 as the polyhedron bounded by a dotted red line around the central node.

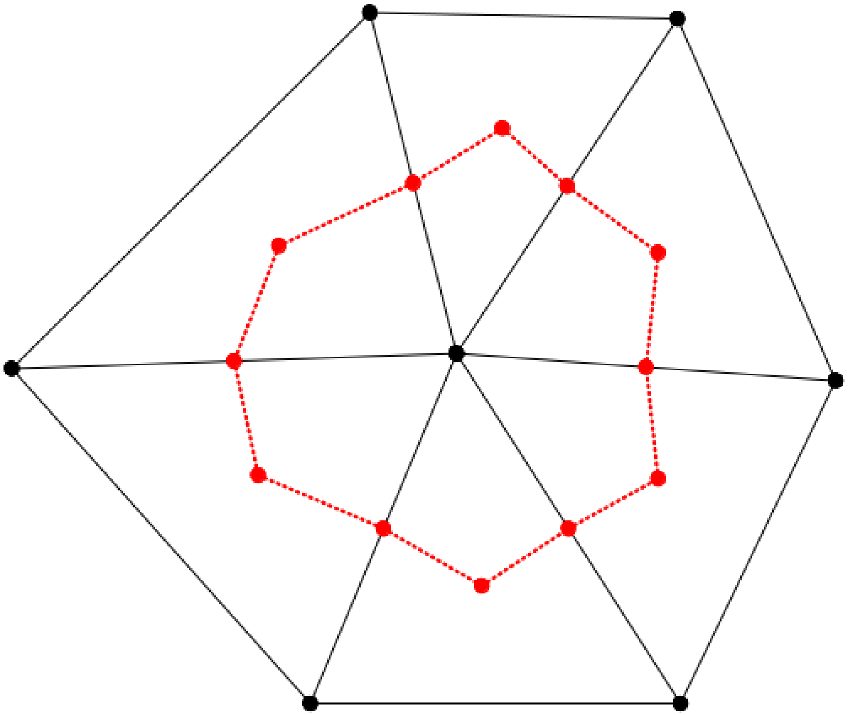


Figure 16 Control volume for center node, constructed as median dual

Under the Roe scheme, the surface fluxes are computed as numerical approximations along these control volume boundaries using solution variable values from the nodes on each side of the control volume boundary surface as input to a one-dimensional Riemann problem:

$$\Phi=\frac{1}{2}(F(Q_L)+F(Q_R))-\frac{1}{2}\tilde{A}(Q_R,Q_L)(Q_R-Q_L) \qquad (18)$$

where $\tilde{A}=\tilde{R}\tilde{\Lambda}\tilde{R}^{-1}$ and *R* is constructed from the right eigenvectors of the flux Jacobian, while $\tilde{\Lambda}$ is a square matrix having the eigenvalues along the diagonal and zero elsewhere (Hyams 2003).

Node values are designated “Left” and “Right” by convention, using a winding rule for consistency. The following averages are then used for values at the volume face:

$$\bar{\rho}=\sqrt{\rho_L \rho_R} \tag{19}$$

$$\bar{u}=\frac{\sqrt{\rho_L}u_L+\sqrt{\rho_R}u_R}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{20}$$

$$\bar{v}=\frac{\sqrt{\rho_L}v_L+\sqrt{\rho_R}v_R}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{21}$$

$$\bar{w}=\frac{\sqrt{\rho_L}w_L+\sqrt{\rho_R}w_R}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{22}$$

$$\bar{H}=\frac{\sqrt{\rho_L}H_L+\sqrt{\rho_R}H_R}{\sqrt{\rho_L}+\sqrt{\rho_R}} \tag{23}$$

The governing equations are discretized using a finite volume technique, wherein the surface integrals are approximated by a quadrature over the surface of the control volume of interest using the averages above at each edge midpoint. The resulting formulation for a given node *n* takes the form

$$\frac{\partial q_n}{\partial t}+\Re_n=0 \tag{24}$$

wherein q_n represents variable values at the nodes, and spatial terms are entirely represented within the residual $\Re$, which contains all contributions from approximations to the viscous and inviscid terms (Hyams 2003). By convention, this is moved to the “right-hand side” of the equations, while the temporal derivative (i.e. the solution update term for each timestep) forms the left. A fully implicit backwards Euler approximation is used to calculate the temporal derivative i.e. the solution update at each time step, where the linear system is solved using LU decomposition and Symmetric Gauss-Seidel algorithms (*Tenasi Documentation* 2012). For unsteady simulations, where accuracy within each timestep is of importance (in addition to the iterative convergence of the solution update), Newton's method is used to solve for the spatial residual at the new timestep *n+1* by linearizing about the known solution Q^n. (Hyams 2003).

In both simulations, the Menter SAS one-equation turbulence model is used for calculating the Reynolds stress terms necessary for closure of the RANS equations. The diffusive terms are discretized using the same finite volume method as the viscous terms in the mean flow (Sreenivas et al. 2005). The turbulence model is solved independently and combined with the convective solution in each timestep update. This loosely coupled solution methodology allows for the implementation and use of several different turbulence models in the flow solver in addition to the one employed here.

Solution Convergence

A decreasing value for the norm of the residual (i.e. the finite differences of variable values with respect to the time interval, or the approximate magnitude of the solution update at each timestep) determines convergence in steady-state models. Results from the simulation of the isolated ROBIN fuselage were run to steady-state convergence. However, the rotor-body interaction models in this study involve transient relative grid motion and therefore will not approach a true steady state. Nonetheless, steady calculations are performed for the first several hundred time steps (two rotor revolutions, in the ROBIN case) in order to establish the initial conditions for the flow field with respect to the rotor in relative grid motion. In this case, specifying to Tenasi that “steady” calculations should be performed refers simply to the lack of extra calculations performed in unsteady models in order to more accurately compute the solution update for each time step. In previous research with Tenasi it has been determined that this can overcome instability that might appear in the initial timesteps of a numerical simulation and accumulate to invalid or unrepresentable floating-point values.

After initialization, the cases are restarted with the solver directed to begin performing unsteady calculations using Newton iterations to drive convergence of the residual within each time step, as described above. Convergence of the overall solution, then, is designated as the lack of a significant fluctuation in rotor thrust coefficient over a full rotor revolution from one revolution to the next.

Parameters

For this study, the time step ($\Delta\tau$) was specified in order to achieve one step per degree of rotor revolution. For the isolated rotor test case in hover, the tip Mach number was 0.877 (298.43 m/s) or 2493.3 rpm, and therefore the timestep $\Delta\tau$ = 6.685e-05 seconds. For the isolated rotor test case in forward flight, a tip Mach number of 0.6713 is equivalent to a rotation frequency near 1908.5 rpm, and the timestep to achieve one degree of rotation was $\Delta\tau$ = 8.73e-05 seconds. For the ROBIN rotor in both hover and forward flight, at $V_\infty = \Omega R$ = 591 ft/sec and R = 2.823 ft, rotation frequency Ω = 209.35 radians/sec ≈ 2000 rpm. The time step was approximately $\Delta\tau$ = 8.33e-05 seconds.

In both cases, the first two revolutions, or 720 time steps, were run as a steady-state problem using local time stepping to assist in stability, as above, with a Courant number CFL = 15.0, ramped from 1.0 over 1000 time steps. The CFL number is used to control the magnitude of the local time stepping in steady-state calculations, where the solution in each control volume is advanced as far as possible with respect to numerical stability given local conditions in that volume. The time step is calculated by analogy with a scalar hyperbolic equation based on the relative magnitudes of the flux Jacobian eigenvalues (Hyams 2003). Subsequently, time advancement iterations were performed as unsteady with Newton's method and dual time stepping. Dual time stepping refers to the technique of allowing internal Newton iterations and linear sub-iterations to use local time stepping, in order to improve stability, while using a fixed step in the outer time-marching loop to maintain unsteady time accuracy. When used in conjunction with minimum time stepping, the CFL number is used as a multiplier to generate a timestep $\Delta\tau$ for the pseudo-time derivative, but with local time stepping, the computed local time step is used for $\Delta\tau$ (*Tenasi Documentation* 2012). Fluxes were limited using the Barth method to avoid accumulation of numerical error, i.e. spurious oscillations, from driving the solution to divergence. Inputs of coordinates for the center of rotation and axis of rotation, as well as rotation speed for both relative and absolute motion and cyclic pitch coefficients were provided in the boundary conditions input file.

Boundary Conditions

For all simulations, blade, fuselage, and hub surfaces were treated as solid viscous adiabatic – that is, exerting frictional forces on the surrounding fluid, with no heat transfer, and no-slip (fluid velocity at the wall constrained to the local velocity of the surface). Sliding interface surfaces and their partner surfaces were specified as such, with the appropriate calculations performed as described above. Farfield boundary conditions imposed freestream flow on the outside of the domain. For all cases but isolated rotor forward flight, flow was parallel to the x-axis. For the isolated rotor forward flight case, the flow vector was rotated 2º forward about the y-axis in order to simulate a -2º tilt of the rotor tip path plane and thrust vector relative to the freestream or “translational” velocity vector, depending on one's reference frame.

CHAPTER IV

RESULTS

Isolated Rotor Model

The isolated rotor was run in both hover and forward flight configurations with first-order spatial accuracy for the first 360 time steps to establish the flow field, and second-order spatial accuracy thereafter. First-order temporal accuracy was specified for the “unsteady” part of the hover case (that is, the time steps in which Newton iterations are performed), but as the hover case converges to a steady state, second-order temporal accuracy was not used. In the forward flight cases, which are not steady-state, first-order and second-order temporal accuracy are compared. The same grid was used for each. In the hover case, U_{tip}= Mach 0.877 = 298.43 m/s. Figures 17 through 21 show comparison of pressure coefficient distribution along the non-dimensionalized blade chord vs. results from simulation using the TURBO flow solver at Mississippi State University (Webster 1994) and experimental results (Caradonna and Tung 1981) at several radial blade stations. The pressure coefficient from TURBO uses local velocity of the blade at the radial blade station *r/R* as reference velocity $V_{\infty(TURBO)}$, whereas in Tenasi the pressure coefficient uses a specified reference velocity V_∞, in the present work this is rotor tip speed. For comparison of values, the TURBO coefficient data was therefore scaled by $V_{\infty(TURBO)}{}^2/V_\infty{}^2$. Note also that the pressure coefficient is negated on the *y*-axis in order that values should appear relative to the surface on which they are observed (i.e. suction or negative pressures, on the top of the blade, are above positive pressure values that occur on the underside). There was no significant difference between running the case in a single rotating reference frame (“Nonsliding” results) vs. holding the farfield stationary and rotating the blade assembly (“Sliding” results), using the same grid. Results are plotted together to illustrate this. Both Tenasi cases were run using relative frame rotation. Values plotted are averages over one rotor revolution, i.e. all variable values for each point have been averaged over 360 timesteps.

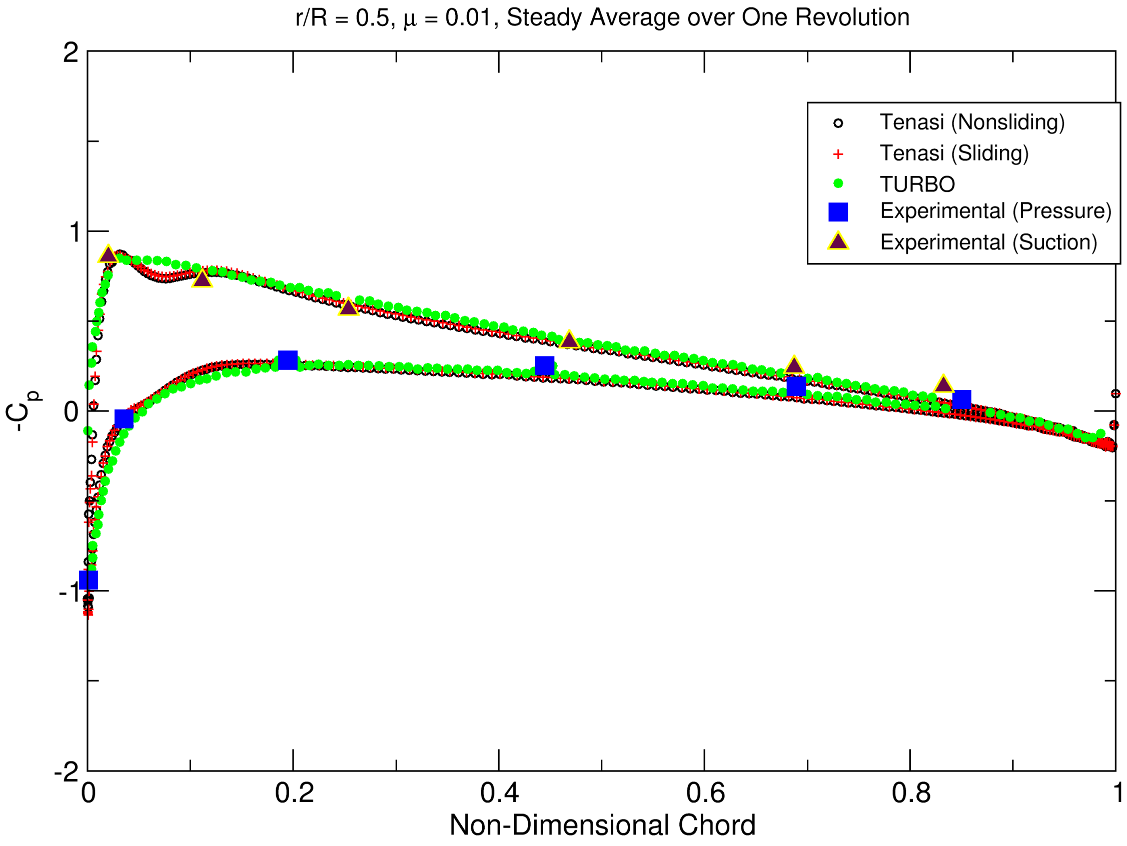


Figure 17 Pressure on blade surfaces at r/R=0.5, hover case

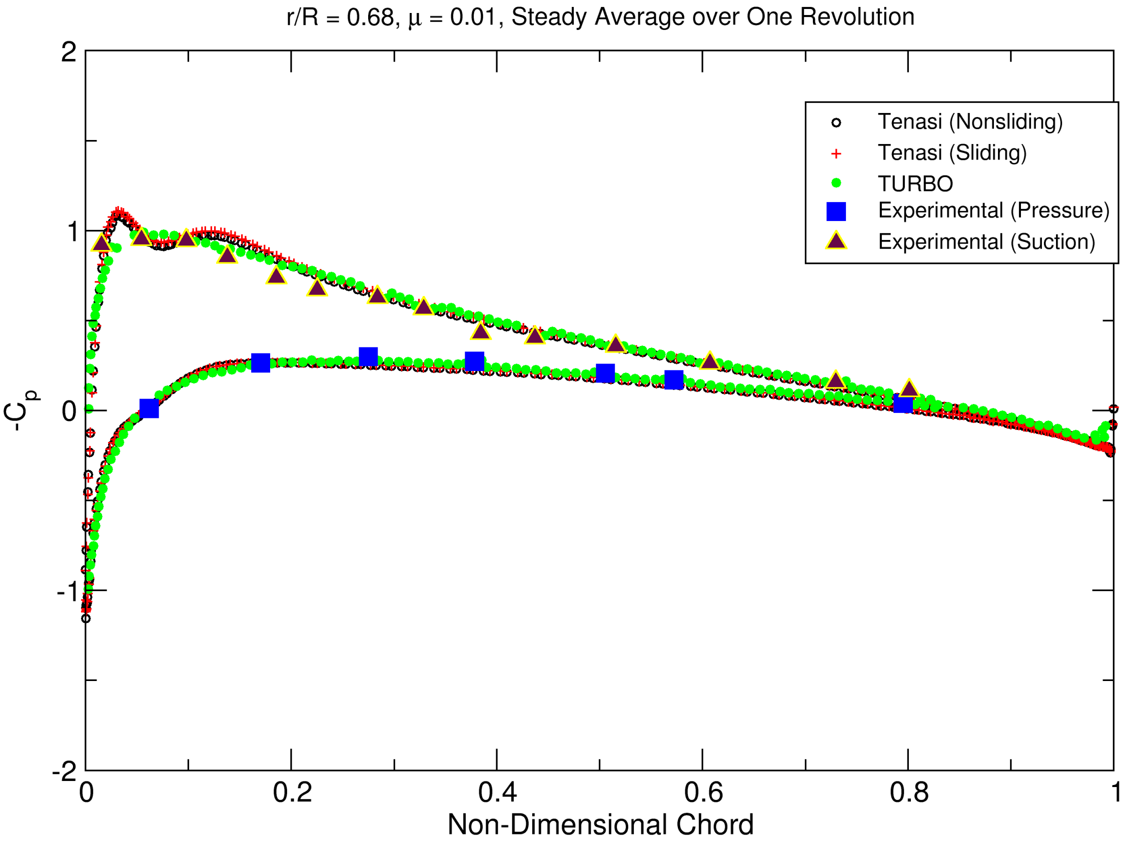


Figure 18 Pressure on blade surfaces at r/R=0.68, hover case

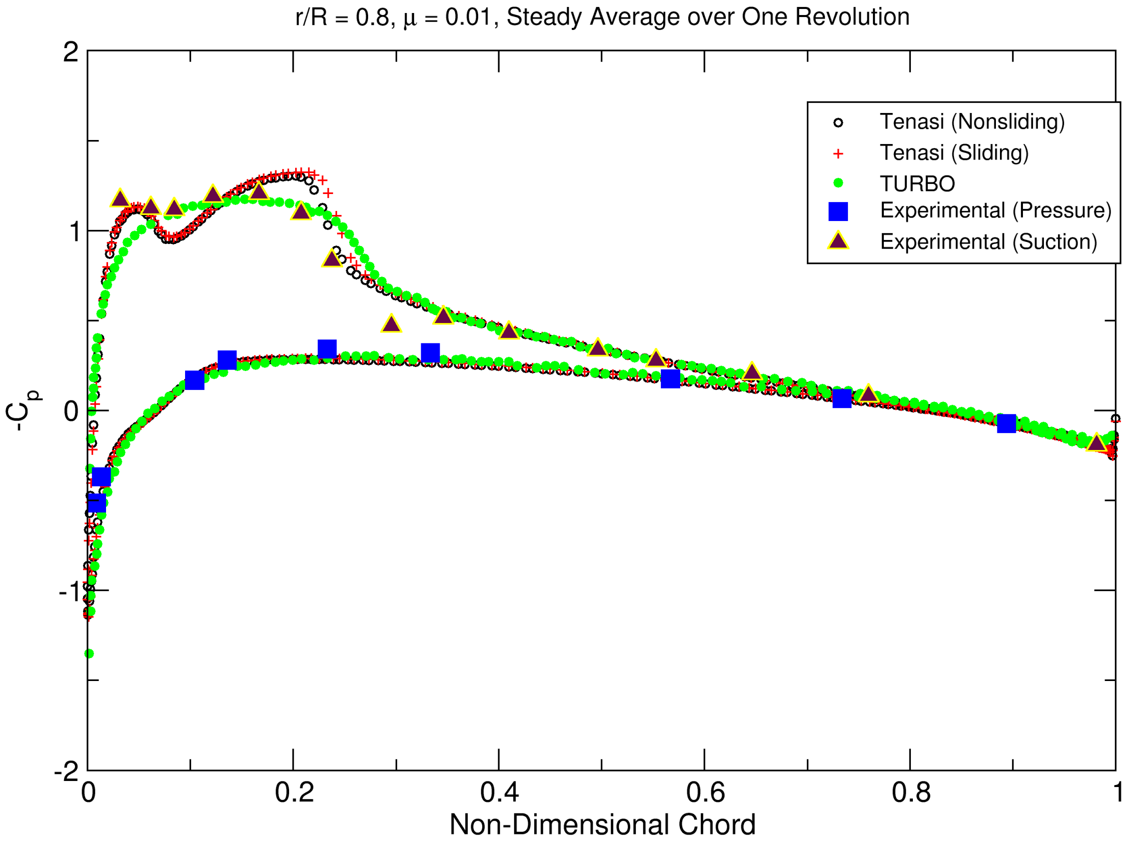


Figure 19 Pressure on blade surfaces at r/R=0.8, hover case

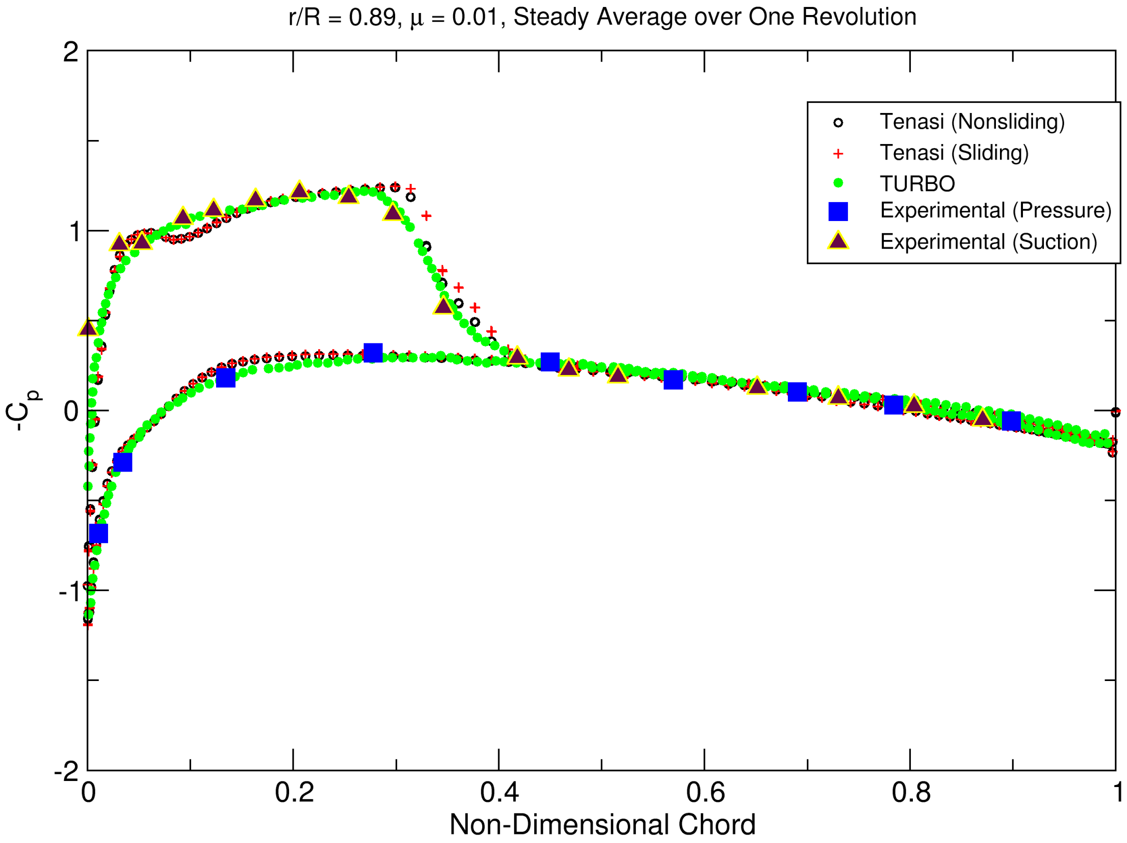


Figure 20 Pressure on blade surfaces at r/R=0.89, hover case

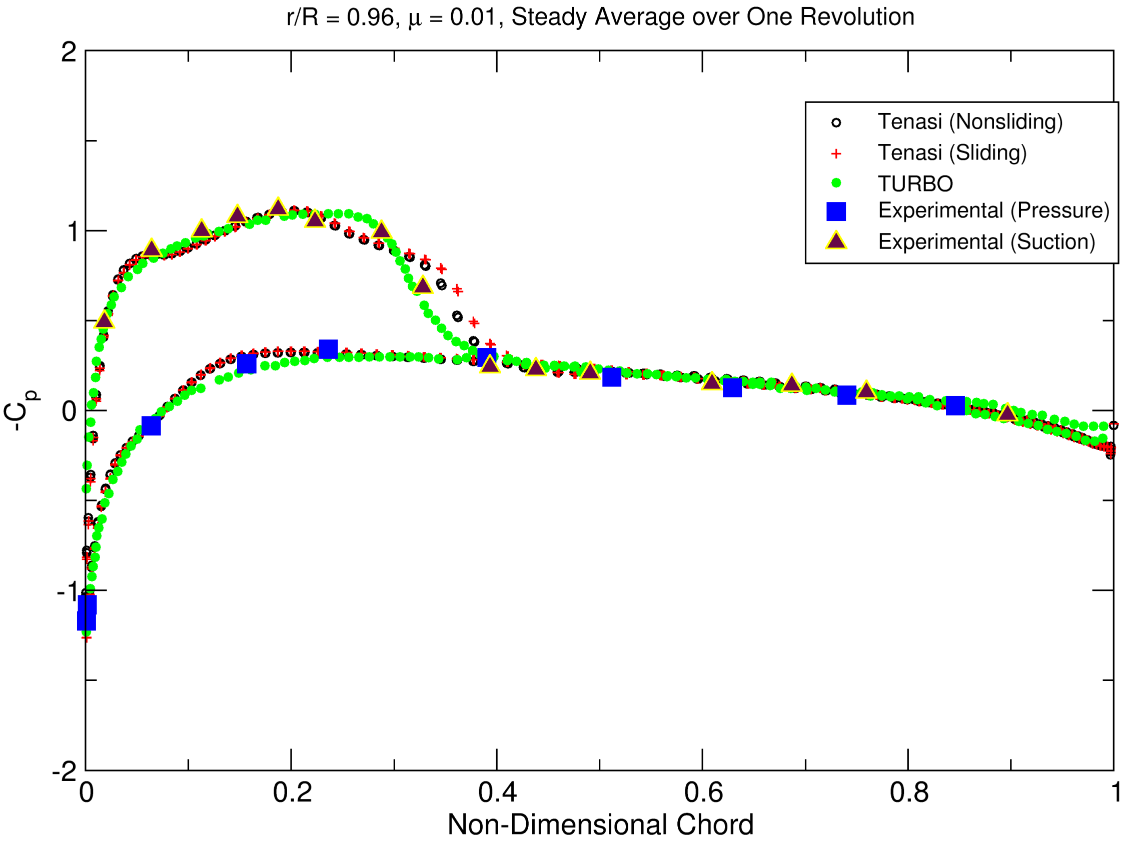


Figure 21 Pressure on blade surfaces at r/R=0.96, hover case

Agreement is generally good, though in the Tenasi results there is an initial “peak” and dip near the leading edge of the suction surface of the blade that can be observed at *r/R* = 0.68 and 0.8. These secondary shocks develop when flow around the leading edge accelerates and becomes locally supersonic, decelerates as pressure is encountered on the blade surface, and then accelerates again to supersonic speeds before encountering the primary recompression shock. The appearance of this phenomenon in these results can be attributed to increased grid resolution vs. the compared study, which was run using structured discretization methods with coarser meshes. The minimum spacing normal to blade (i.e. in the viscous boundary layer) in TURBO was 4e-06, while, as mentioned above, the Tenasi mesh used a viscous spacing of 1e-05. For the TURBO model, the chordwise point spacing on the blade surface was 1e-04 at the leading edge and 1.5e-03 at the trailing edge, whereas for the Tenasi models, the leading and trailing edges had spacing of 2e-03. Spanwise (in the radial direction), the initial spacing for TURBO at the blade tip was 1.35e-03, while for Tenasi it is 5.6e-03. The tightest spacings near the corners of each blade are then slightly coarser in Tenasi – however, for the TURBO model, spacing in the spanwise and chordwise directions was very rapidly increased away from the connector ends such that the total number of points on the blade surface was 35x22x71 or 54,670. The spacings for Tenasi are consistent along the blade length, and the number of points on each blade surface was 121,596. A comparison of the mesh surfaces is seen in Figure 22. The increased number of points both on the surface and in the space surrounding the blades very likely added to accuracy in resolving velocity gradients and curl (vorticity) and other flow field phenomena around the blades.

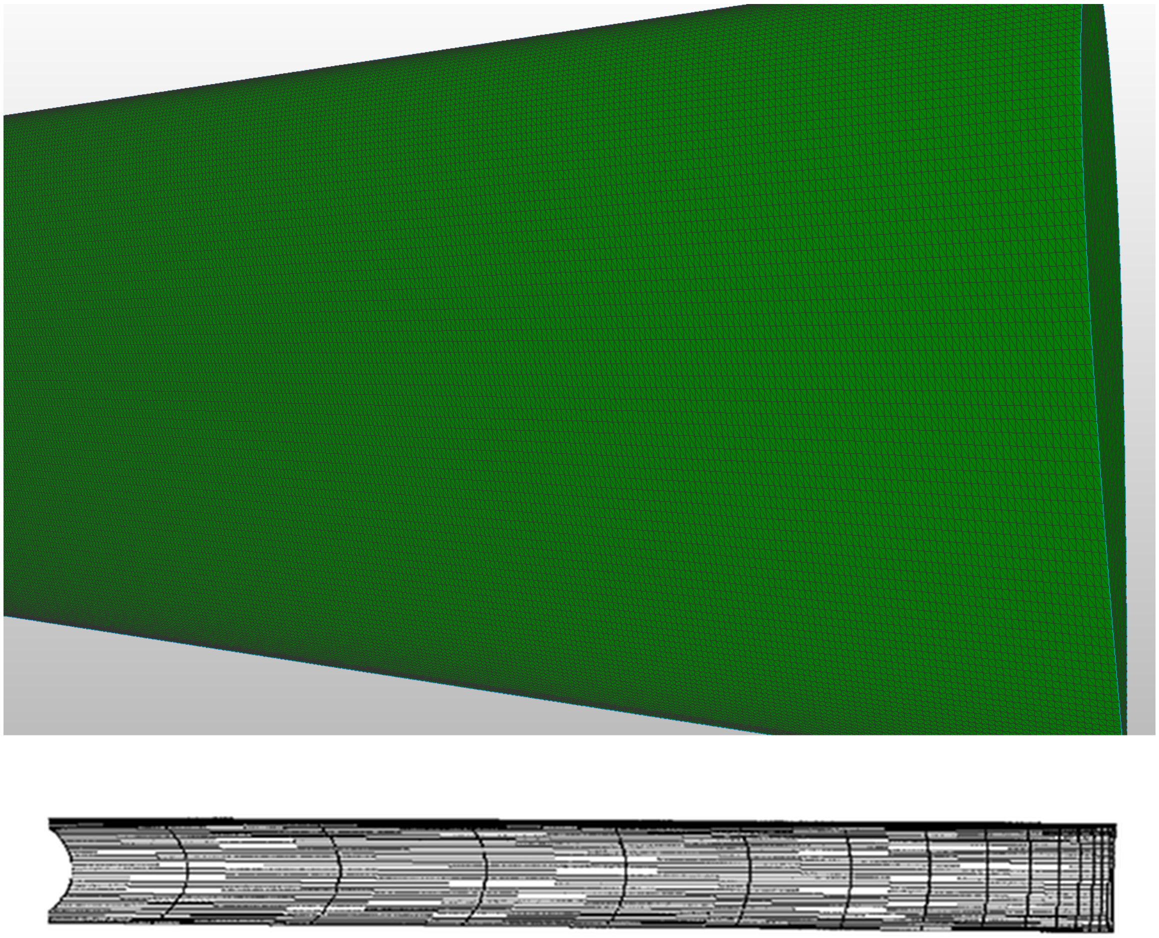


Figure 22 Tenasi isolated rotor blade surface mesh vs. TURBO blade surface mesh

In addition to the hover case, a forward flight case allowed for testing of cyclic pitching grid motion code. Forward flight cases were run with the same spatial and temporal accuracy (second and first-order, respectively), as well as second-order temporal accuracy. Coefficients for the forward flight condition were as follows: collective $\theta_0 = 8+1.37 = 9.37^{\circ}$, longitudinal $\theta_{1s} = -3.23^{\circ}$, lateral $\theta_{1c} = 1.11^{\circ}$. It should be noted here that cyclic pitching parameters in this work are the negative of those used in Webster (1994). While this caused quite a delay in creating discarded simulation runs and isolating the cause of the discrepancy in results, it is a simple difference in convention. The advance ratio μ for the forward flight condition is 0.15, thus the maximum advancing-tip Mach number is $M_{at} = 0.772$. Translational Mach is $M_t = 0.1007$. Mach 1 is 761.21 miles/hour or 340.29 m/s at Standard Sea Level conditions or 15.0ºC and 101325 Pa (Webster 1994), and rotational Mach $M_{bt} = 0.6713$. The rotational tip Mach gives a tip speed of 0.6713*340.29 m/s = 228.44 m/s. With the blade tip describing a circle with circumference = 7.1817 m, this is approximately 1908.5 rotations per minute (199.86 rad/s). With that in mind, using a timestep of 8.7329e-05 seconds/timestep will result in roughly one degree of rotation per timestep. For forward flight, the rotor shaft tilt $\alpha = -2^{\circ}$ is implemented as a free stream flow vector: (x,y,z) = (0.99939 -0.034899 0.0).

Experimental data were not available for the forward flight case, therefore Webster (1994) was used as the sole reference. Three cases were run in order to determine the effects of temporal accuracy and to validate relative frame rotation with respect to absolute frame or full grid motion with submotion of cyclic pitching blocks – absolute frame first order temporal, absolute frame second order temporal, and relative frame (first order temporal). Figures 23 through 30 show a comparison of values for the absolute frame and TURBO results. As above, pressure on both the “pressure” and “suction” surfaces of the blade is examined at a particular radial station (in this case, only at $r/R = 0.89$, where transonic effects can be observed). For these plots, however, values at different blade azimuth locations ψ show the development of transonic deceleration shock as the blade travels from the advancing side (port or $\psi = 90^{\circ}$) to the retreating side (starboard or $\psi = 270^{\circ}$). Note, in particular, the shock that develops at $\psi = 240^{\circ}$ and becomes more pronounced at $\psi = 270^{\circ}$ where velocity of the advancing blade at all stations is directly opposed to freestream and maximum relative tip velocity is achieved. The shock smooths out from $\psi = 300^{\circ}$ through $\psi = 360^{\circ}$.

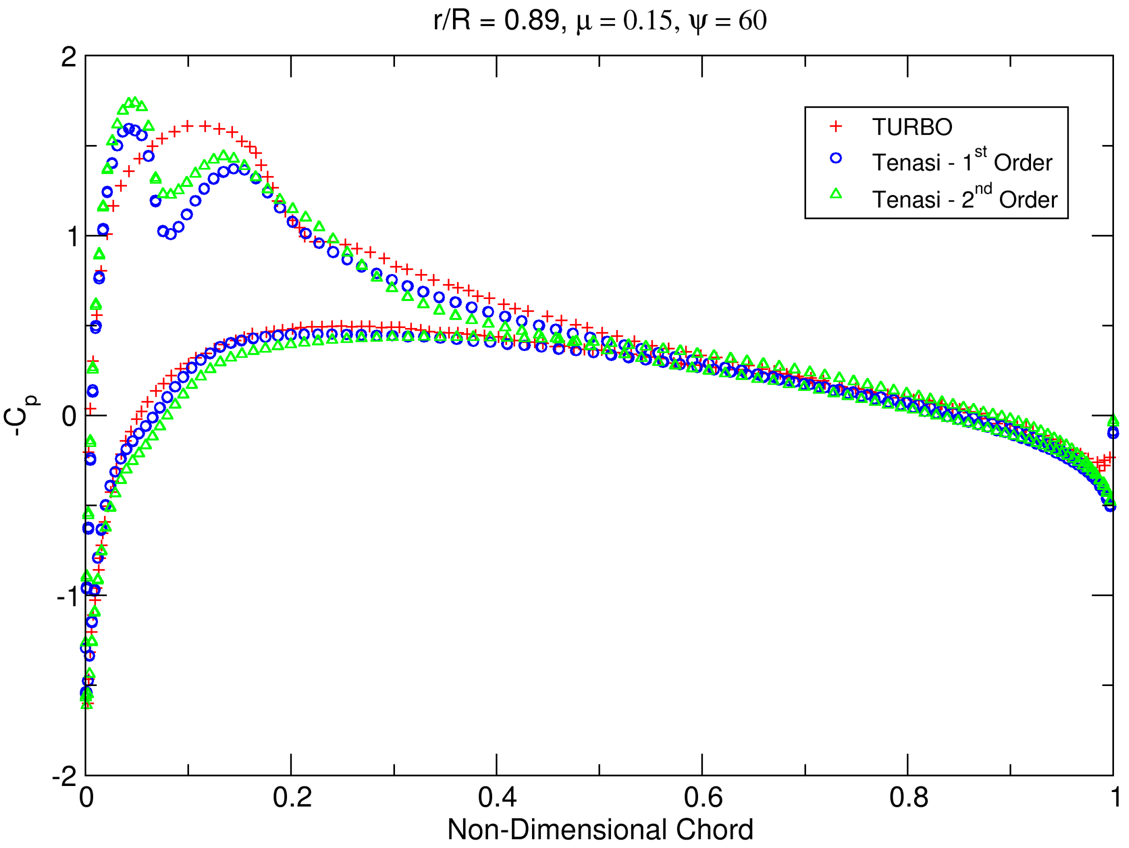


Figure 23 Pressure on blade surfaces at 60º, forward flight case

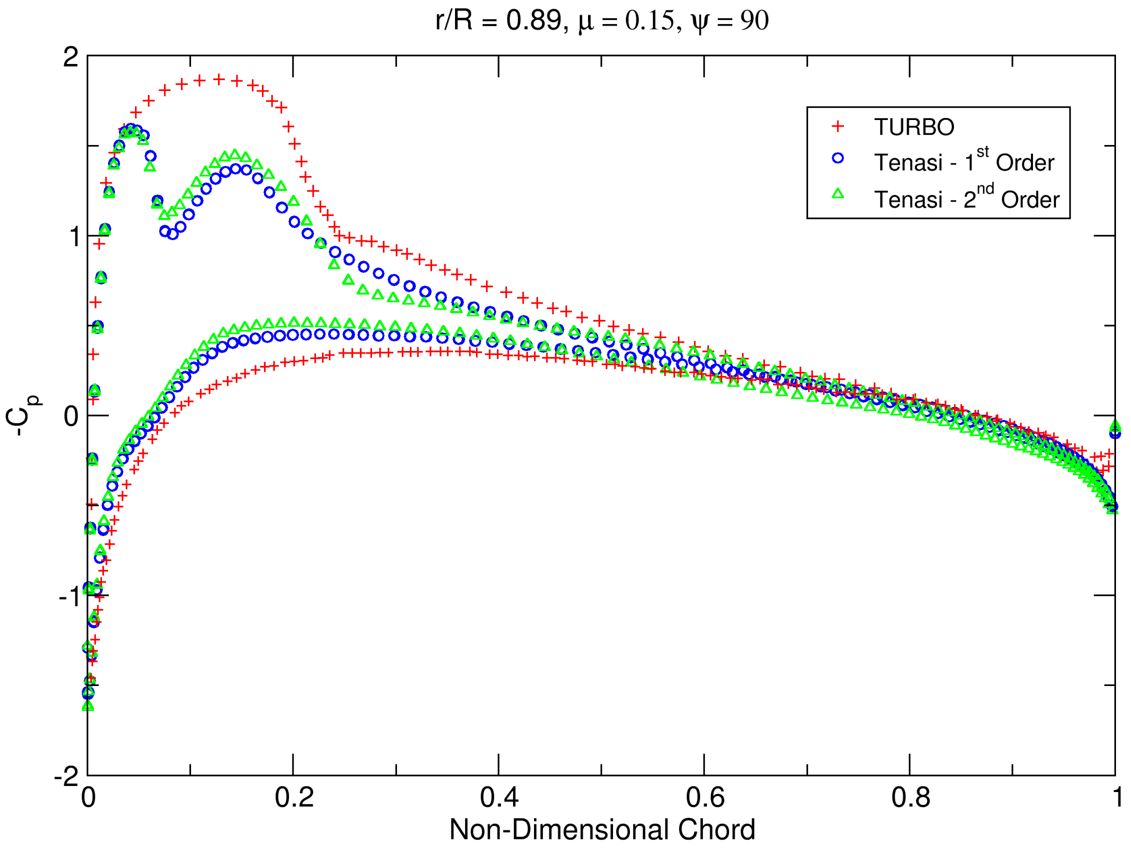


Figure 24 Pressure on blade surfaces at 90º, forward flight case

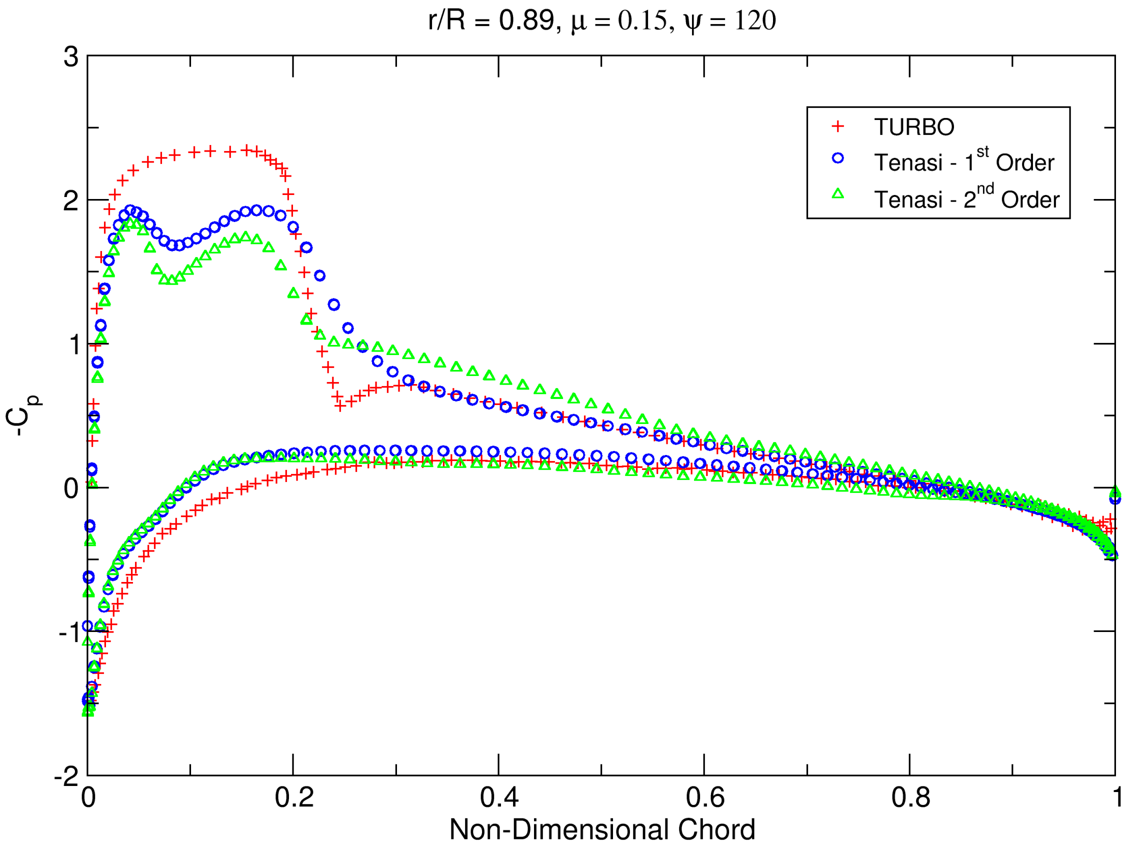


Figure 25 Pressure on blade surfaces at 120º, forward flight case

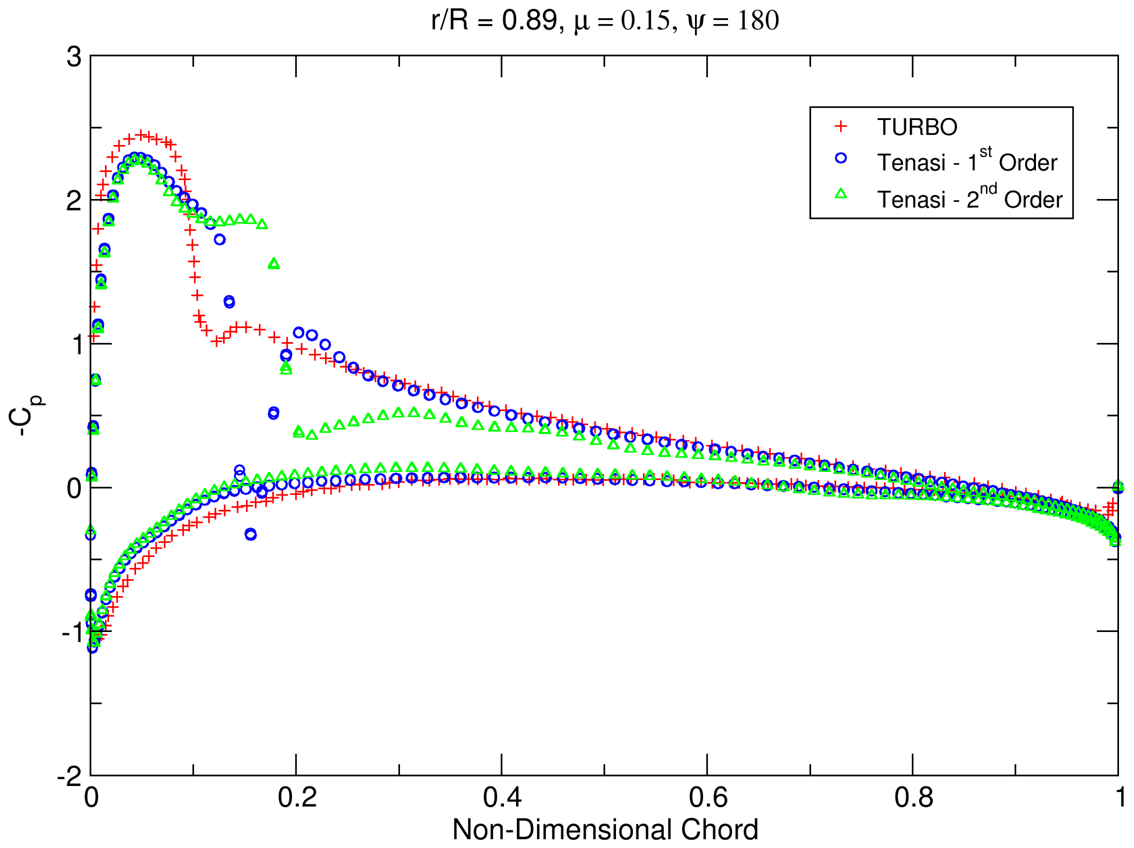


Figure 26 Pressure on blade surfaces at 180º, forward flight case

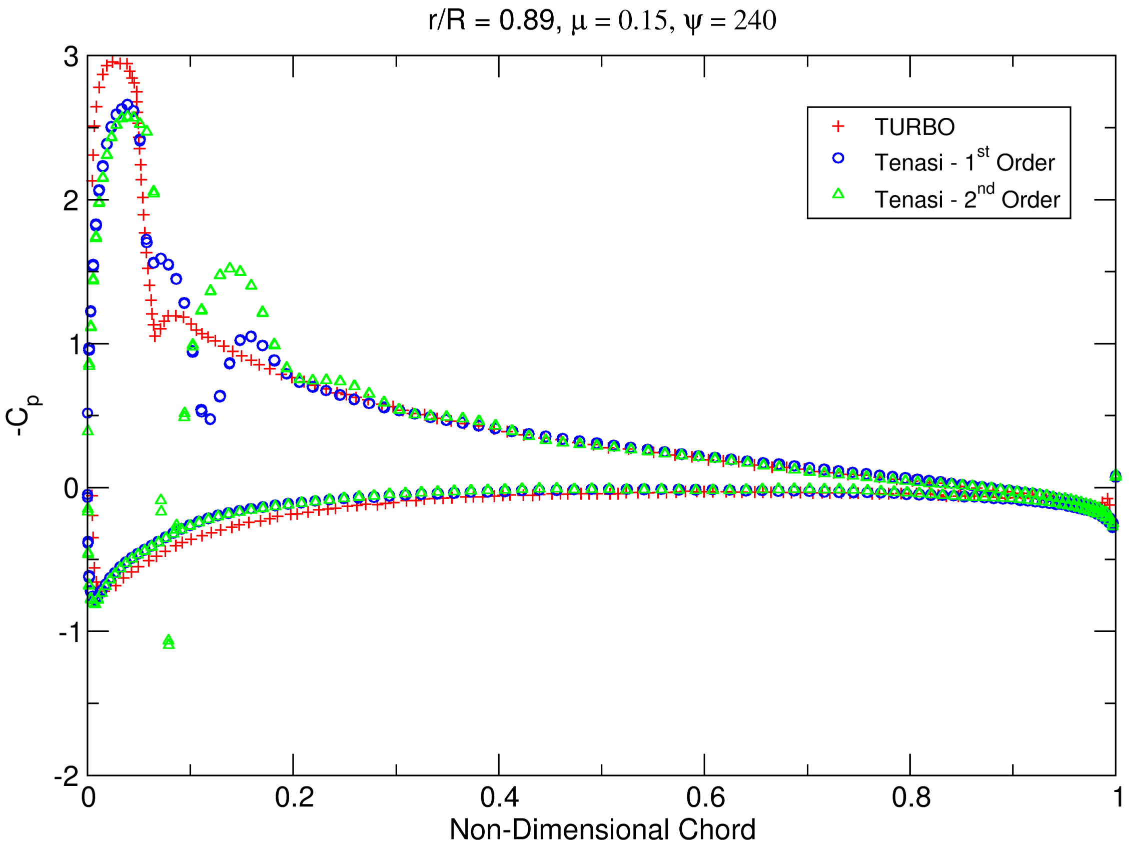


Figure 27 Pressure on blade surfaces at 240º, forward flight case

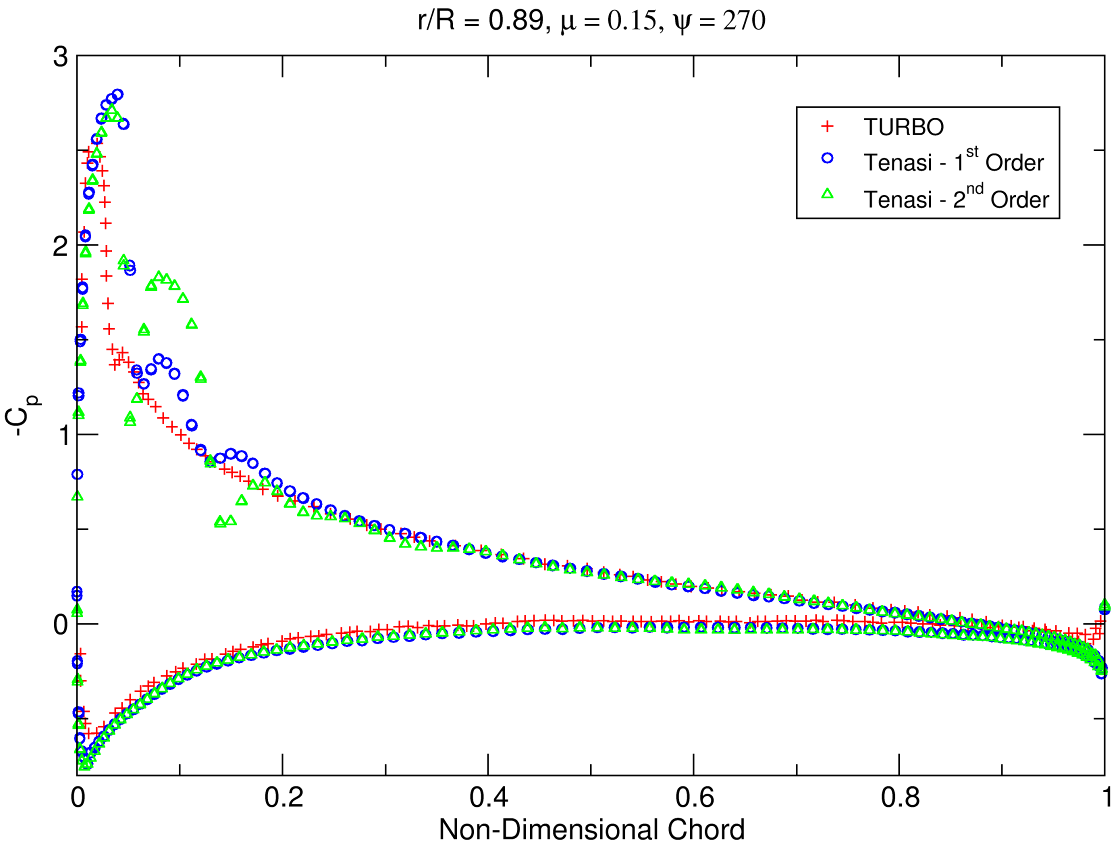


Figure 28 Pressure on blade surfaces at 270º, forward flight case

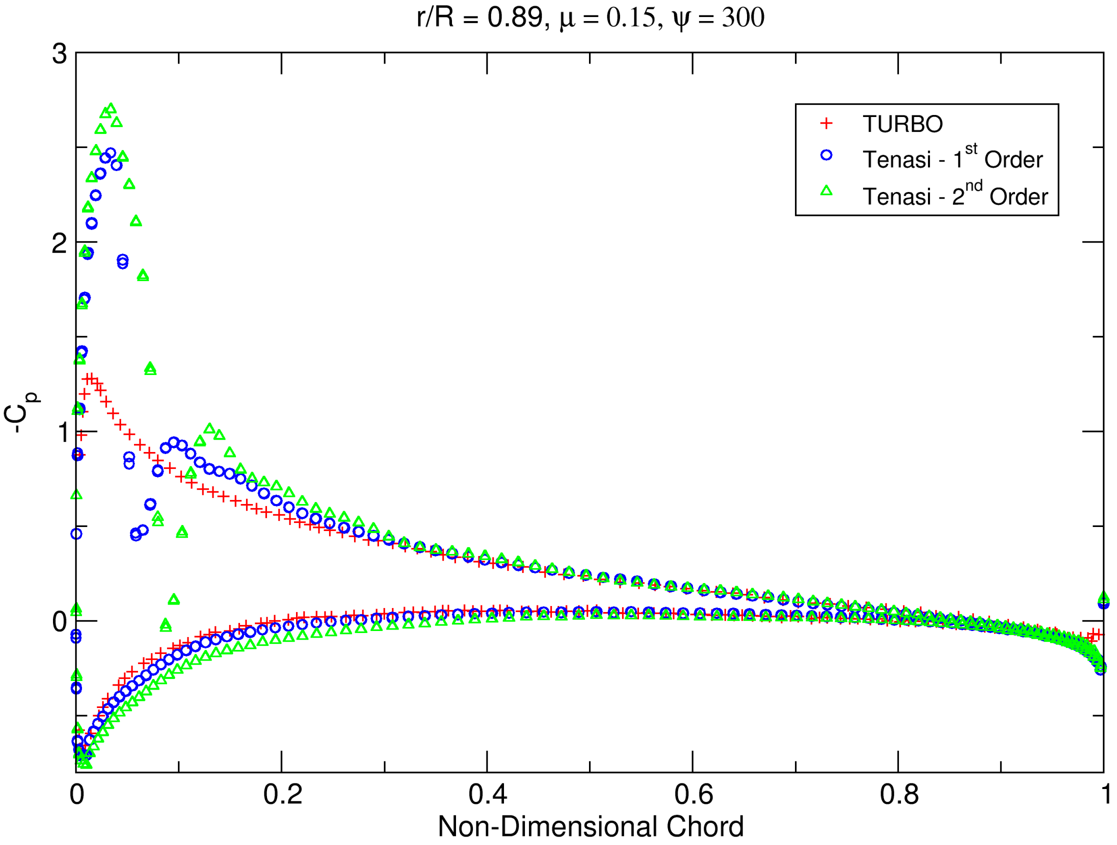


Figure 29 Pressure on blade surfaces at 300º, forward flight case

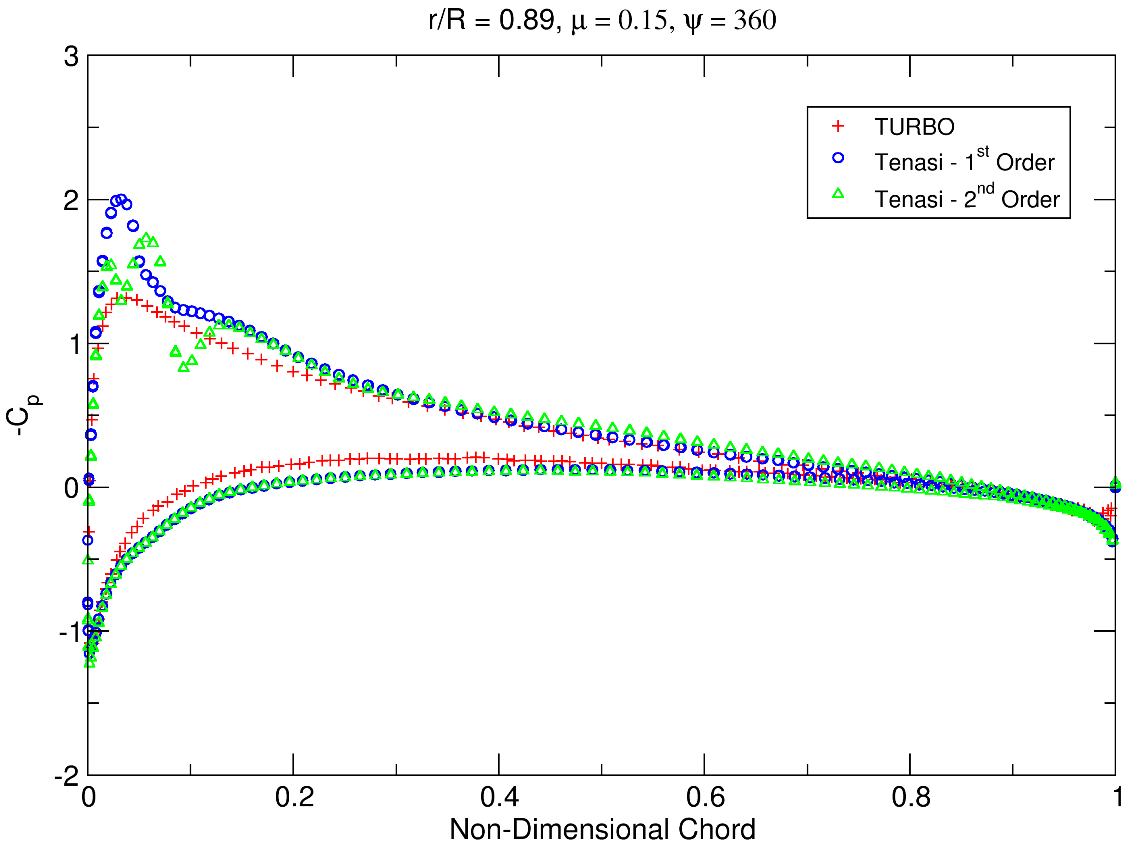


Figure 30 Pressure on blade surfaces at 360º, forward flight case

For additional comparison to TURBO, Figures 31 and 32 show transient measurement of pressure at several point locations on the rotor blade surface, specifically at *r/R* = 0.5 and *r/R* = 0.89, with output from "sensors" at each timestep defined in a python extension module to Tenasi (further use of this capability is made in the unsteady ROBIN case below). The sensors were positioned at 10%, 30%, and 50% of the chord distance along the blade top surface ("suction" side), and the corresponding coordinates are presented in Table 3. Agreement is again good, with an additional flow feature showing in the Tenasi results at *r/R* = 0.89 at the 10% chord location or leading edge of the blade, with a shock appearing and then disappearing between 0.25 and 0.5 revolutions (or 90º and 180º blade azimuth). Here, the transonic nature of the flow is more apparent with increased grid resolution and modeling of viscous effects near the blade surface as it advances through its highest speed relative to free stream flow. One can also see, at ψ = 60º and 240º, secondary shocks that are resolved by the higher grid resolution used with Tenasi, and that, particularly at 240º, second order time is picking up stronger signals thereof.

Table 3 Variable value "sensor" output location coordinates for isolated rotor forward flight

	10% chord			**30% chord**			**50% chord**		
	X	**Y**	**Z**	**X**	**Y**	**Z**	**X**	**Y**	**Z**
r/R = 0.5	-1.8750	0.0420	0.0888	-1.8750	0.0327	-0.0362	-1.8750	0.0108	-0.1593
r/R = 0.89	-3.3375	0.0420	0.0888	-3.3375	0.0327	-0.0362	-3.3375	0.0108	-0.1593

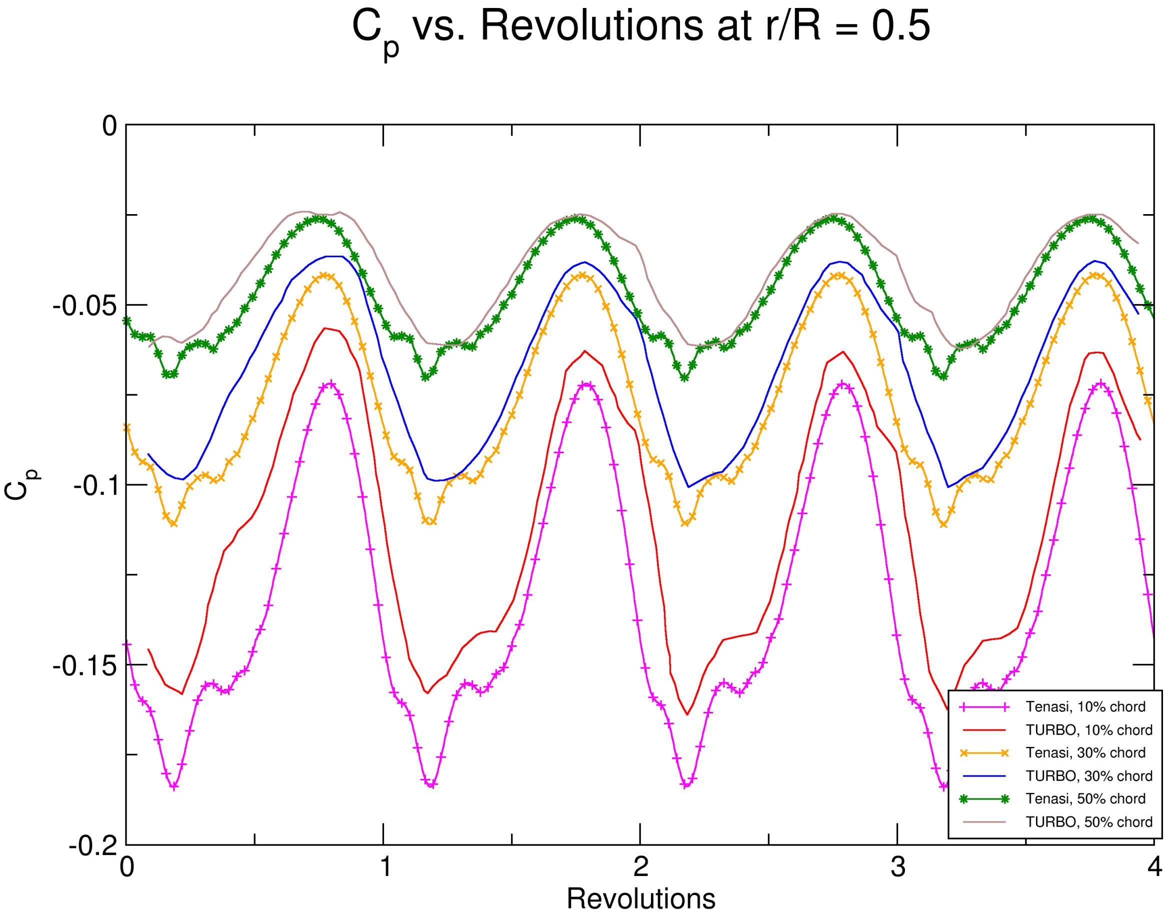


Figure 31 Pressure coefficient on blade surface vs. rotor revolutions, forward flight, *r/R* = 0.5

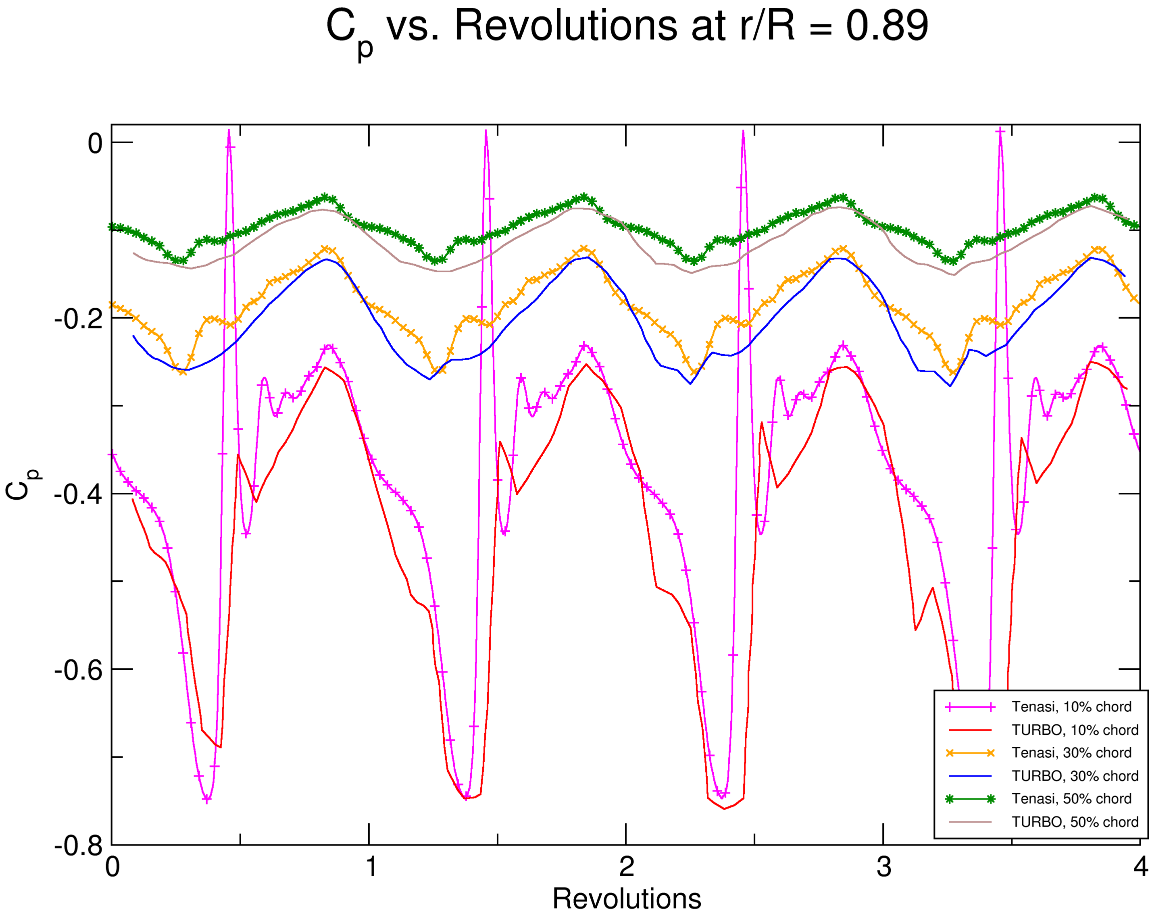
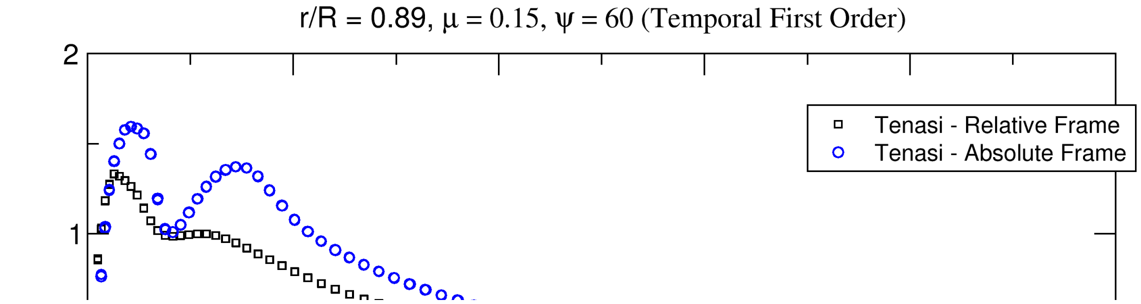


Figure 32 Pressure coefficient on blade surface vs. rotor revolutions, forward flight, *r/R* = 0.89

Finally, presented in Figures 33 through 35 are results from running the isolated rotor in the same mesh, using grid motion to achieve rotation (as above) vs. “relative frame” rotation, wherein the equations are recast in order to represent relative motion, i.e. grid velocity terms, in a non-moving mesh (that is, having the observer spinning with the rotor, as it were). This approach avoids the vector arithmetic necessary to move all elements of the mesh in question, and is a commonly-used technique when modeling turbomachinery to improve computational efficiency. However, rotorcraft in forward flight have an additional component of motion, namely cyclic pitching of the blades to counteract a rolling moment and impart forward velocity thrust redirection, as explained above. Here, we find that the grid motion (in absolute frame) of the blocks enclosing the blades is not properly accounted for as relative frame motion is implemented in Tenasi. This is apparent most at the advancing blade, where relative velocity is larger in magnitude, as seen at the azimuths plotted. Therefore, this feature of Tenasi currently will **not** produce valid results for rotorcraft forward-flight simulations, or any simulation where block submotion is modeled inside the relative frame block.



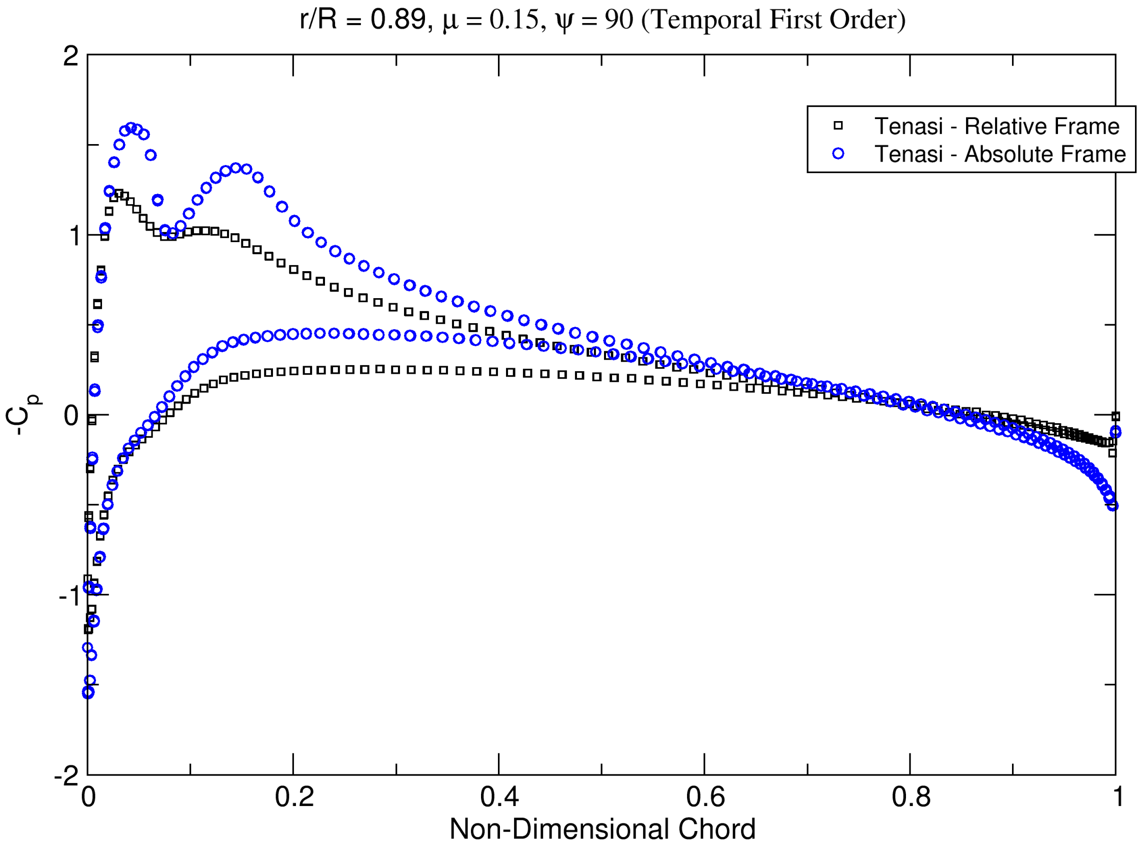


Figure 34 Relative vs. absolute frame rotation modeling, all other parameters being equal, $\psi = 90^{\circ}$

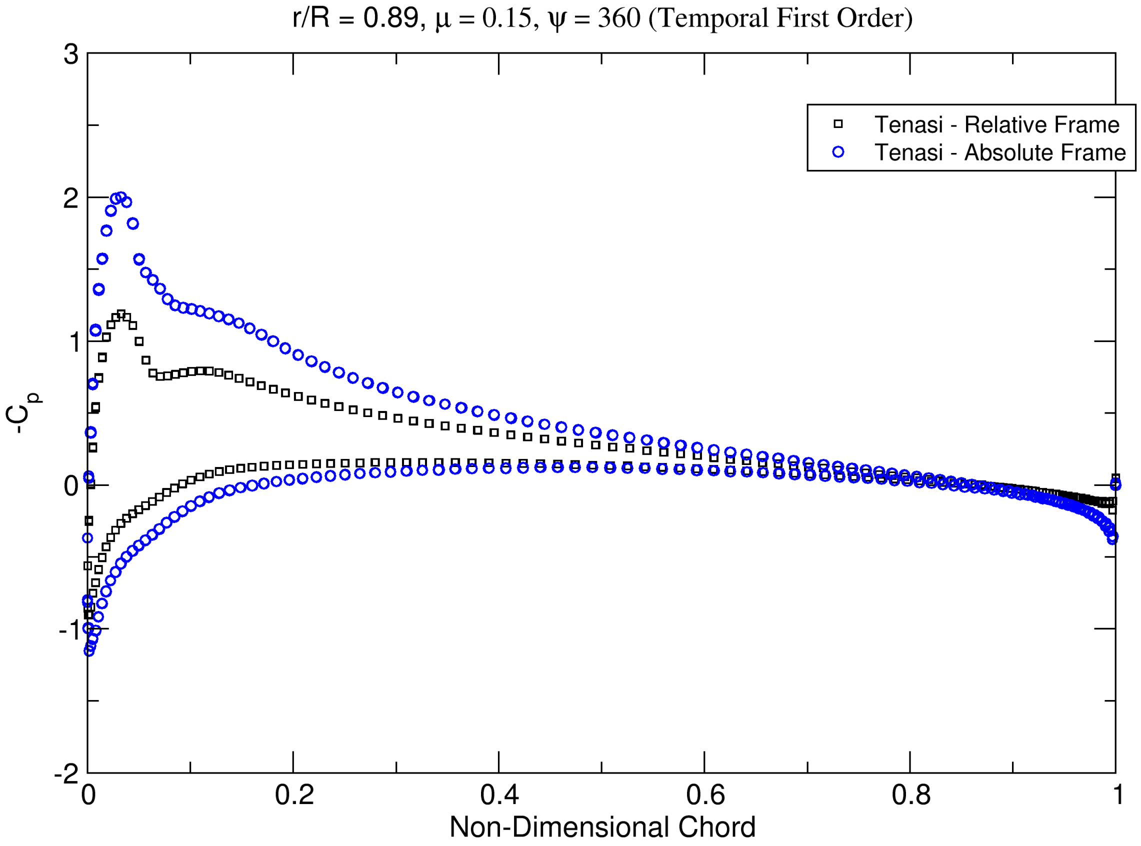


Figure 35 Relative vs. absolute frame rotation modeling, all other parameters being equal, $\psi = 360^{\circ}$

Steady-state, ROBIN Model

Two cases of the ROBIN fuselage alone, without a rotor system, were run to steady-state convergence at two different angles of attack: 0.0° and -5.0°. Results were compared to a simulation run by Schweitzer (1999) using the solver PUMA at The Pennsylvania State University and to wind tunnel test data gathered by Freeman and Mineck (1979). The wind tunnel test was run at NASA Langley's V/STOL closed return atmospheric tunnel at a Mach number of 0.062 and an effective

Reynolds number of 4.46 x 10^6 using a reference length of *L* (where the body length is 2*L*) and standard atmospheric conditions for air. Pressure data were gathered from taps located along the fuselage of the ROBIN geometry (a diagram of locations as presented in Schweitzer is here included as Figure 36). These surface-pressure data were time-averaged. Note that Schweitzer (1999) does not present results for $X/R = 0.2563$.

The PUMA simulation was of the ROBIN fuselage without rotor, in an unstructured parallel solver for the Euler equations. That study compared favorably to the experimental results except in areas where flow separation could be expected; for example, around the fore and aft sections of the pylon. In addition, the PUMA simulations were run at a high Mach number of 0.3 to avoid the need for pre-conditioning. The Tenasi case was run for the port half of the fuselage, using a symmetry boundary condition at the $y = 0$ plane. Simulation results are presented in Figures 37 - 40. We note excellent agreement with previous results. Figure 41 is included simply as an illustration of steady-state freestream surface flow conditions and the symmetrical boundary surface of the computational mesh.

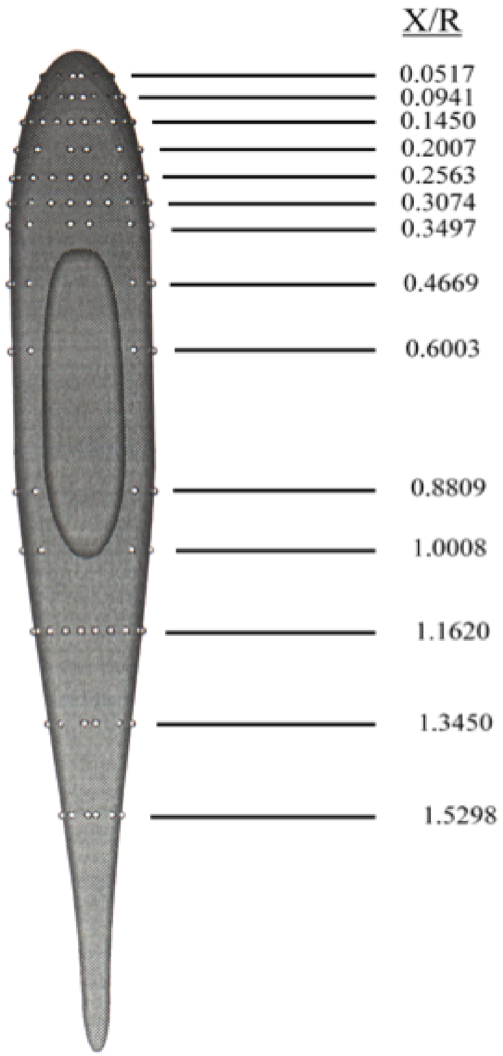


Figure 36 ROBIN Geometry Experimental Pressure Tap Locations

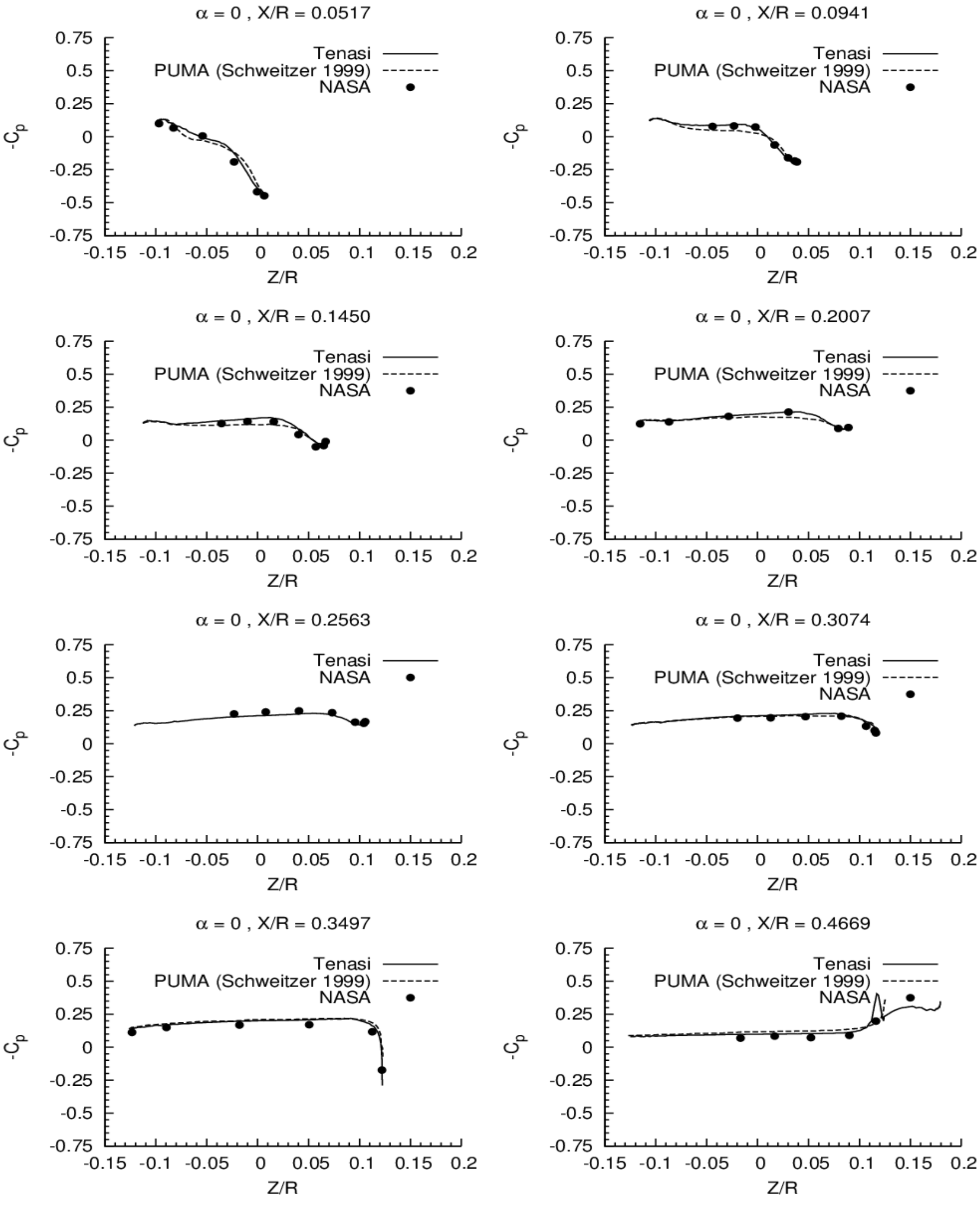


Figure 37 Steady-state surface pressure, 0° angle of attack, X/R < 0.4669

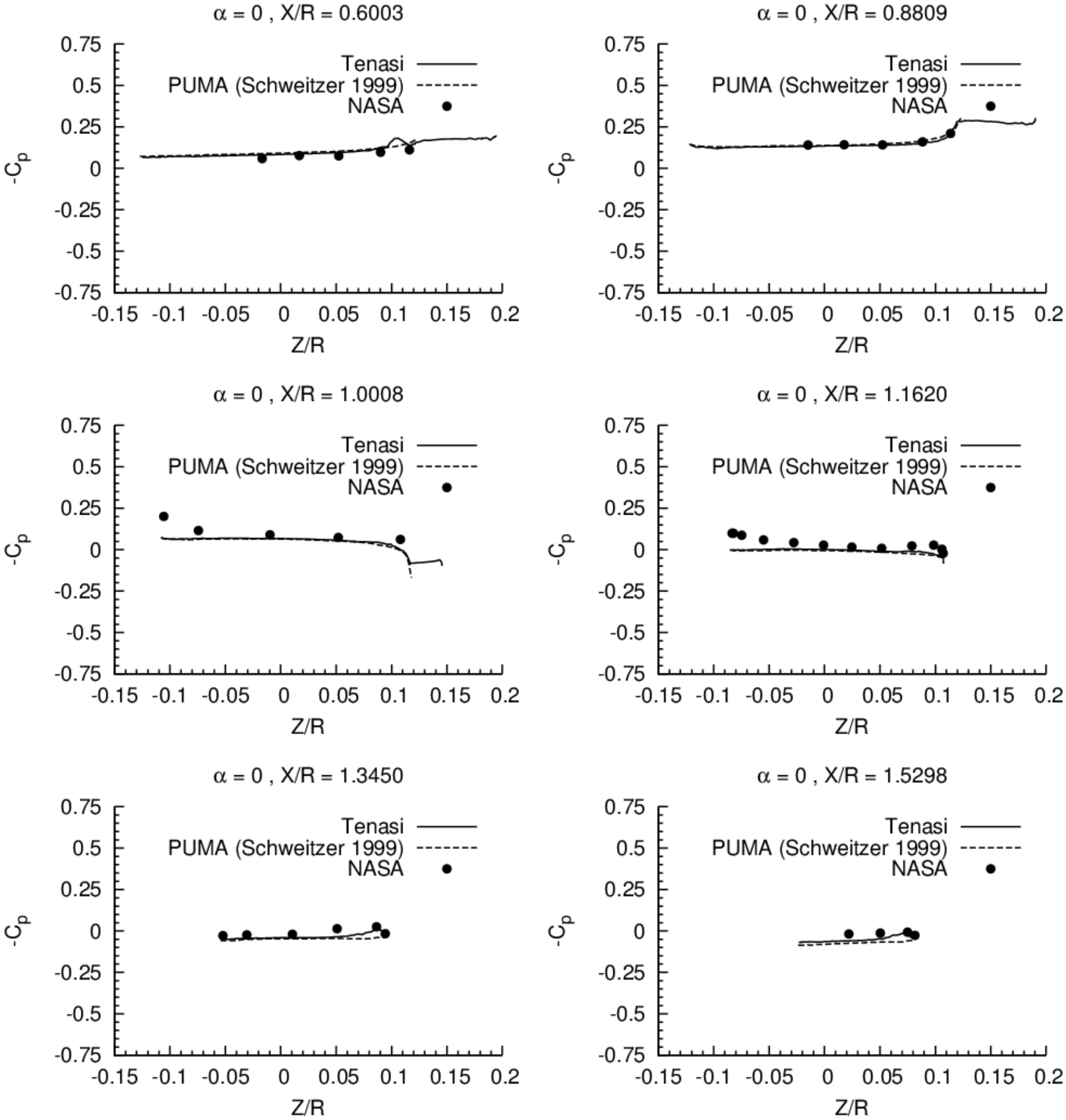


Figure 38 Steady-state surface pressure, 0° angle of attack, X/R > 0.6003

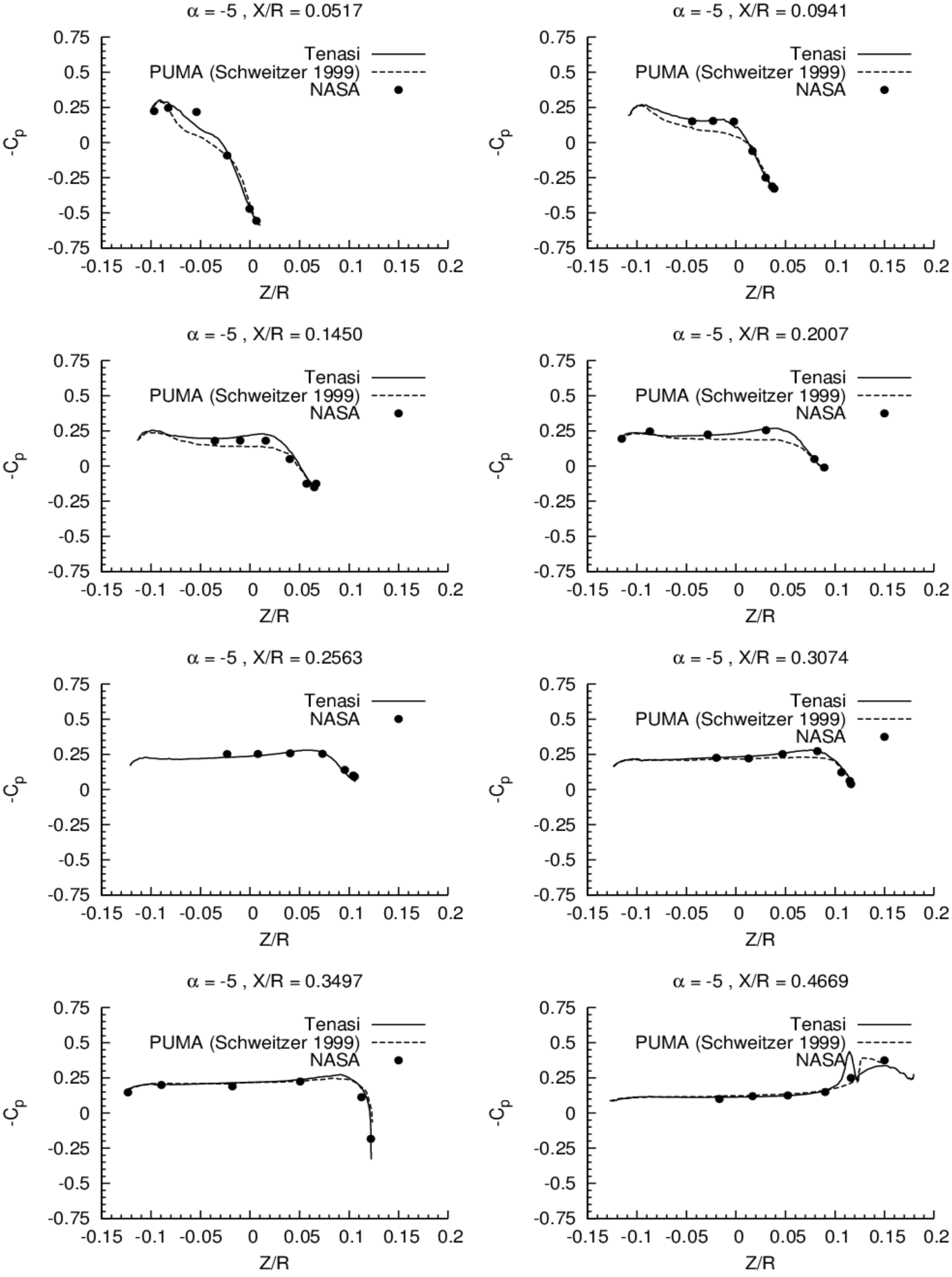


Figure 39 Steady-state surface pressure, -5° angle of attack, X/R < 0.4669

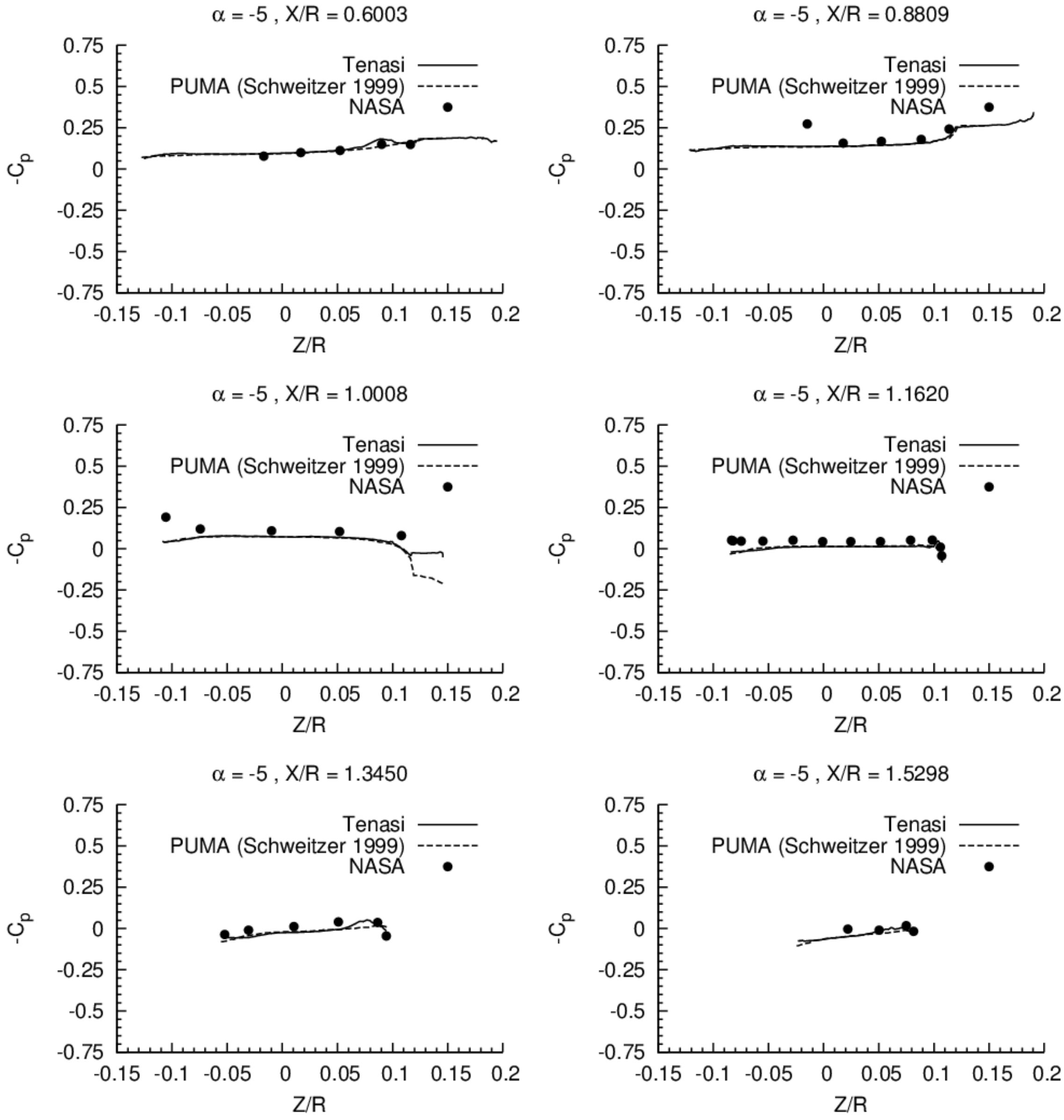


Figure 40 Steady-state surface pressure, -5° angle of attack, X/R > 0.6003

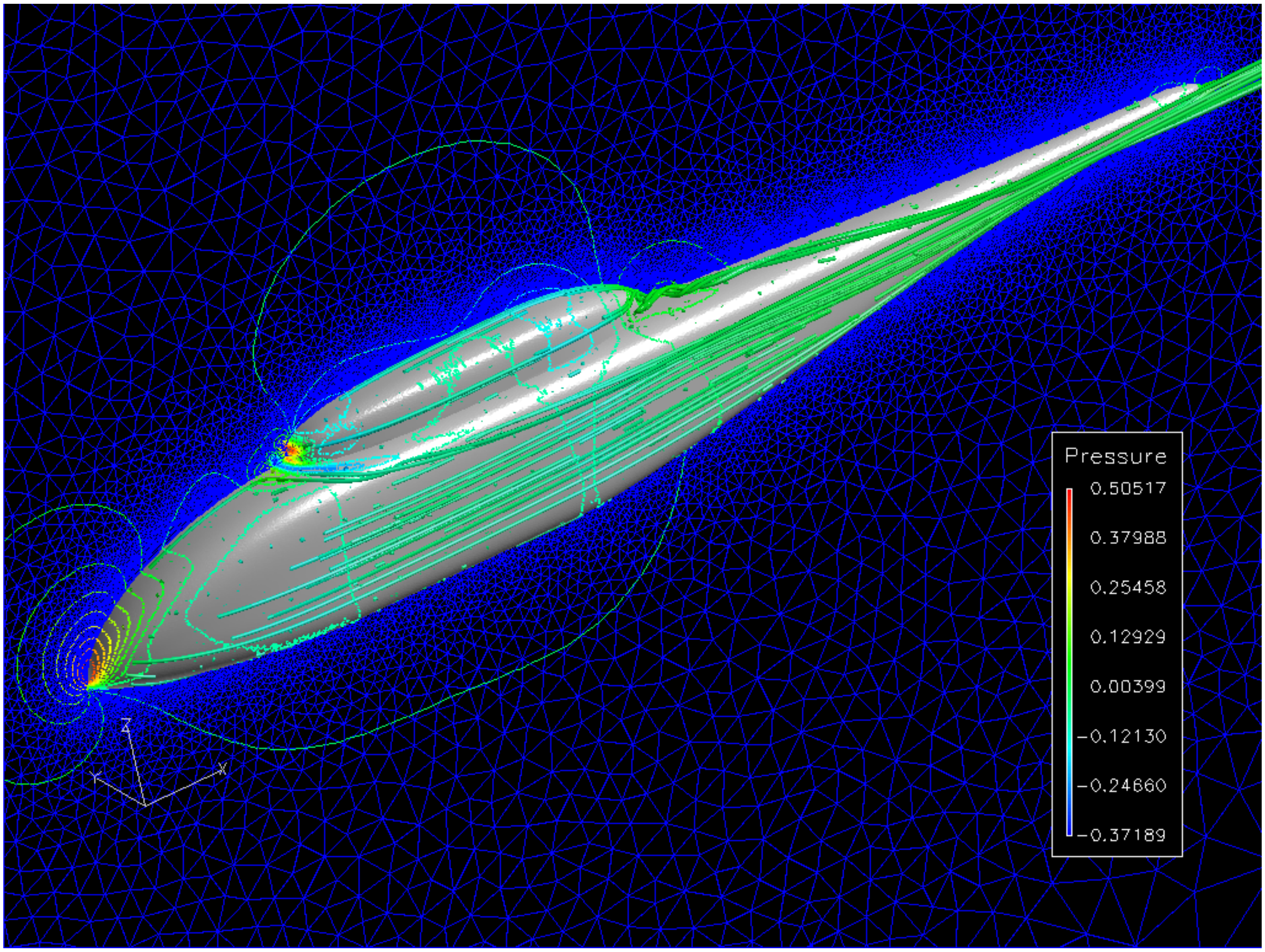


Figure 41 ROBIN steady case in Tenasi, representative flow streamlines and pressure contours

Unsteady, ROBIN Model

Unsteady cases of the ROBIN fuselage and rotor in hover and forward flight provided validation cases for sliding interfaces that allowed for relative rotation of the rotor and examination of flow effects on the body. Experimental data for this case were collected in a wind tunnel at NASA Langley and reported in NASA Technical Memorandum TM-2000-210286 (Mineck and Gorton 2000). Computational simulations have also been used for comparison: the unstructured-grid simulation of forward flight using the Euler equations (i.e. inviscid) from Korea Advanced Institute of Science and Technology (KAIST) (Nam, Park, and Kwon 2006), and a structured simulation of forward flight in both the Euler equations and the full Navier-Stokes from the University of Liverpool (Steijl and Barakos 2009), both using sliding interface techniques. We also compare results for the hover case with an Euler simulation of the hover condition using unstructured grids and overset methods from KAIST in 2010 (Lee et al. 2010).

For the ROBIN model, thrust values were compared as a coefficient C_T scaled by a rotor solidity coefficient σ, which is described in the list of symbols. The flight conditions for this study were selected to match available data, i.e. for forward flight, advance ratio $\mu = 0.15$ and in hover, $\mu = 0.01$. In both cases, the rotor thrust coefficient $C_T/\sigma = 0.0656$. Trim conditions in simulation that would produce this thrust coefficient were calculated by Nam et. al. (2006) using a Newton-Raphson method, in which the thrust and moment coefficients are optimized given the collective and cyclic pitch angles as dependent variables. This is necessary in simulation because, among other factors, rotor blade movement due to flapping is not modeled by mesh movement but by grid rotation to achieve blade pitching using pitch-flap equivalence (Webster 1994). Thrust vector direction is affected by cyclic pitching, fluctutations in flow velocity over the blades (relative to a reference frame moving with the free stream) due to rotation, and passage of the blades over the body surface, and is therefore unsteady.

However, using the trim conditions as given in Nam et. al. (2006) and Steijl and Barakos (2009), the rotor thrust in Tenasi, time-averaged over one full rotation, "converged" to $C_T/\sigma = 0.0653$ (see Figure 42).

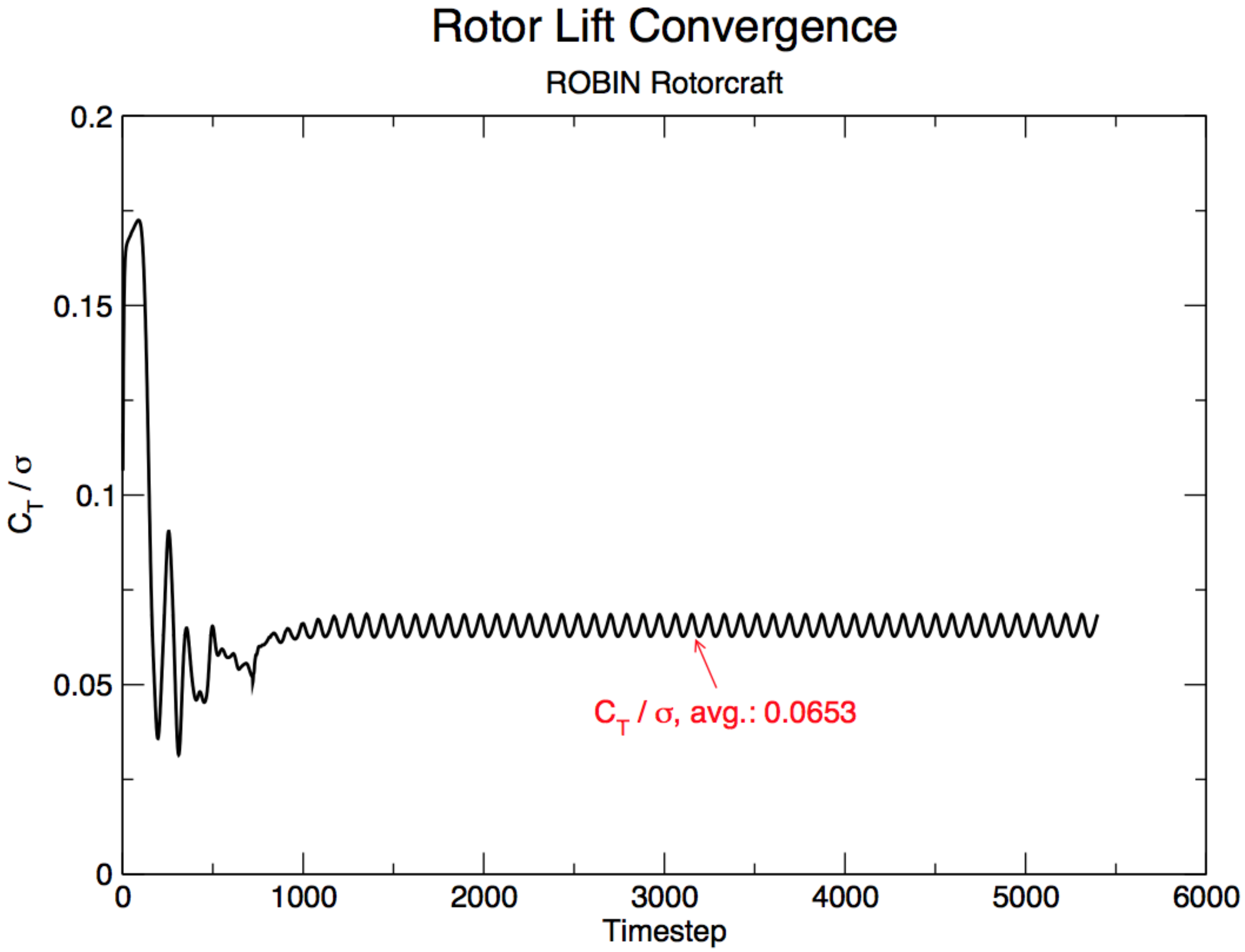


Figure 42 Unsteady convergence of coefficient of thrust over sigma vs. timestep for ROBIN rotorcraft in Tenasi

Unsteady pressure data was gathered in the experimental case via taps located along the body surface, as shown in Figures 43, and 44. Coordinates for these tap locations are included in Table 4, along with adjusted coordinates used for simulation sensors. It was found that the coordinates given for some sensor locations on the physically constructed fuselage were beneath the body surface as generated analytically. The adjusted coordinates include an offset necessary to place the sampling locations outside of the generated ROBIN body surface coordinates. To obtain dependent variable values at these locations for each simulation timestep, an extension module was provided to Tenasi giving these adjusted coordinates in Python code (see Appendix B).

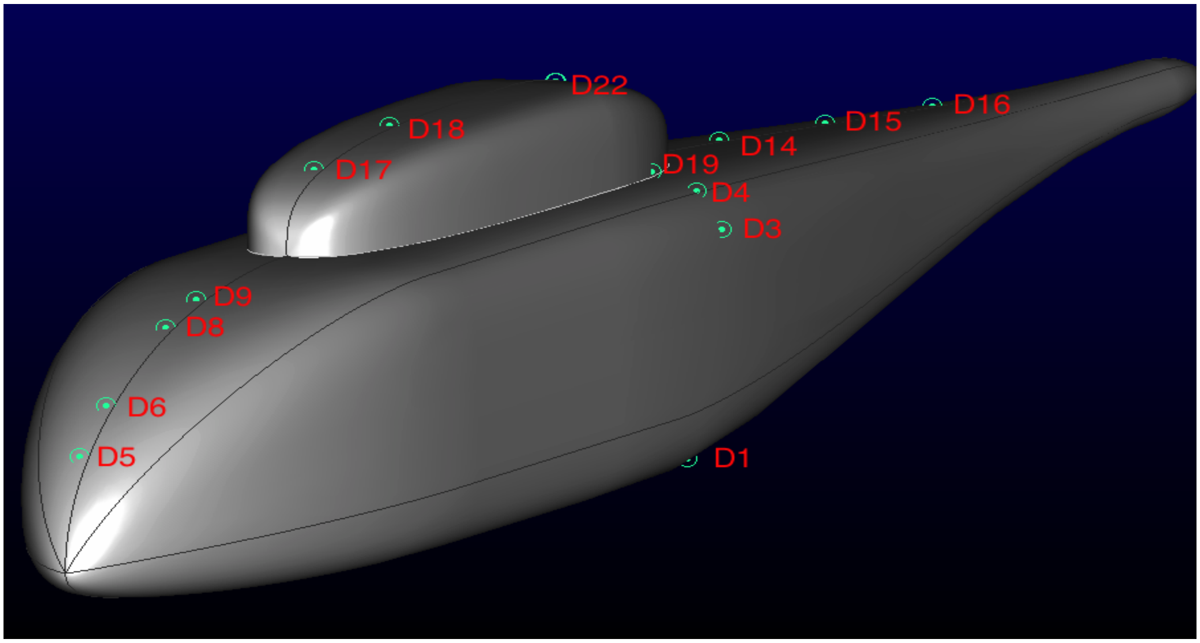


Figure 43 ROBIN sensor locations, port side

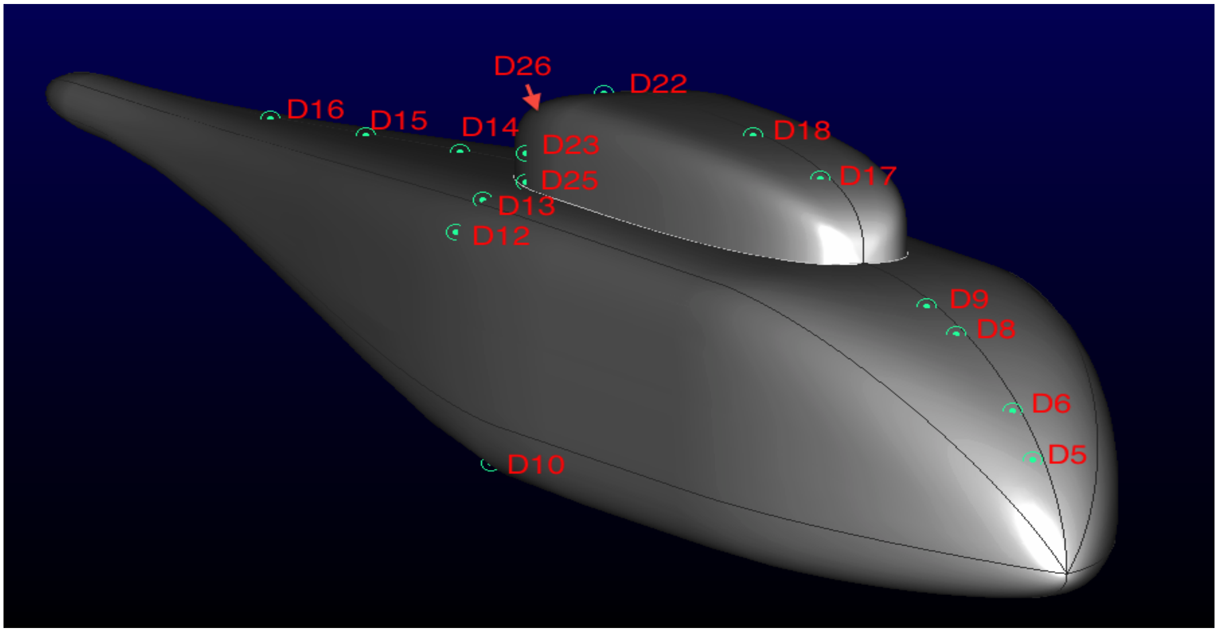


Figure 44 ROBIN sensor locations, starboard side

Table 4 Orifice/Sensor Locations for Unsteady Pressure Measurements

	Coordinates on Experiment Body, Crown			**Coordinates on Simulation Body, Crown**		
Sensor	**x/l**	**y/l**	**z/l**	**x/l**	**y/l**	**z/l**
D5	0.052	0.007	0.004	0.052	0.007	0.008653
D6	0.096	0.006	0.037	0.096	0.006	0.041440
D8	0.201	0.007	0.090	0.201	0.007	0.090193
D9	0.256	0.007	0.110	0.256	0.007	0.106196
D17	0.467	0.007	0.185	0.467	0.007	0.179424
D18	0.600	0.007	0.202	0.600	0.007	0.194343
D22	0.896	0.007	0.200	0.895	0.007	0.187737
D26	1.001	0.007	0.150	1.001	0.007	0.145459
D14	1.180	0.007	0.100	1.180	0.007	0.106102
D15	1.368	0.007	0.087	1.368	0.007	0.092695
D16	1.556	0.007	0.073	1.556	0.007	0.079658
	Coordinates on Experiment Body, Ring			**Coordinates on Simulation Body, Ring**		
D1	0.897	-0.091	-0.117	0.896431	-0.089118	-0.112276
D3	0.895	-0.117	0.080	0.895024	-0.117342	0.080075
D4	0.895	-0.096	0.106	0.895387	-0.099154	0.110412
D19	0.895	-0.067	0.125	0.896407	-0.074639	0.125000
D22	0.895	0.007	0.200	0.892686	0.006999	0.188190
D23	0.895	0.067	0.150	0.896393	0.074525	0.150280
D25	0.895	0.067	0.125	0.896407	0.074439	0.125000
D13	0.895	0.094	0.109	0.895262	0.096032	0.112436
D12	0.897	0.116	0.086	0.896974	0.115673	0.085900
D10	0.897	0.094	-0.115	0.896496	0.092120	-0.110967

Hover conditions were simulated and compared with results from Lee et al. (2010). The rotor trim conditions that would produce the correct thrust vector (magnitude C_T/σ = 0.0063) at advance ratio μ = 0.012 were: collective θ_0 = 8.8º, longitudinal θ_{1s} = 0.2º, lateral θ_{1c} = -0.1º. This model was run for 28 rotor revolutions, and the last four revolutions were time-averaged for each azimuth degree. The results are compared in figures 45 through 52. Note that for these plots an azimuth offset or lag of 30º that appears at some sensor locations in the NASA data has not been corrected for. This phase offset between experimental and simulated results is acknowledged in the current literature (Kenyon and Brown 2009), can be observed in results from previous studies (Park, Nam, and Kwon 2003), and is most likely due to signal delay in the physical instrumentation.

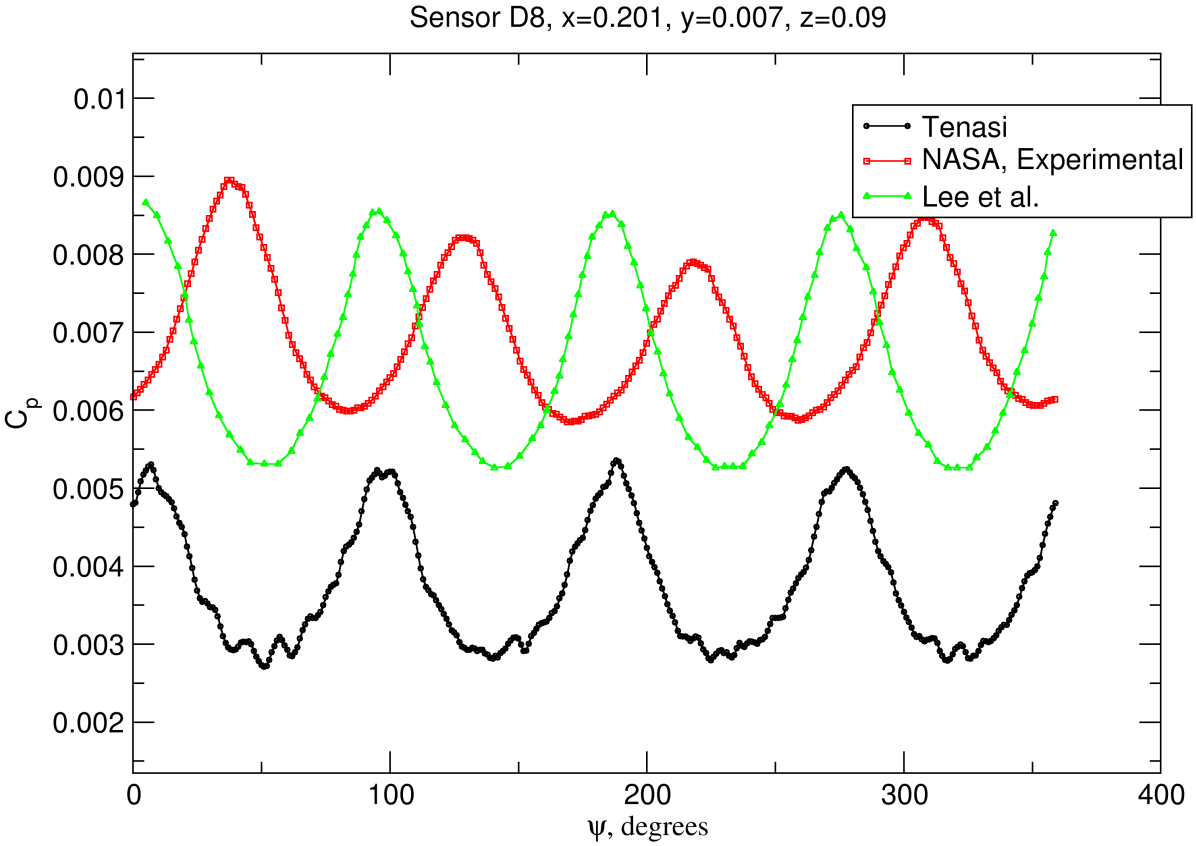


Figure 45 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D8

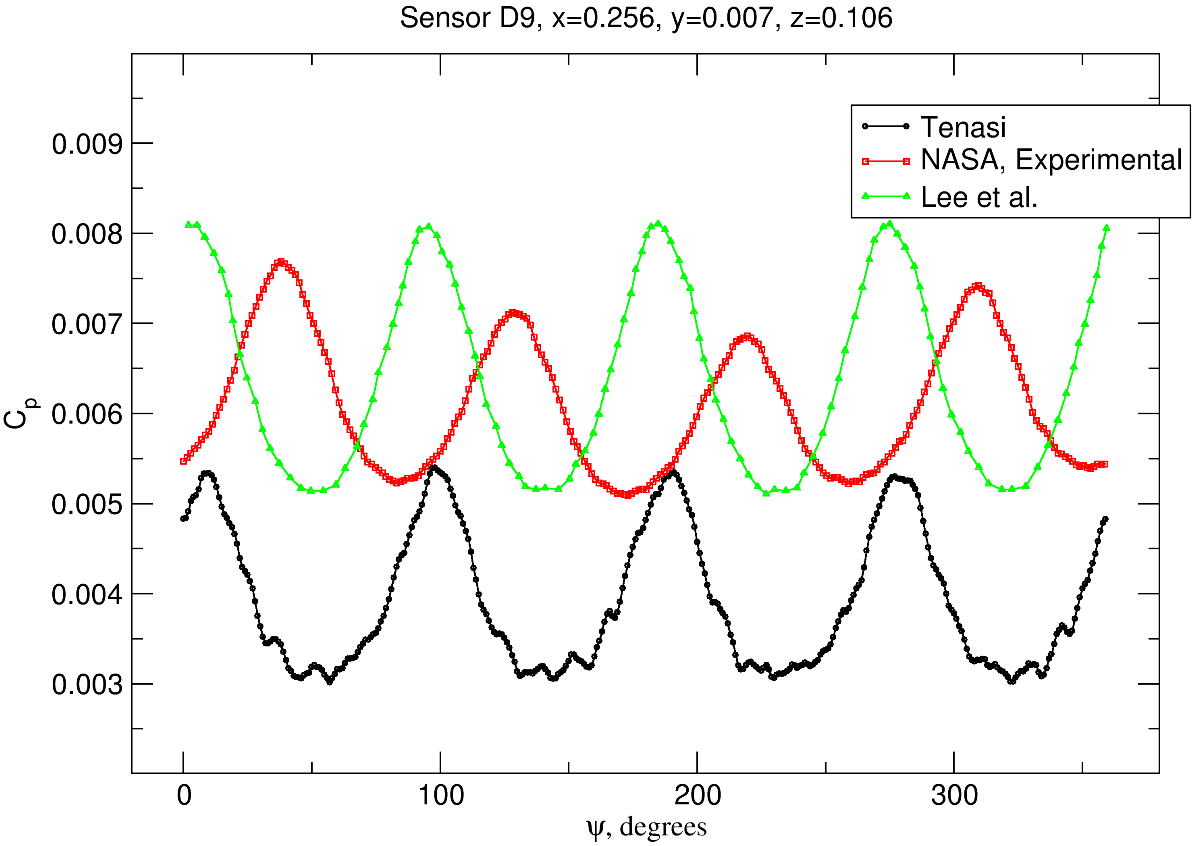


Figure 46 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D9

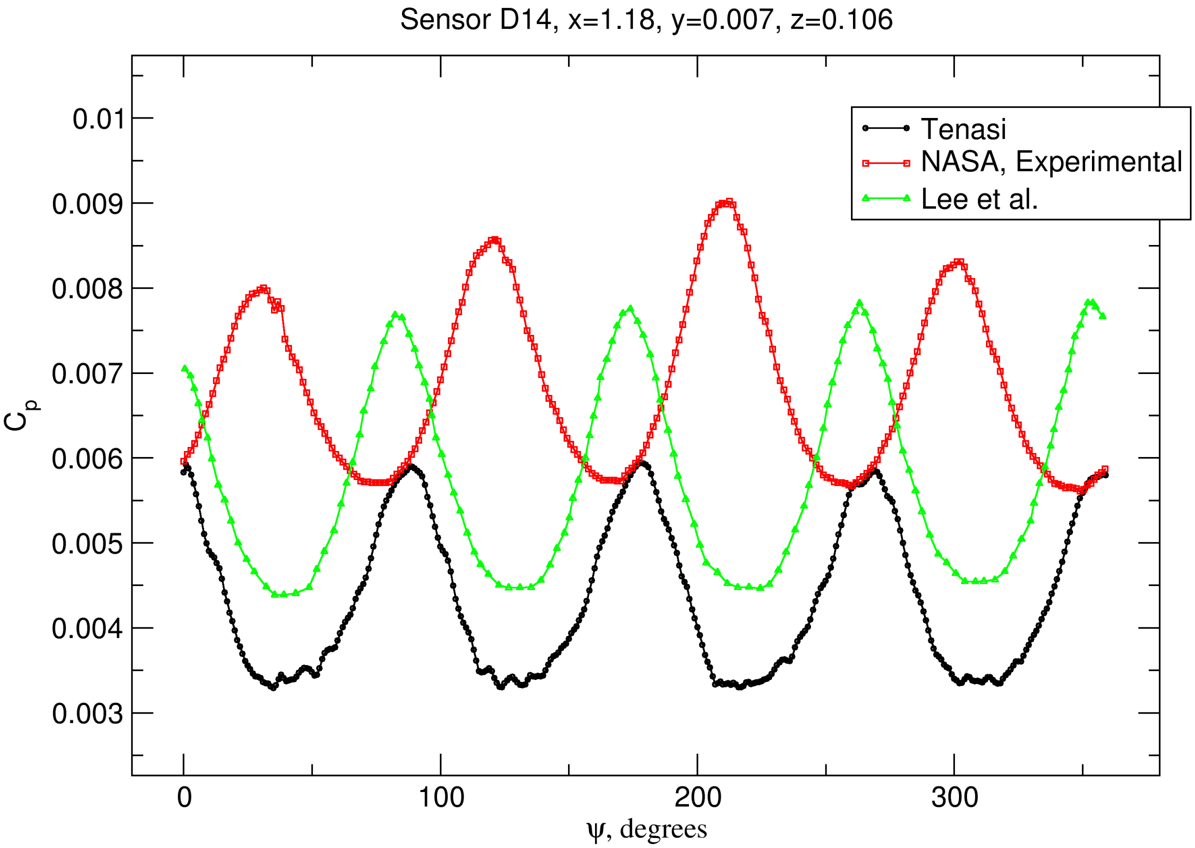


Figure 47 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D14

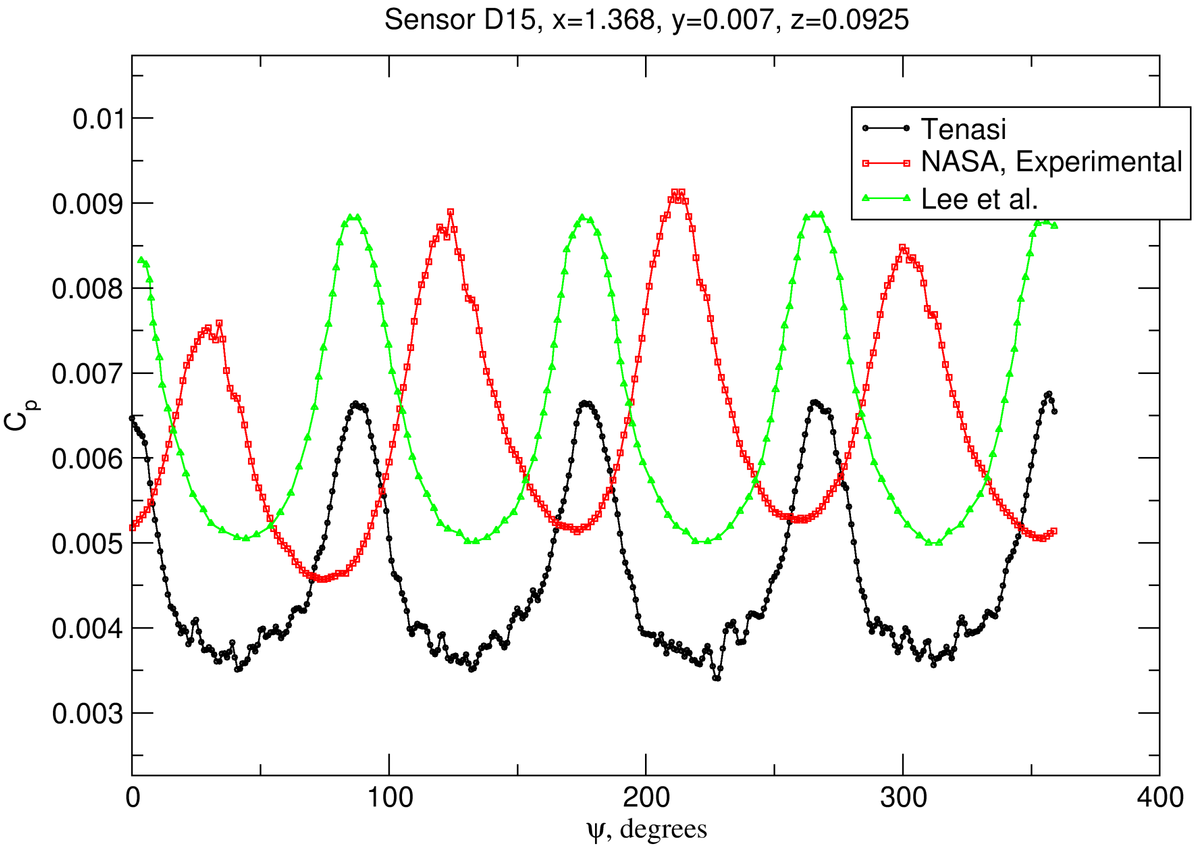


Figure 48 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D15

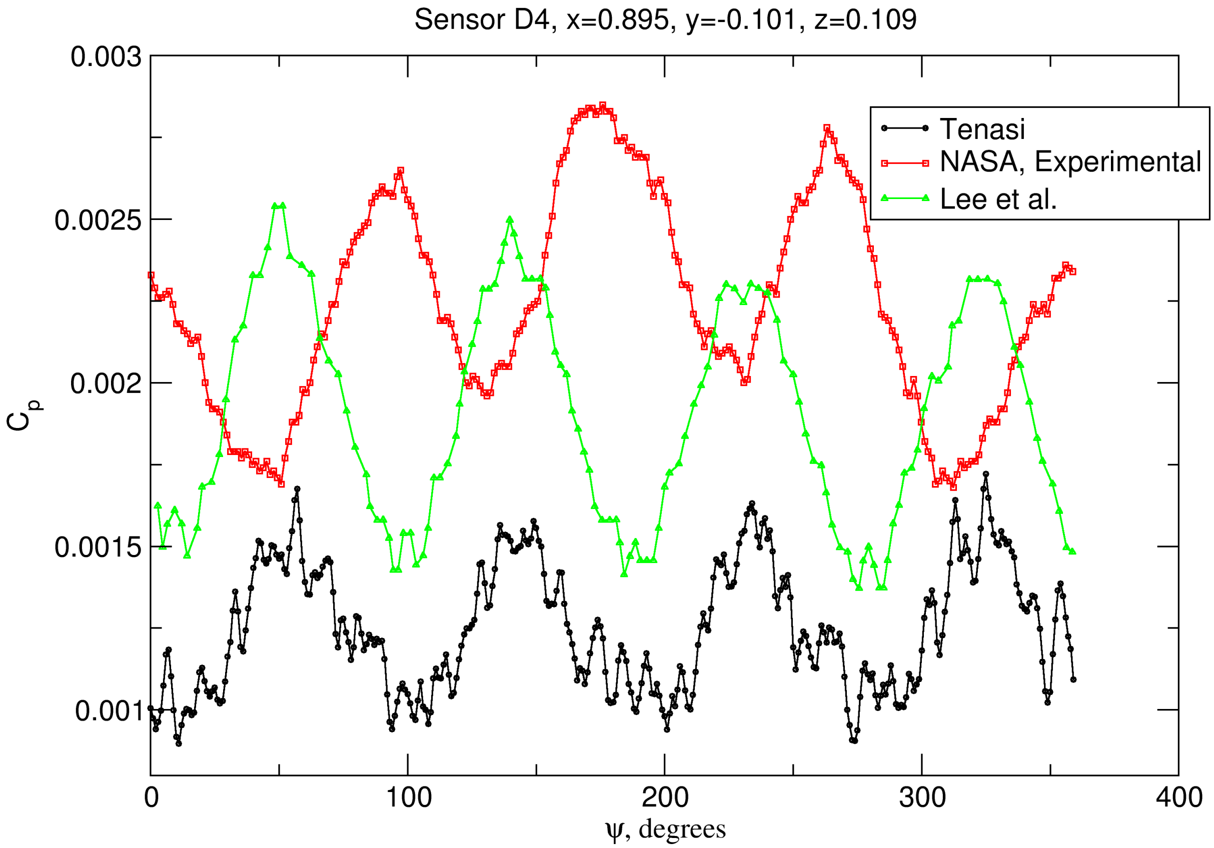


Figure 49 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D4

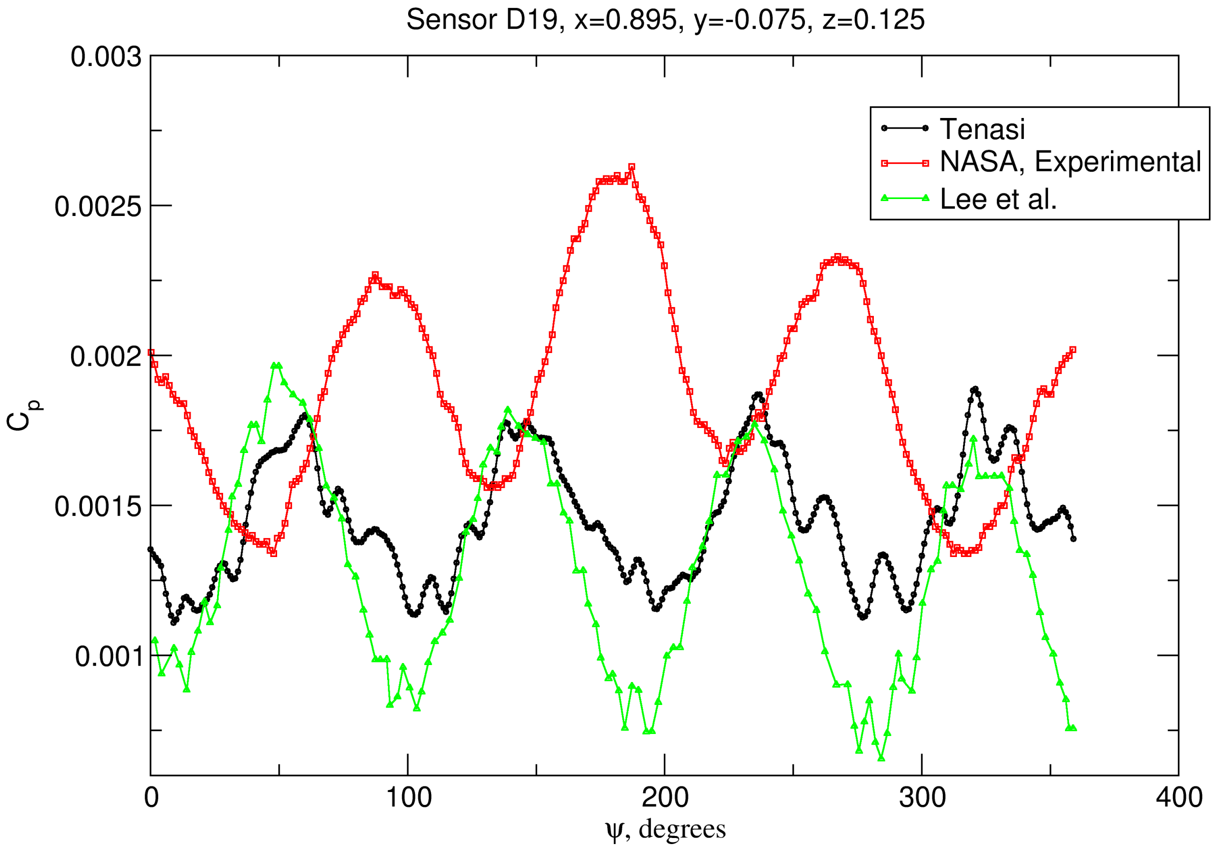


Figure 50 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D19

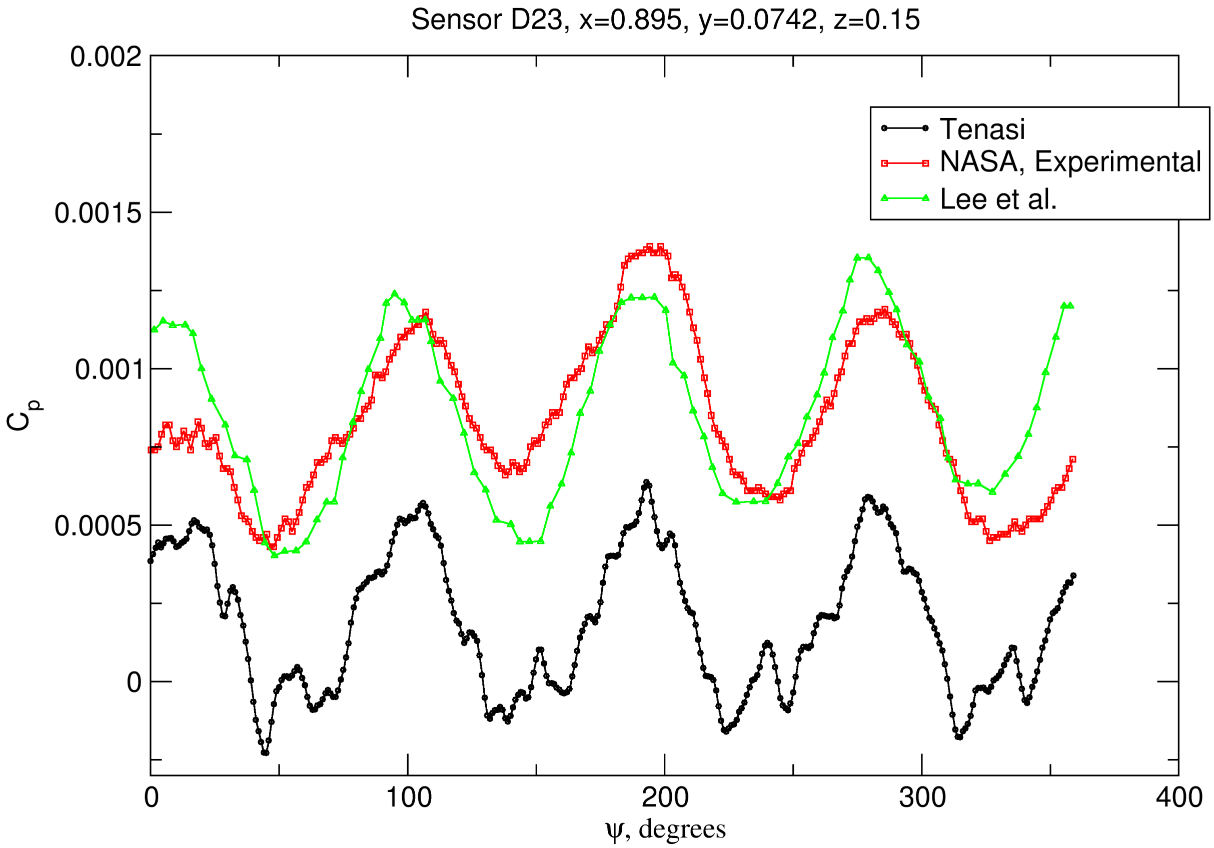


Figure 51 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D23

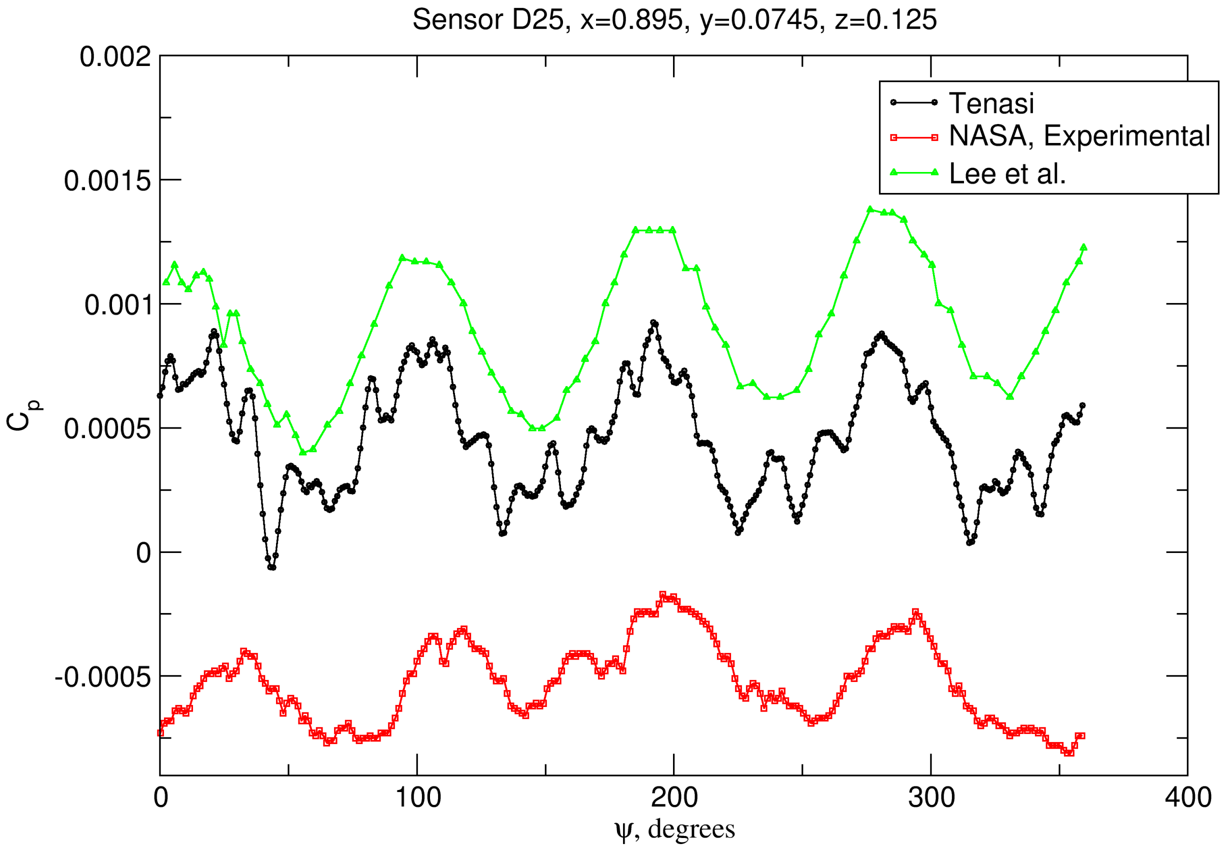


Figure 52 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, D25

There was, for some of these sensors, quite a large discrepancy between results. Further experiments using an increased number of Newton iterations significantly improved accuracy. As mentioned above, it is thought at this time that, while the dual time-stepping technique allows for a stable simulation even when using relatively large time steps for the spatial cell size (in conjunction with a very low maximum CFL specified for local time stepping), many more Newton iterations may be required to fully solve for the residual term in each time step. Figures 53 through 60 show data for the same sensor locations as above, using the same solution restarted and run for three more revolutions, but with 30 Newton iterations in each time step. Note that these values are only for one revolution, i.e. not averaged.

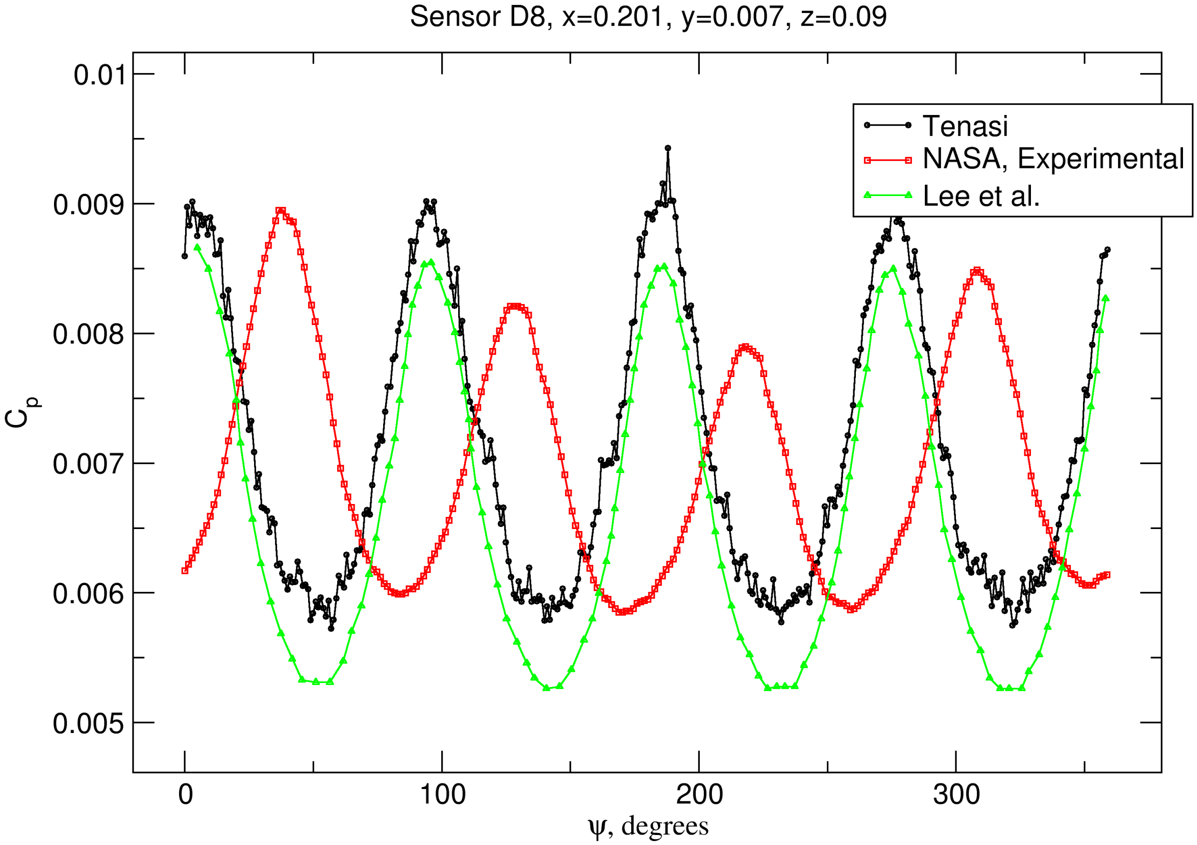


Figure 53 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D8

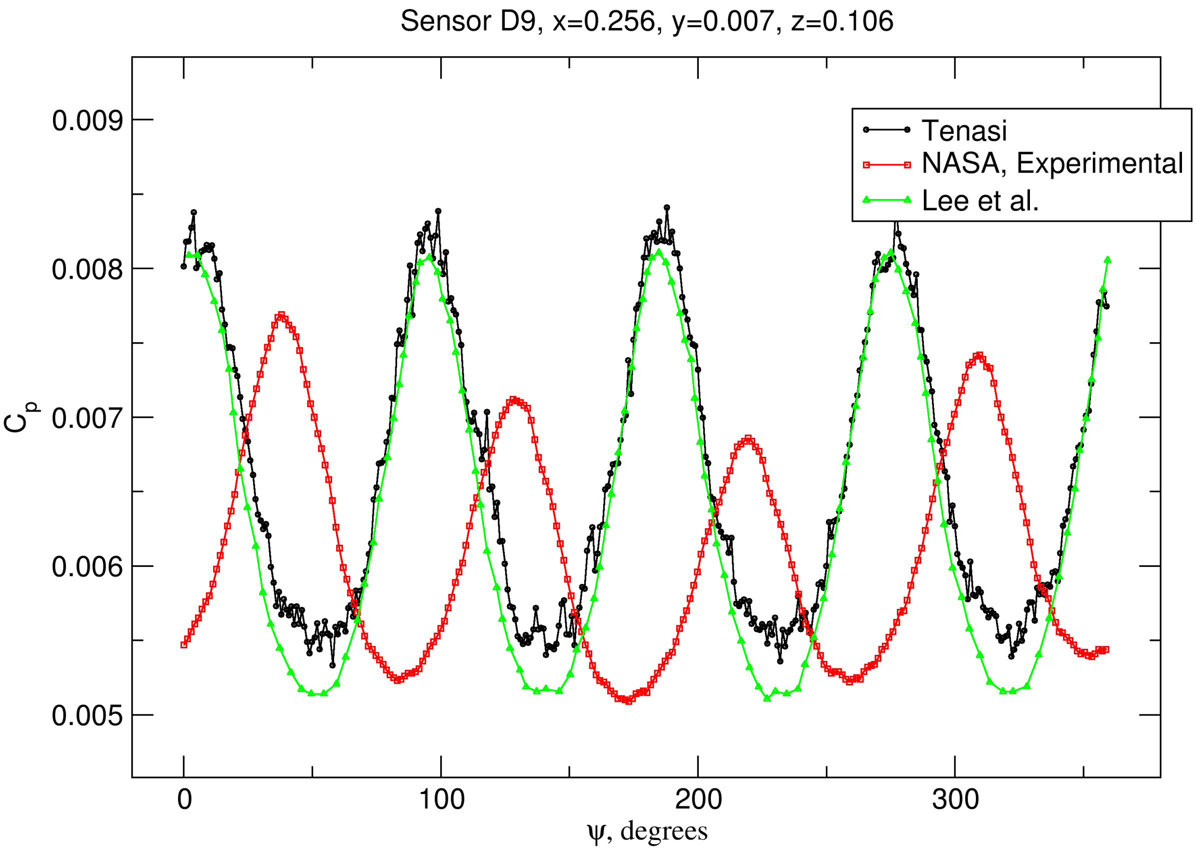


Figure 54 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D9

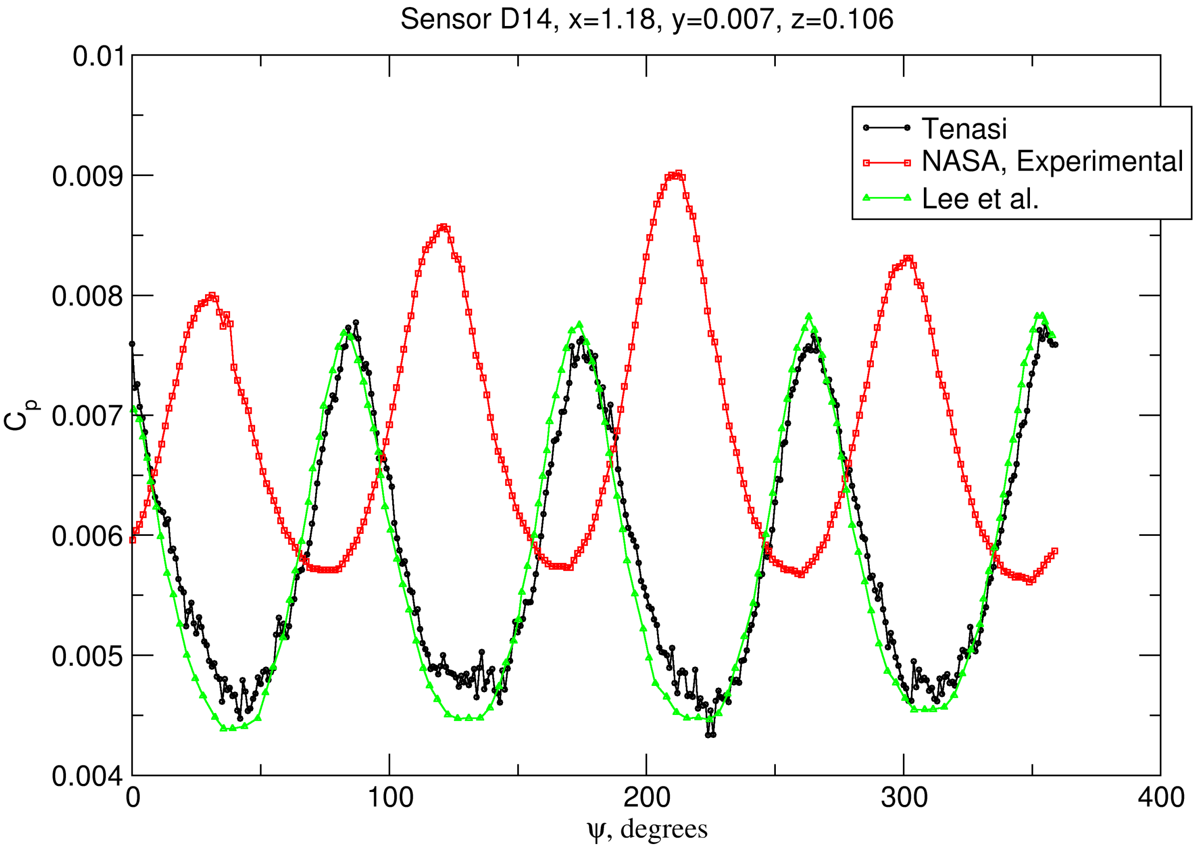


Figure 55 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D14

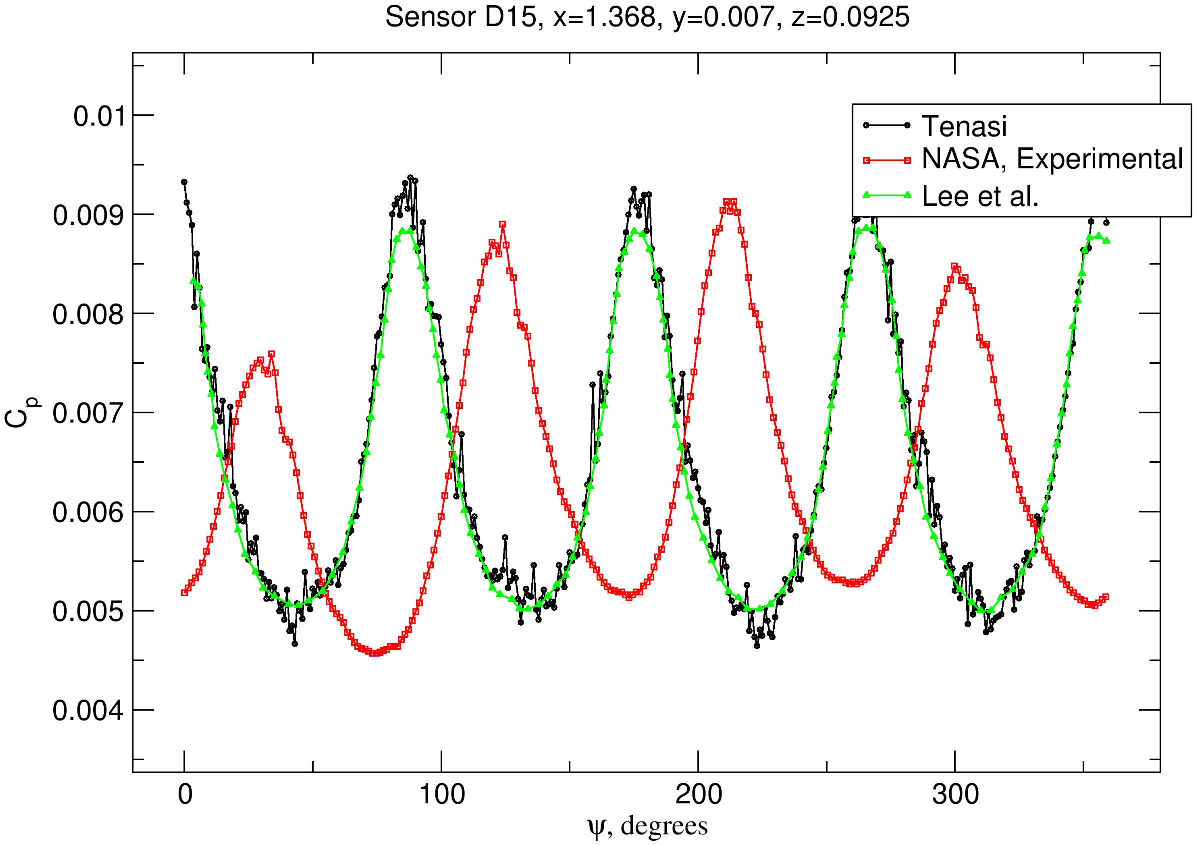


Figure 56 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D15

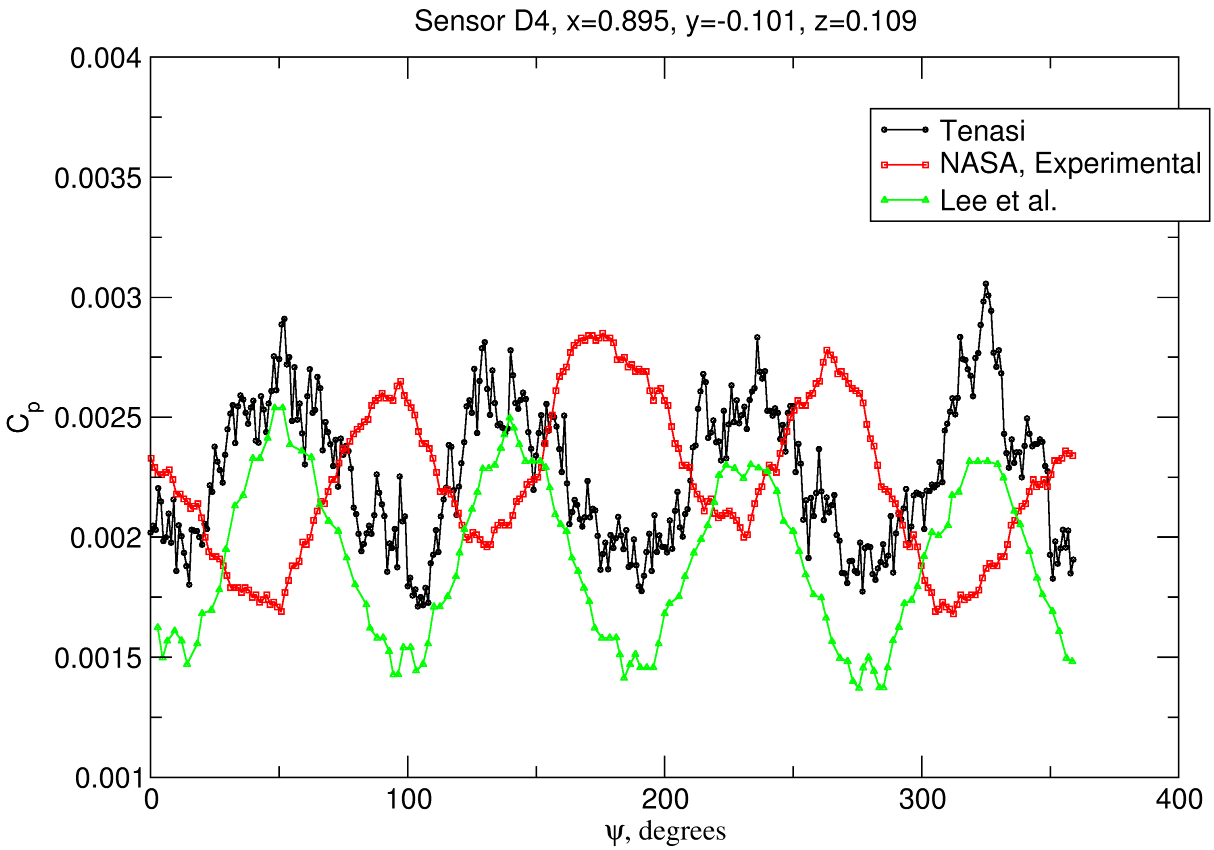


Figure 57 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D4

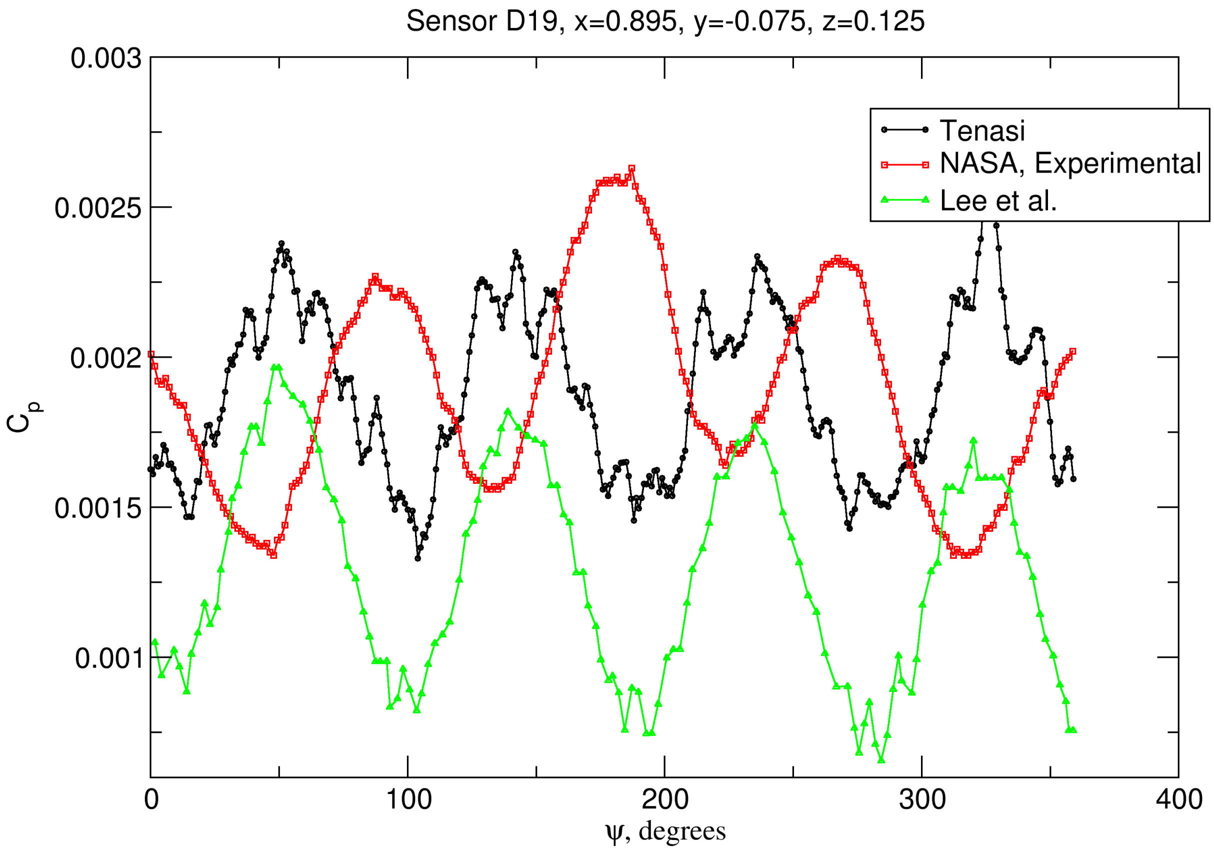


Figure 58 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D19

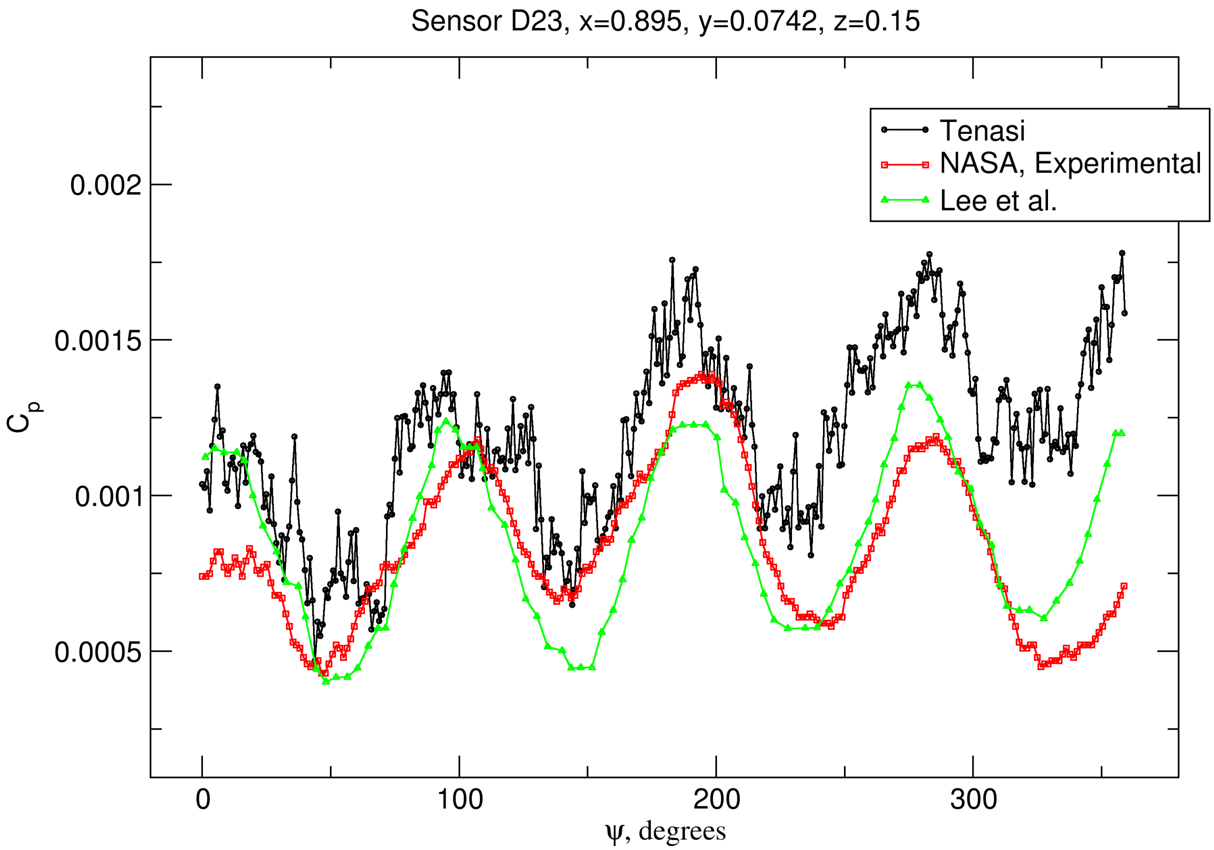


Figure 59 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D23

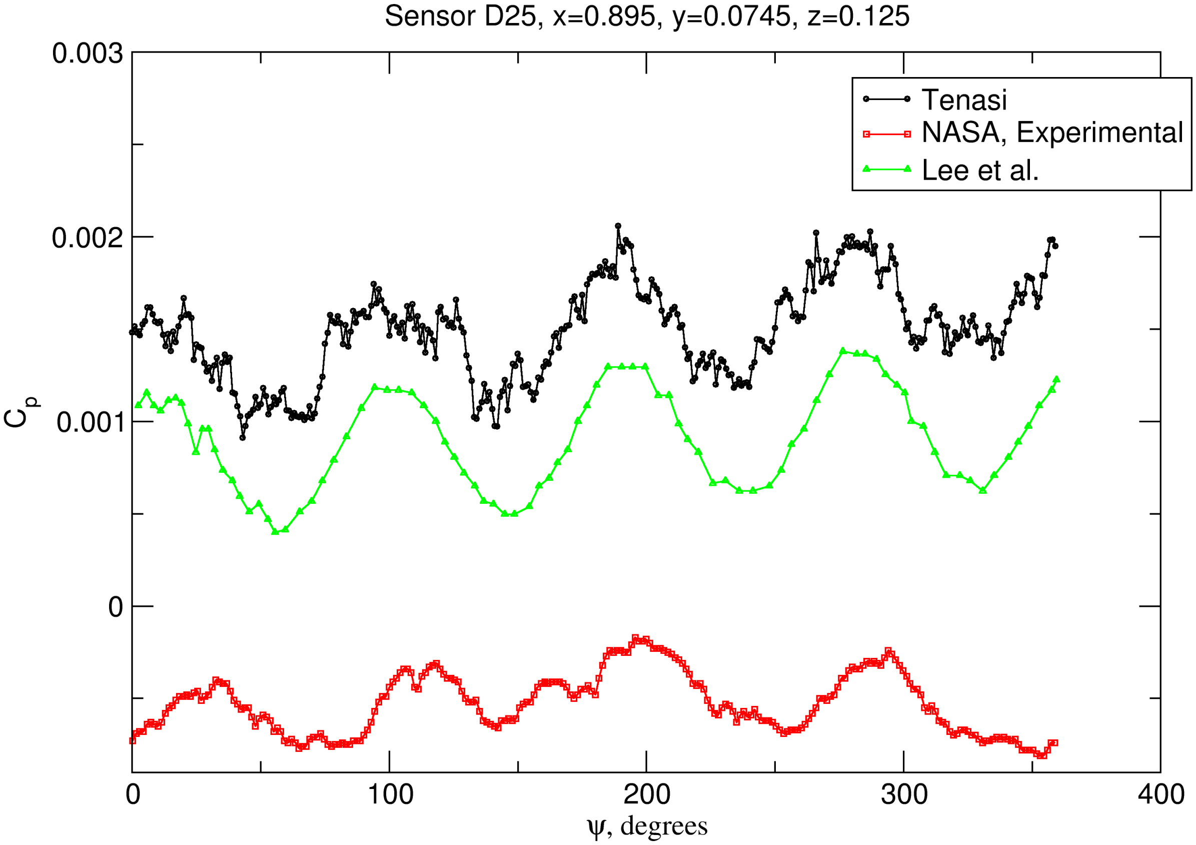


Figure 60 Pressure on ROBIN body vs. blade azimuth, hover μ=0.012, 30 Newton iterations, D25

One final observation on these hover results: note the large difference between results for sensor D19 and D25. These sensors are symmetrically opposed to each other on the sides of the pylon, near where it meets the body. However, results are very different. While this could theoretically be due to the slight offset of the rotor from centerline above the body, it is also likely that here, as in the phase of pressure signal and the collective pitch vs. thrust coefficient correspondence, the NASA results must be regarded as rough guidelines for simulation and adjustments need to be made to account for unknown variables.

The results shown in Figures 61 through 65 include comparison data available from other simulations for the forward flight case (advance ratio $\mu = 0.15$). The rotor trim conditions for the forward flight case were: collective $\theta_0 = 6.5^{\circ}$, longitudinal $\theta_{1s} = 2.2^{\circ}$, lateral $\theta_{1c} = -2.0^{\circ}$. An axial (forward) tilt $\alpha_s = -3.0^{\circ}$ for the rotor was applied to the model and the volume mesh was reinitialized. Since only a few sensor locations were available for comparison with simulation data, results for all sensors as compared with the NASA Langley experiment are presented in Appendix A. It should be noted that for these plots and those in the appendix, the NASA data have been shifted roughly 30º to allow for better comparison.

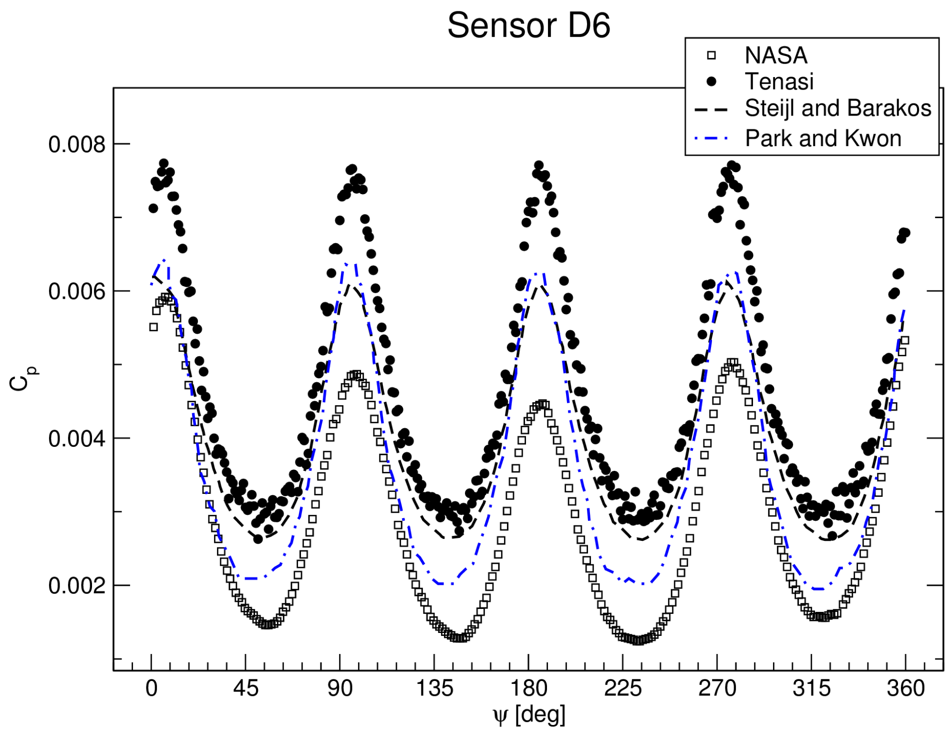


Figure 61 Pressure on ROBIN body vs. blade azimuth, forward flight μ=0.15, D6

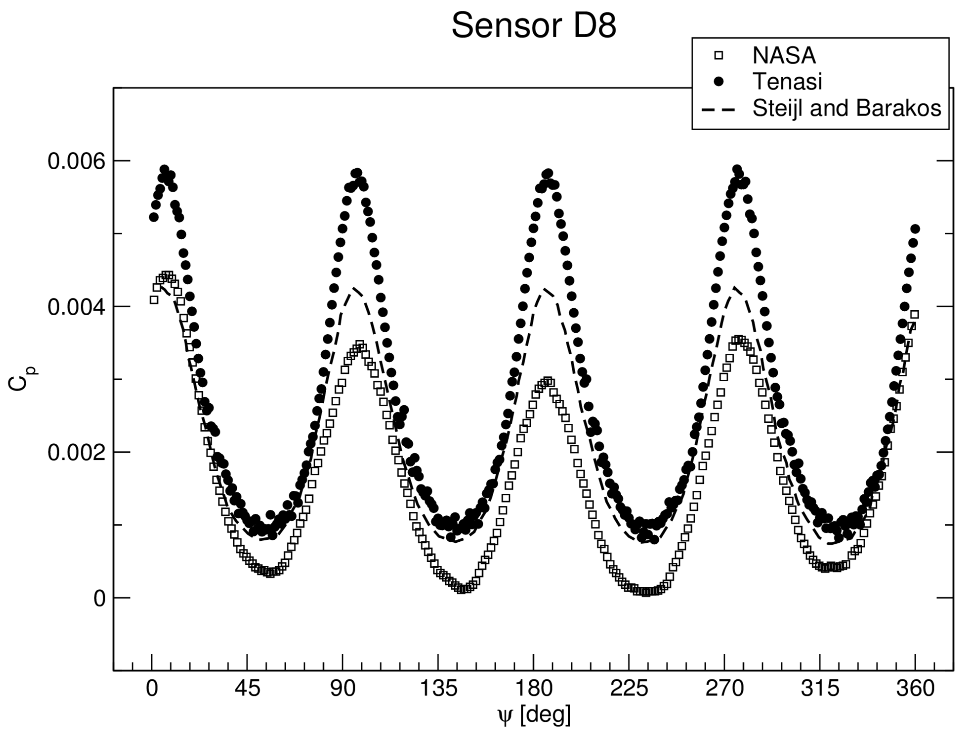


Figure 62 Pressure on ROBIN body vs. blade azimuth, forward flight μ=0.15, D8

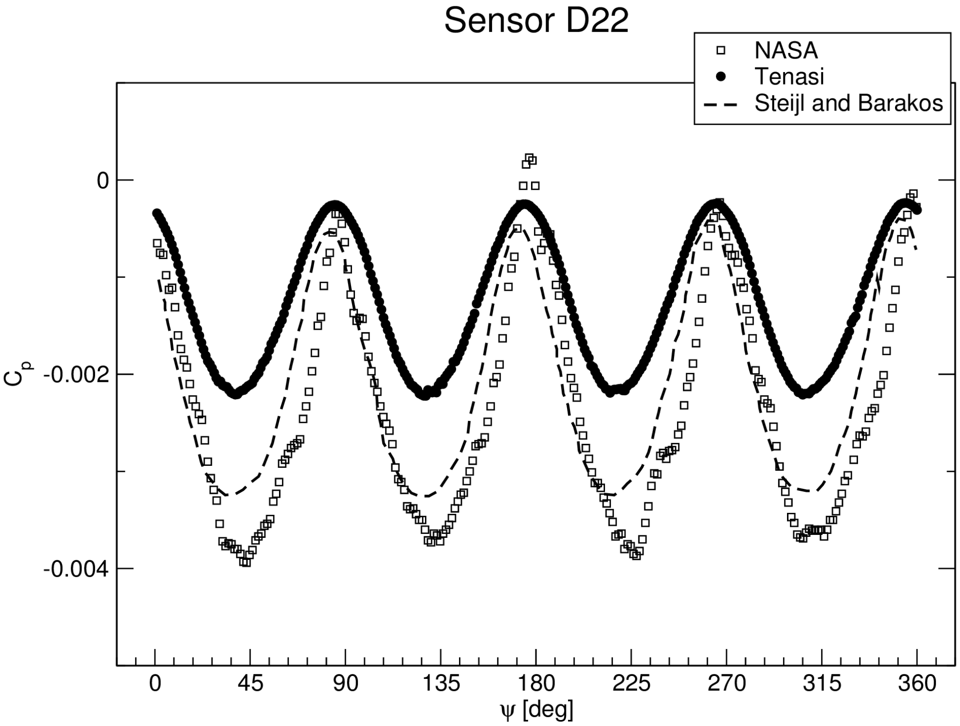


Figure 63 Pressure on ROBIN body vs. blade azimuth, forward flight μ=0.15, D22

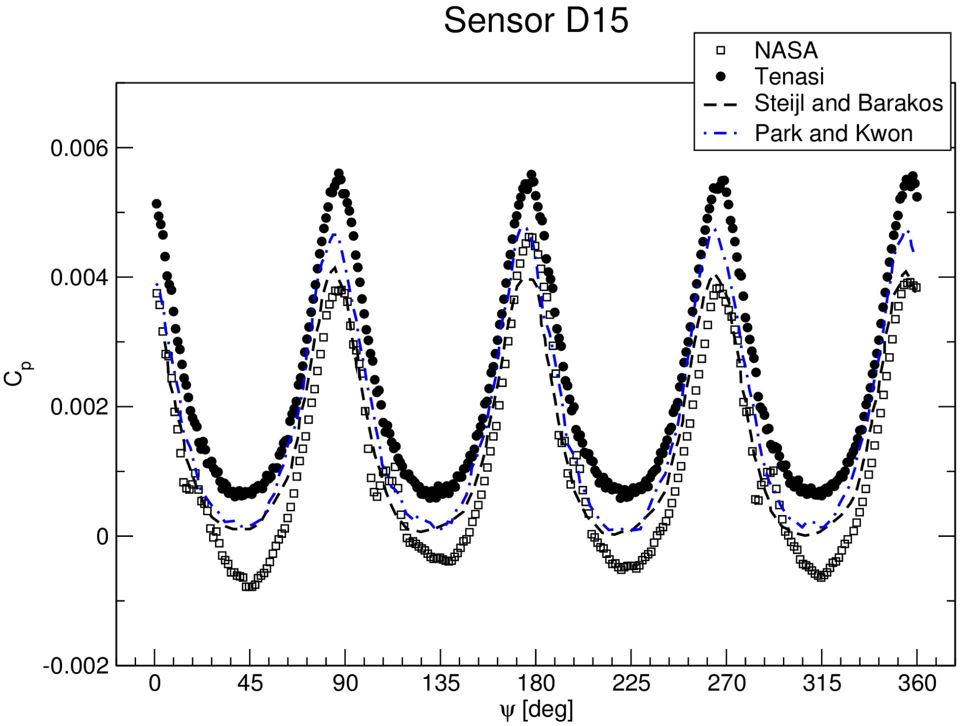


Figure 64 Pressure on ROBIN body vs. blade azimuth, forward flight μ=0.15, D15

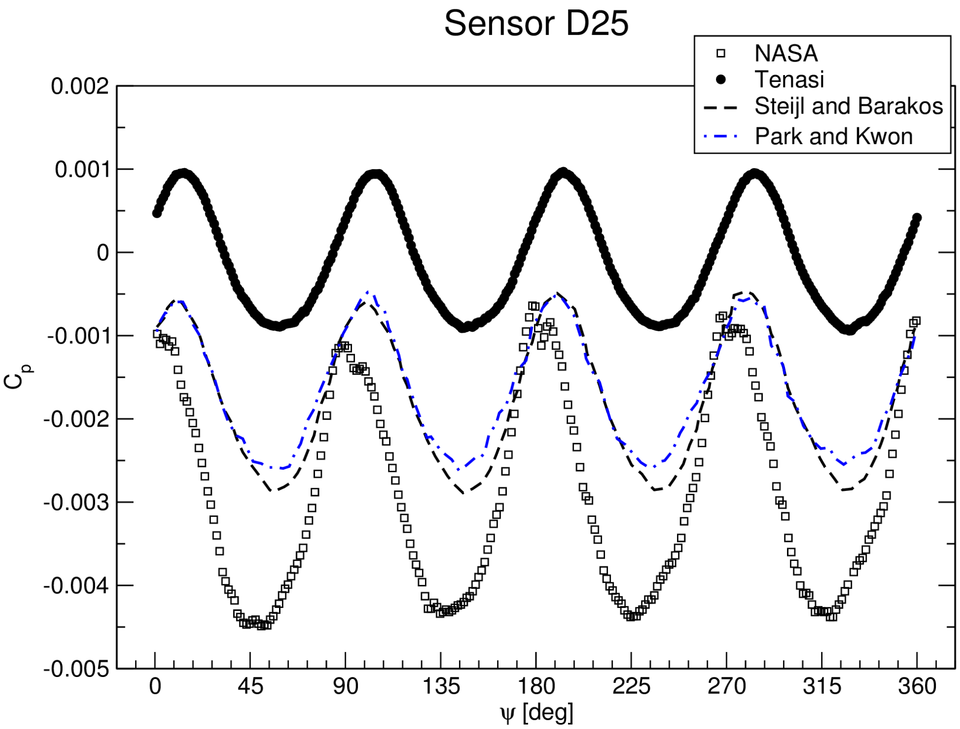


Figure 65 Pressure on ROBIN body vs. blade azimuth, forward flight μ=0.15, D25

The results from Tenasi show a well-established periodic pressure induced on the body. Phase is present as related to blade azimuth ψ, where $\psi = 0$ has been established for blade 1 when it is directly over the aft centerline of the fuselage. The NASA data are not as evenly periodic as CFD results. In the NASA results, each blade has a slightly different effect on body surface pressure as it passes over. This could be due to small imperfections in blade surfaces as constructed, or in differing longitudinal blade rigidity giving rise to flapping magnitude inconsistencies. However, peak-to-peak variation, in terms of magnitude, compares well to other CFD simulations (except perhaps at sensor D25). There is a discrepancy between simulation and experiment in the magnitude of the average pressure coefficient at each tap location. We speculate that this is due to an imperfect optimization across all parameters affecting the outflow velocity vector field, given that optimization was performed on thrust coefficient alone, and furthermore that better results might be obtained at some sensor locations through adaptive grid refinement on velocity gradient magnitude to better capture effects at smaller length scales. In fact, this should be the next step in future work.

For some sensors there is a greater difference in periodic pressure magnitude than others. This could be due to several factors. The rotor hub was modeled for Tenasi as a simple ellipsoidal surface of rotation, not as a full rotor drive system. Also, the employed trim state is an estimation selected to agree with the work of Park, Nam, and Kwon (2003), and it has a greater effect near the tip of the blade (Steijl and Barakos 2009). Therefore, while the generated thrust coefficient for the entire rotor may match acceptably with experiment, the curve of thrust vs. station along blade length may deviate. For sensors downstream of the pylon, i.e. D15, experimental and simulated pressure magnitudes more closely correspond. Surface pressure data for all sensors show periodicity, even where variations in magnitude occur. It should be noted that sensors positioned near body supports that were present in the NASA physical model but were not represented by boundary surfaces in computational models should be expected to show differences; this has also been posited by Steijl and Barakos. Boundary layer flows could also be affected by small variations in the constructed physical model surface that were not represented in the mathematically generated body geometry. We found a sensitivity to sensor positioning in pressure measurements on the body, which can be explained by high pressure gradient magnitude in certain body surface locations, see Figure 66. Given that mesh nodes must be located for sensor sampling output within a given bounding box, mesh spacing in the surface planar direction should be increased for measurement fidelity. Finally, as is noted in the hover case results, more Newton iterations may be necessary with the dual time-stepping technique (which allows enough stability to run the simulation with a fairly large time step, i.e. one timestep per degree of rotation), but for the forward flight case these conditions were not tested due to a lack of time.

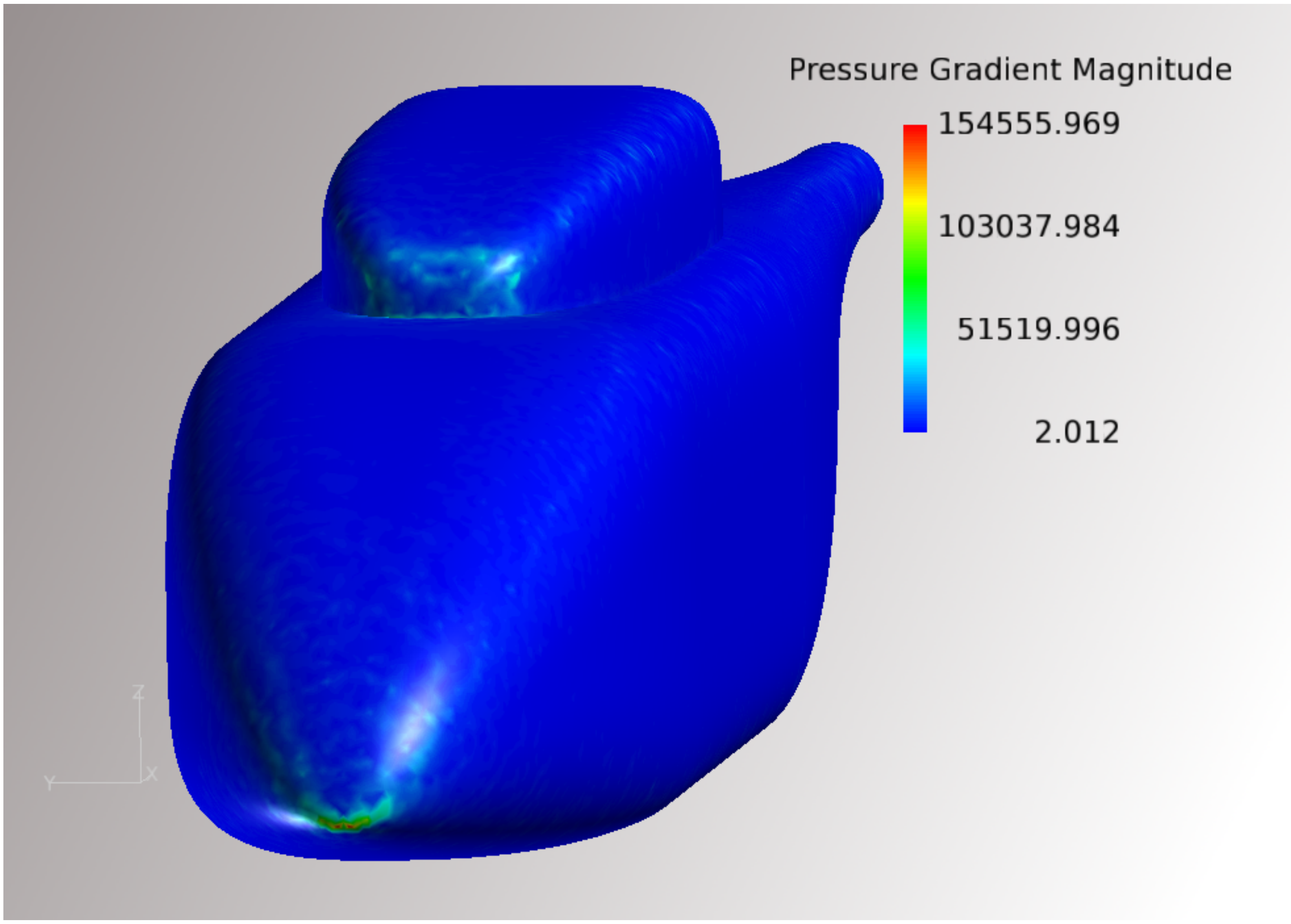


Figure 66 ROBIN unsteady pressure gradient magnitude (Pa) on body surface, timestep 5400

In addition to the coefficient of pressure data, a plot of streamlines seeded in the rotor disk representing rotor blade tip vortex generation and and downstream propagation are presented in Figure 67; comparisons of surface restricted flow from Tenasi with streaklines produced on the physical model at NASA Langley are shown in Figure 68; and instantaneous vorticity magnitude is compared with vorticity contours from Nam et al. in Figure 69 (Nam, Park, and Kwan 2006).

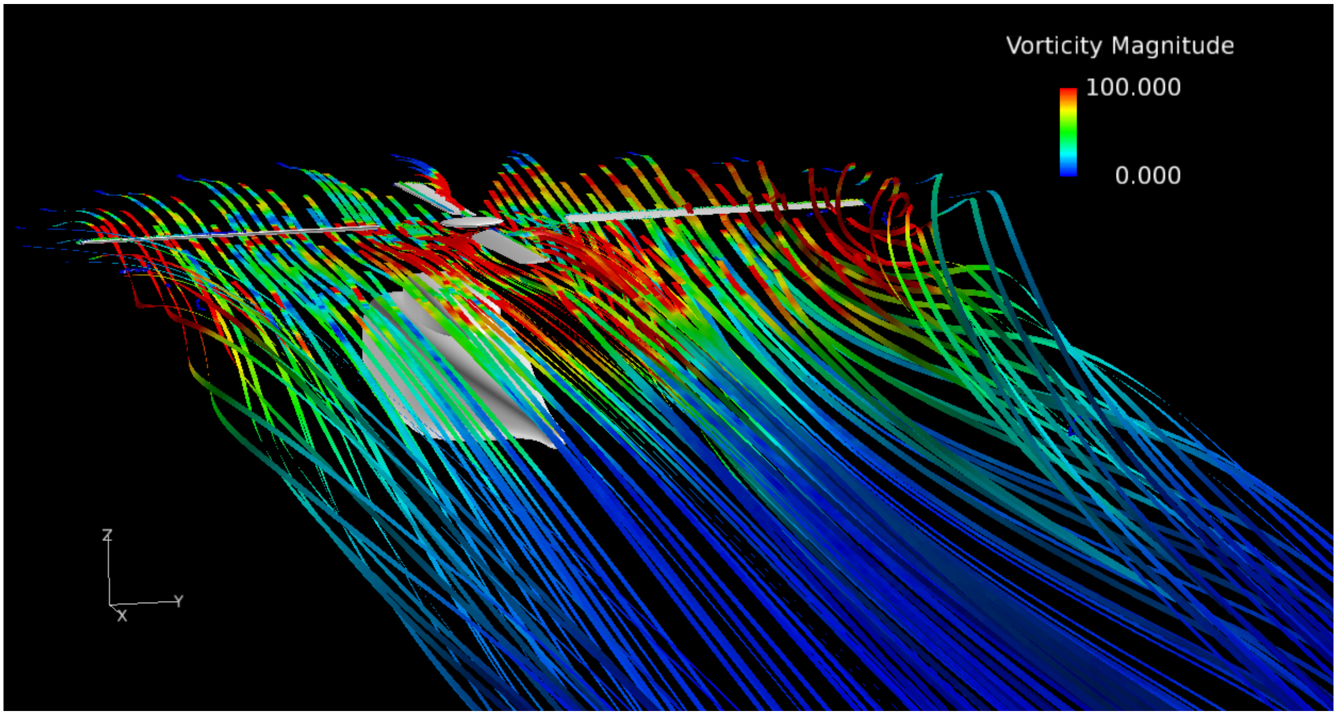


Figure 67 Streamlines from rotor disk with vorticity magnitude (rad/s), forward flight

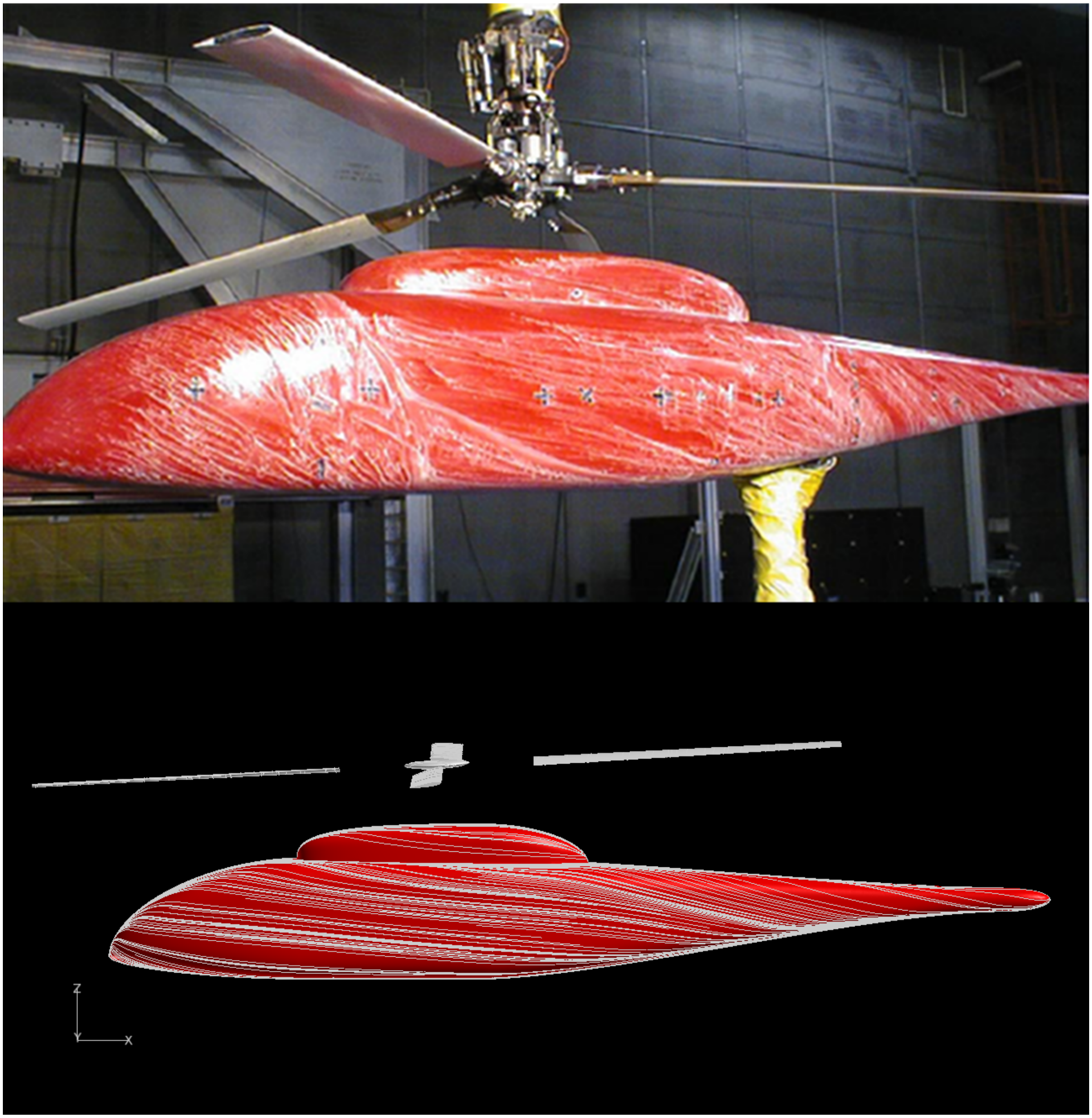
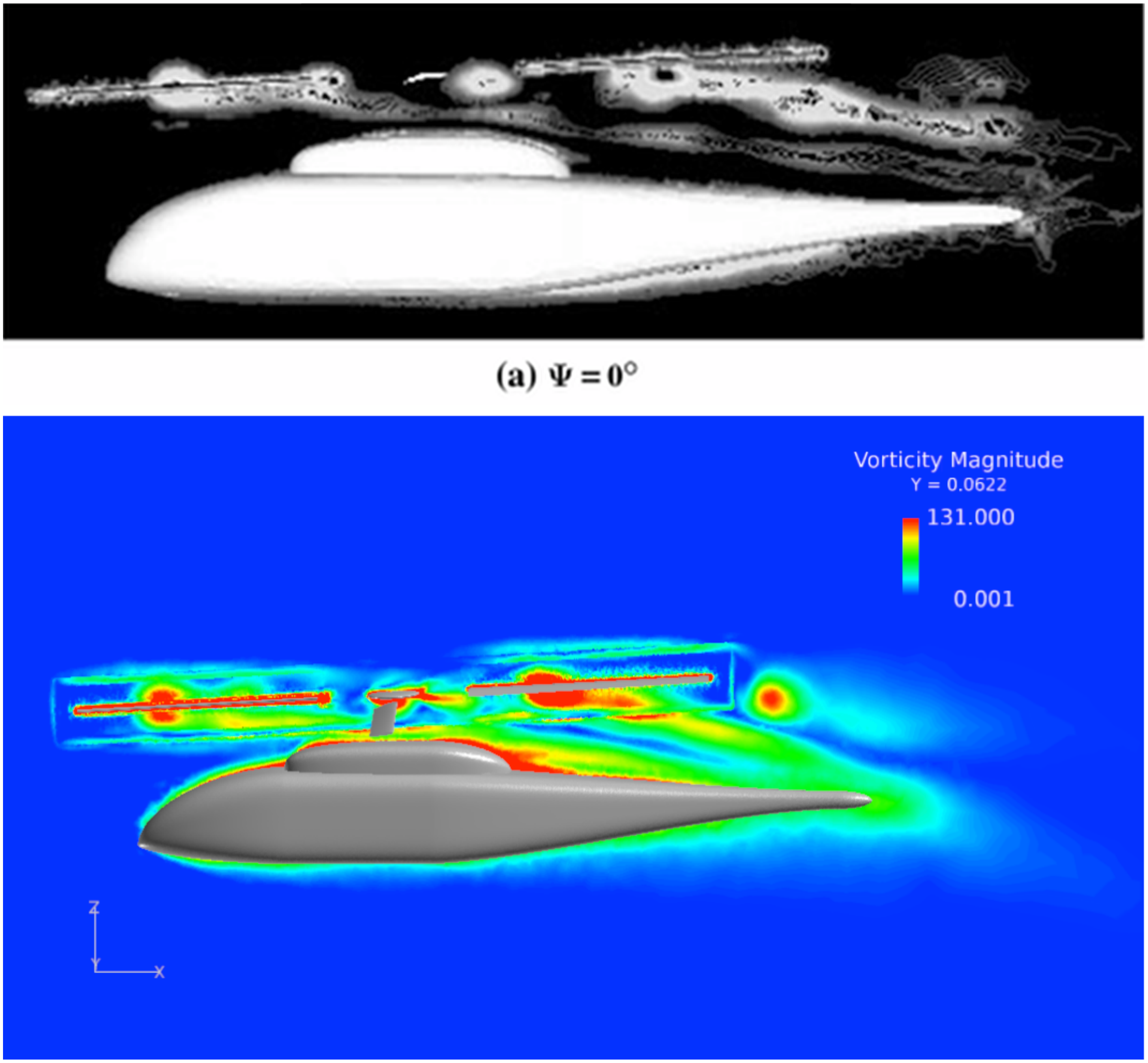


Figure 68 Physical streaklines vs. computed body restricted surface flow streamlines



(a) $\Psi = 0°$

Figure 69 Vorticity contours as shown in Nam et al. (2006) vs. vorticity magnitude (rad/s), Tenasi

CHAPTER V

CONCLUSIONS AND FUTURE WORK

Conclusions

The isolated rotor cases showed that proper flux computation across sliding interfaces allowed for inflow and outflow necessary to produce the appropriate lift as shown in pressure values at all examined blade radial stations, validating the sliding interface implementation in Tenasi. Agreement with previous simulation and experiment was good, giving confidence in the sliding interface technique as well as the cyclic pitching code, once some initial confusion in the sign conventions for the pitching coefficients was resolved and the pressure coefficients were appropriately defined and scaled. In particular, the forward flight case provided an important result in showing that relative frame rotation with cyclic pitching submotion does not match results for absolute frame (grid motion) for rotation and cyclic pitching. With investigation, however, it is possible that the correct terms for fluid acceleration could be derived for grid submotion inside a relative frame, and the advantages of this approach could be recovered.

Steady-state modeling of the isolated ROBIN body compared very favorably with previous models in prediction of surface flow characteristics and provided validation of the constructed body geometry as represented computationally against the wind-tunnel model used at NASA Langley.

The rotor thrust output figures from Tenasi for the ROBIN rotor configuration showed confirmation of the rotor trim characteristics identified by Nam, Park, and Kwan (2006) and used by

Steijl and Barakos (2009) and Lee et al. (2010) in producing a similar coefficient of thrust under simulation. We can posit that adjustments made to cyclic pitch while maintaining a certain thrust coefficient might further improve simulation accuracy, but the sensitivity analysis is outside the scope of the current work. It was found, however, upon an investigation of the literature, that no simulation has been able to reproduce the nominal thrust coefficients from wind-tunnel experiments using the stated parameters for collective and cyclic pitching, relying instead on comptational design optimization techniques to reproduce results. We can only surmise that there may have been either measurement inaccuracy or clerical error.

The unsteady ROBIN results presented herein, especially when a larger number of Newton iterations are applied to unsteady calculations, still provide confirmation of the sliding interface methodology currently implemented in Tenasi both in terms of a reasonable approximation of transport across the interface and of the compuational simulation results as compared to both real-world wind-tunnel models and to other flow solvers. With adaptive refinement techniques and further modeling of features present in the wind tunnel testing (i.e. shapes representing the body support, rotor drive system, etc.), it is reasonable to assume that the predicted results would more closely match observed values for body surface pressure coefficient vs. blade azimuth. Regardless, periodicity of body surface flow effects generated by the rotor blade passage over the fuselage is well-established in the ROBIN case. This interaction is an important area of study in rotorcraft aerodynamics, and the sliding interface technique enables its simulation.

Future Work

Several techniques might be applied for improvement of the results obtained. The first step for the ROBIN model should be an investigation of the number of Newton iterations necessary to compensate for the large time step that is enabled by the dual time-stepping technique. Also, the application of adaptive refinement techniques to improve solution fidelity in areas of high gradient magnitude and reduce the computational effort required in areas that have low gradients and/or do not affect fluid transport between the rotor and body.

Both cyclic and collective pitching conditions could be optimized for the desired thrust coefficient. Regarding this, a flapping model could be implemented that would calculate normal stresses on rotor blades and employ finite-element linear-elastic techniques to estimate the amount of blade deformation under load, then translate that to an equivalent pitching motion. In this way, higher simulation fidelity of actual rotor conditions in-flight could be achieved without increasing the grid spacing necessary to model rotors having highly elastic blades. This would improve the applicability of the solver to different rotorcraft. In some rotorcraft (notional or production), there is not enough space for axisymmetrical sliding interfaces around the individual blades, therefore blade pitching would need to be implemented using, for example, mesh deformation techniques with definition of custom cyclic pitching motion functions for the blade boundary surfaces. Python extension method callouts already exist in Tenasi that could be used for this purpose. With this, validation of fuselage surface flow effects against existing data for this rotorcraft would provide another useful comparison case.

Data are also available from the cited wind-tunnel experiments for several different advance ratios and rotor trim conditions (giving different rotor thrust coefficients). Other experimental data are available for rotorcraft having tail rotors. These could warrant further study.

REFERENCES

Blades, Eric L., and David L. Marcum. 2005. “A Sliding Interface Method For Unsteady Unstructured Flow Simulations.” In *Paper 5233, 17th AIAA Computational Fluid Dynamics Conference*, 13. 2005.

Brand, A. G., H. M. McMahon, and N. M. Komerath. 1989. “Surface Pressure Measurements on a Body Subject to Vortex Wake Interaction.” *AIAA Journal* 27 (5) (May): 569–574.

Caradonna, F.X., and L. Tung. 1981. “Experimental and Analytical Studies of a Model Helicopter Rotor in Hover.” *Vertica* 5: 149–161.

Freeman, Carl E., and Raymond E. Mineck. 1979. “Fuselage Surface Pressure Measurements of A Helicopter Wind-tunnel Model with A 3.15-meter Diameter Single Rotor”. Technical Memorandum 80051. NASA.

Ghosh, Amrit. 1996. “Solutions to Three-dimensional Thin-layer Navier-Stokes Equations in Rotating Coordinates for Flow Through Turbomachinery”. M.S. Thesis, Mississippi State University.

Hyams, Daniel. 2003. “An Investigation of Parallel Implicit Solution Algorithms for Incompressible Flows on Unstructured Topologies”. Ph.D. Dissertation, Mississippi State University.

Karman, S. L. 2007. “Unstructured Viscous Layer Insertion Using Linear-Elastic Smoothing.” *AIAA Journal* 45 (1) (January).

Kenyon, Adam R., and Richard E. Brown. 2009. “Wake Dynamics and Rotor–Fuselage Aerodynamic Interactions.” *Journal of the American Helicopter Society* 54 (012003): 1–18. doi:10.4050/JAHS.54.012003.

Lee, Bum Seok, Mun Seung Jung, Oh Joon Kwon, and Hee Jung Kang. 2010. “Numerical Simulation of Rotor-Fuselage Aerodynamic Interaction Using an Unstructured Overset Mesh Technique.” *Int’l J. of Aeronautical & Space Sciences* 11 (1) (March): 9. doi:10.5139/IJASS.2010. 11. 1.001.

Löhner, Rainald, and John Ambrosiano. 1990. “A Vectorized Particle Tracer for Unstructured Grids.” *Journal of Computational Physics* 91: 22–31.

Mineck, Raymond E. 1999. “Application of an Unstructured Grid Navier-Stokes Solver to a Generic Helicopter Body”. 209510. Langley Research Center, Hampton, Virginia: NASA.

Mineck, Raymond E., and Susan Althoff Gorton. 2000. “Steady and Periodic Pressure Measurements on a Generic Helicopter Fuselage Model in the Presence of a Rotor”. 210286. NASA.

Mitchell, Brent J. C. 2007. “A Nonmatching and Sliding Interface Method for Unstructured Flow Simulations”. M.S. Thesis, University of Tennessee at Chattanooga.

Nam, Hwa Jin, Young Min Park, and Oh Joon Kwan. 2006. “Simulation of Unsteady Rotor-Fuselage Aerodynamic Interaction Using Unstructured Adaptive Meshes.” *Journal of the American Helicopter Society* 51 (2) (April): 141–148.

Park, Young Min, and Oh Joon Kwon. 2004. “Simulation of Unsteady Rotor Flow Field Using Unstructured Adaptive Sliding Meshes.” *Journal of the American Helicopter Society* 49 (4) (October): 391–400.

Park, Young Min, Hwa Jin Nam, and Oh Joon Kwon. 2003. “Simulation of Unsteady Rotor-Fuselage Interactions Using Unstructured Adaptive Meshes.” In *American Helicopter Society 59th Annual Forum*. Phoenix, AZ: American Helicopter Society. http://dspace.kaist.ac.kr/bitstream/10203/10409/1/J23.pdf.

Pointwise, Inc. 2010. *Pointwise User Manual*. Pointwise, Inc. 213 S. Jennings Ave. Fort Worth, Texas 76104-1107, USA.

Schweitzer, Steven. 1999. “Computational Simulation of Flow Around Helicopter Fuselages”. M.S. Thesis, The Pennsylvania State University.

Sreenivas, Kidambi, Daniel Hyams, Stephen Nichols, Brent Mitchell, Lafe Taylor, Roger Briley, and David Whitfield. 2005. “Development of an Unstructured Parallel Flow Solver for Arbitrary Mach Numbers.” In *43rd AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Inc.

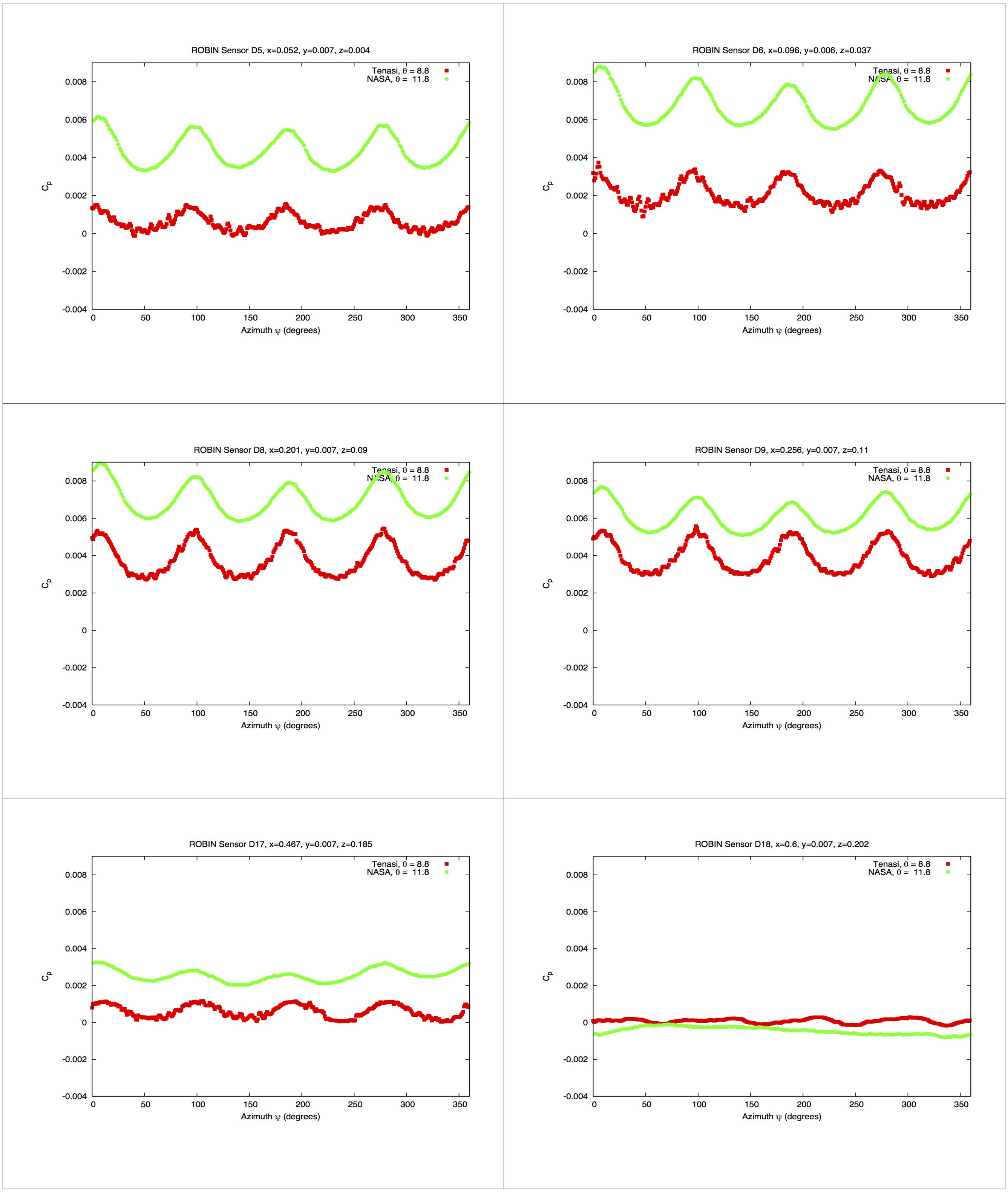
Steijl, R., and G. N. Barakos. 2009. “Computational Study of Helicopter Rotor-Fuselage Aerodynamic Interactions.” *AIAA Journal* 47 (9) (September): 2143–2157. doi:10.2514/1.41287.

Tenasi Documentation. 2012. Release 1.0. University of Tennessee at Chattanooga: SimCenter: National Center for Computational Engineering.

Webster, Robert Samuel. 1994. “Numerical Simulation of a Helicopter Rotor in Hover and Forward Flight”. M.S. Thesis, Mississippi State University.

APPENDIX A

FULL ROBIN UNSTEADY RESULTS, FORWARD FLIGHT


ROBIN Sensor D5, x=0.052, y=0.007, z=0.004

ROBIN Sensor D6, x=0.096, y=0.006, z=0.037

ROBIN Sensor D8, x=0.201, y=0.007, z=0.09

ROBIN Sensor D9, x=0.256, y=0.007, z=0.11

ROBIN Sensor D17, x=0.467, y=0.007, z=0.185

ROBIN Sensor D18, x=0.6, y=0.007, z=0.202

Figure 70 Pressure coefficient vs. rotor azimuth, sensors D5, D6, D8, D9, D17, D18

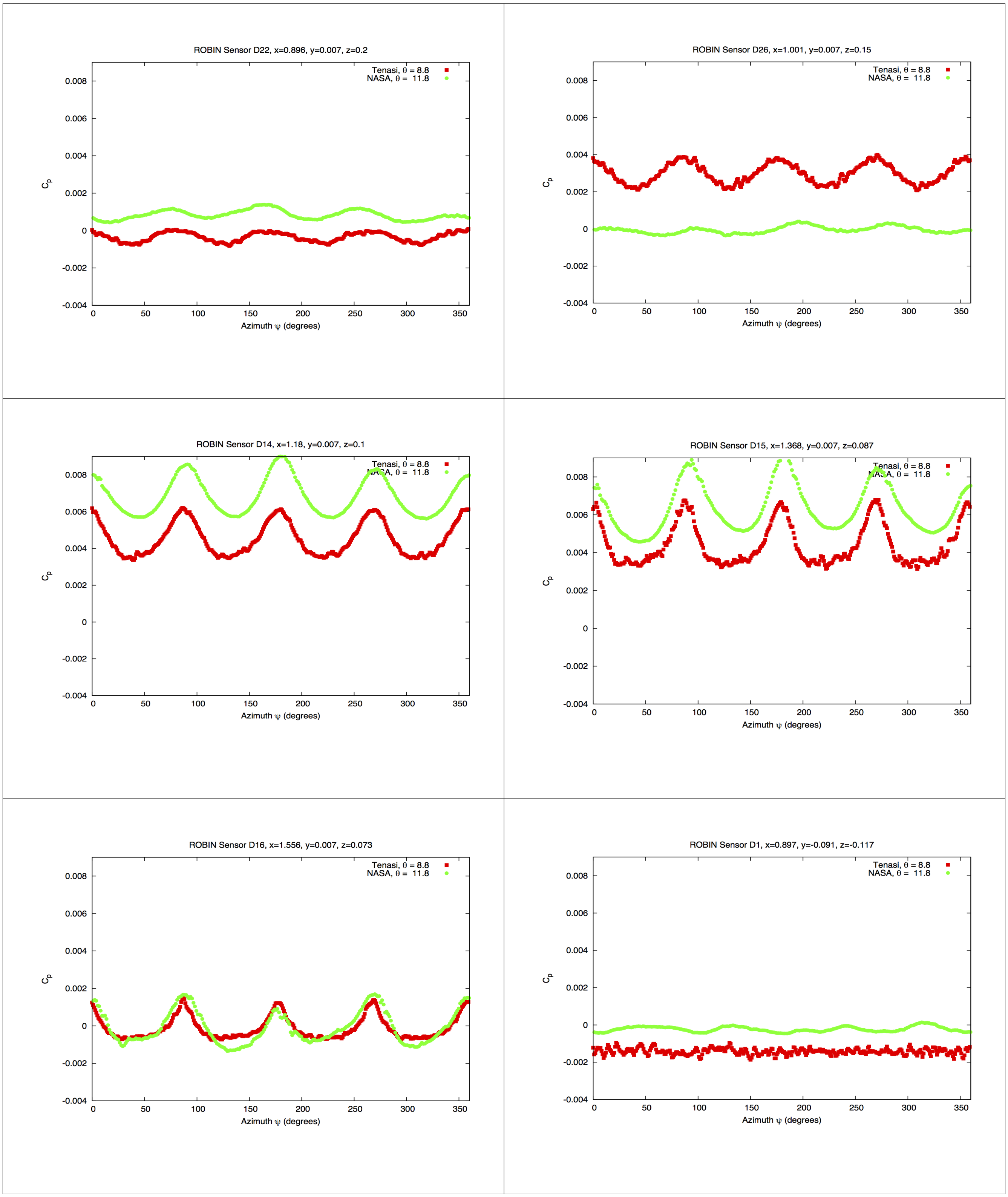


Figure 71 Pressure coefficient vs. rotor azimuth, sensors D22, D26, D14, D15, D16, D1

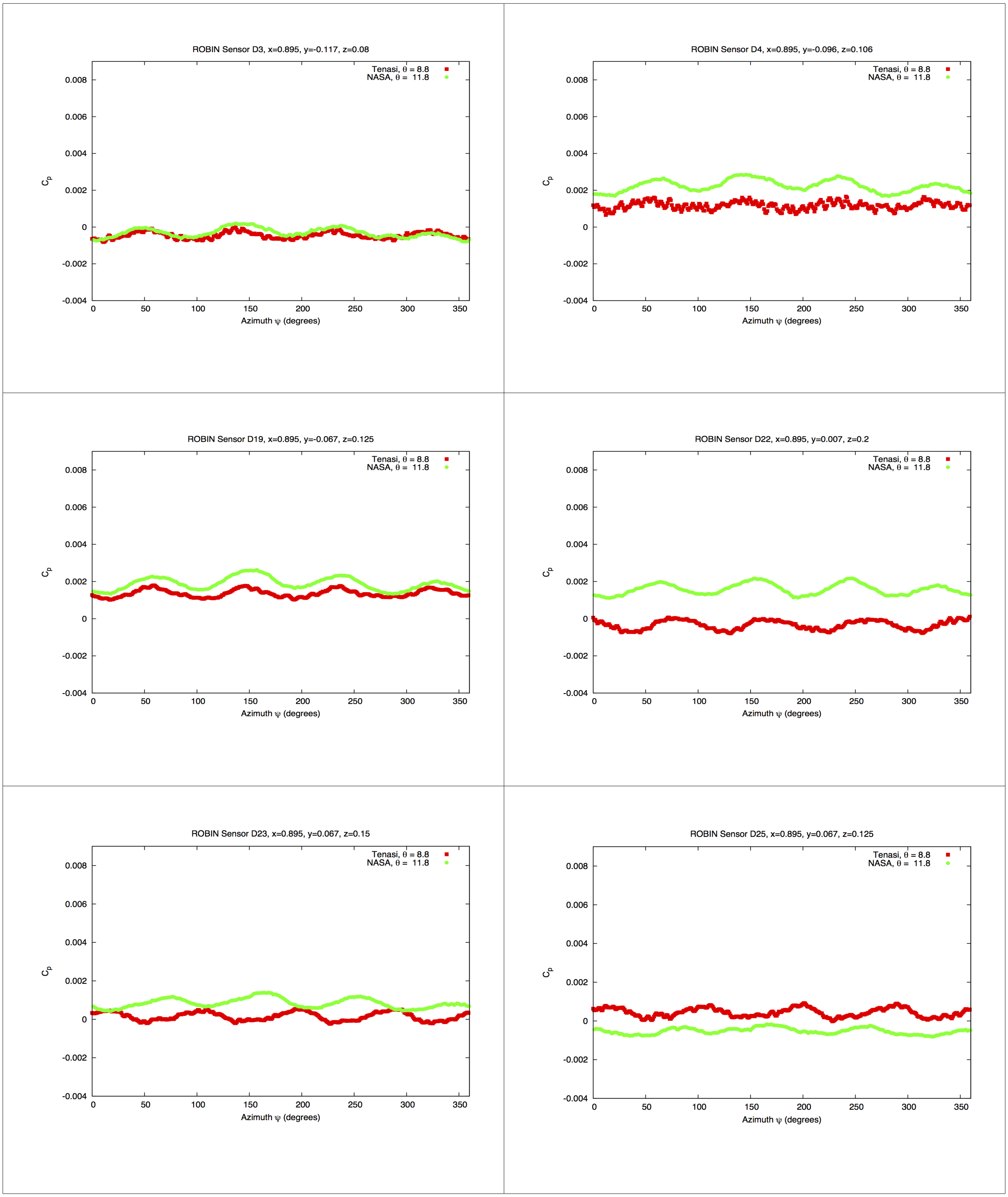


Figure 72 Pressure coefficient vs. rotor azimuth, sensors D3, D4, D19, D22, D23, D25

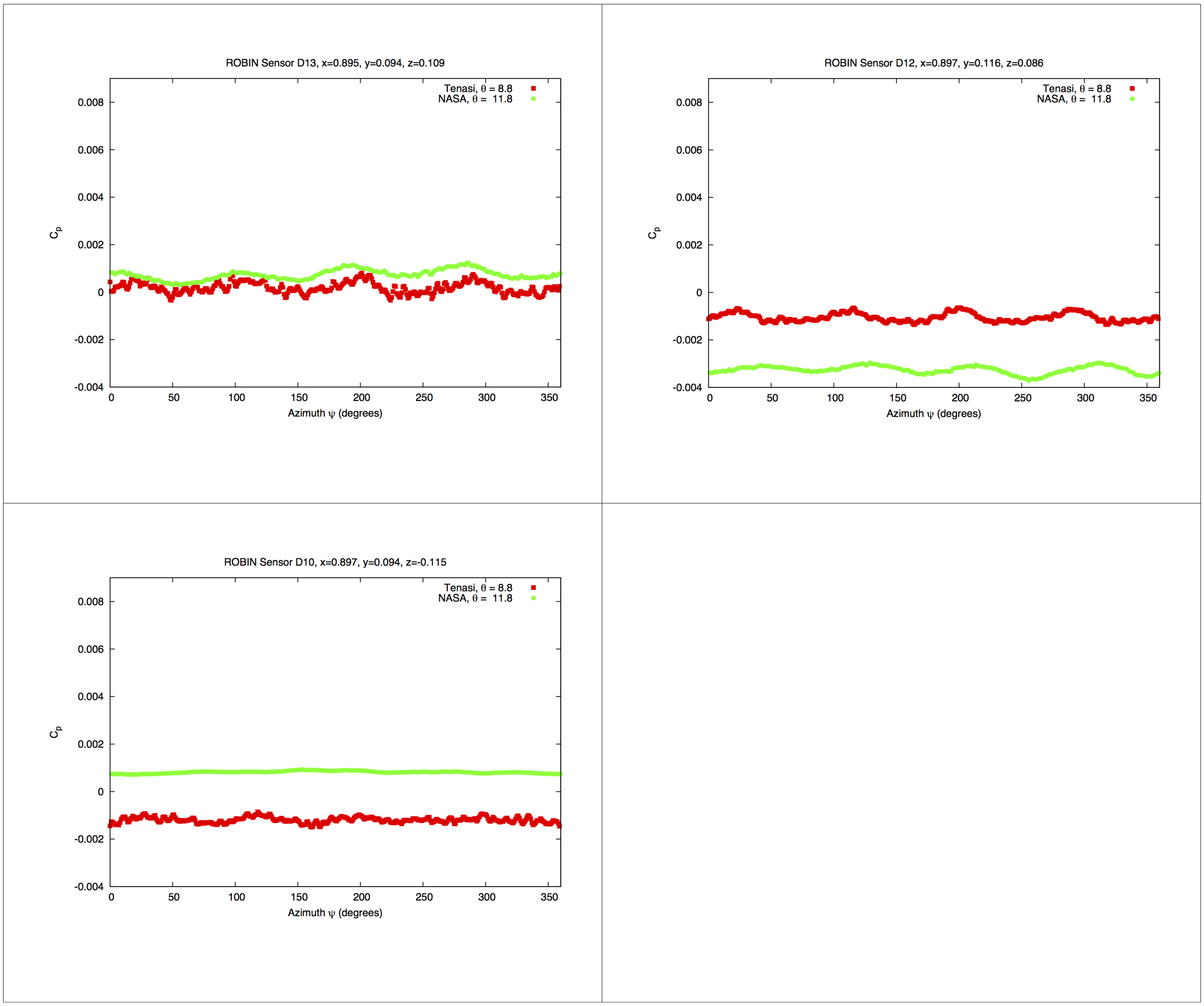


Figure 73 Pressure coefficient vs. rotor azimuth, sensors D13, D12, D10

APPENDIX B

LISTING OF EXTENSION MODULE CODE FOR SENSOR OUTPUT

```
#!/usr/bin/env python
##############################################################################
# ROBIN simulations
##############################################################################
import math,numpy,sys
import simcenter

def UX_get_nsensor(uxp,sdb,grid):
   print "PYTHON: changing nsensor."
   grid['nsensor'][0] = 21
   return 0

def UX_get_sensor_points(uxp,sdb,grid):
   print "PYTHON: getting sensor points."
   nsensor = grid['nsensor']
   sxyz = grid['xyz']
   if nsensor == 0:
      return 0
   sxyz[1,:] = [0.052, 0.007, 0.008653422]
   sxyz[2,:] = [0.096, 0.006, 0.0414404]
   sxyz[3,:] = [0.201, 0.007, 0.0901934625737]
   sxyz[4,:] = [0.256, 0.007, 0.106195857265]
   sxyz[5,:] = [0.467, 0.007, 0.179423627271]
   sxyz[6,:] = [0.6, 0.007, 0.194343339855]
   sxyz[7,:] = [0.895, 0.007, 0.187737186317]
   sxyz[8,:] = [1.001, 0.007, 0.145458628794]
   sxyz[9,:] = [1.18, 0.007, 0.106102032154]
   sxyz[10,:] = [1.368, 0.007, 0.0926950824708]
   sxyz[11,:] = [1.556, 0.007, 0.0796581813966]
   sxyz[12,:] = [0.896431793354, -0.0891183197015, -0.112276291166]
   sxyz[13,:] = [0.89502463615, -0.117342129771, 0.0800751092973]
   sxyz[14,:] = [0.895387340606, -0.0991539019425, 0.1104117382]
   sxyz[15,:] = [0.896406683527, -0.07463850092, 0.125000021066]
   sxyz[16,:] = [0.89268627592, 0.00699924890983, 0.188190492228]
   sxyz[17,:] = [0.896393314003, 0.0745250726884, 0.150280066432]
   sxyz[18,:] = [0.896406683527, 0.07443850092, 0.125000021066]
   sxyz[19,:] = [0.895262016363, 0.0960322575544, 0.112435838349]
   sxyz[20,:] = [0.896974221307, 0.115672851276, 0.085900314048]
   sxyz[21,:] = [0.896496124549, 0.092119811179, -0.110966736786]
   return 0

def UX_get_sensor_box(uxp,sdb,grid):
   print "PYTHON: getting sensor box."
   nsensor = grid['nsensor']
   sbox = grid['box']
   if nsensor == 0:
      return 0
   for i in xrange(1,nsensor+1):
      sbox[i,0] = 5.0e-3
      sbox[i,1] = 5.0e-3
      sbox[i,2] = 5.0e-3
   return 0
```

APPENDIX C

LISTING OF CODE TO GENERATE ROBIN BODY POINTS

```
/* Program to calculate surface coordinates of the ROBIN body and pylon
 * Code by Adam Cofer, September 2009
 * For the National Center for Computational Engineering (SimCenter)
 * University of Tennessee at Chattanooga
 */

#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <map>
#include <vector>
#include <math.h>

using namespace std;

const string body_coordinate_file = "C_Body.txt";
const string pylon_coordinate_file = "C_Pylon.txt";
const string body_dat_file = "ROBIN_Body.dat";
const string pylon_dat_file = "ROBIN_Pylon.dat";
const string body_plot3d_file = "ROBIN_Body.xyz";
const string pylon_plot3d_file = "ROBIN_Pylon.xyz";

double analytic_function(double C[8], double x);
void get_xyz(double H, double W, double Z0, double N, double phi, double * x,
double * y, double * z);
void read_coefficient_block(double CH[], double CW[], double CZ0[], double CN[],
ifstream& Cfile);
void write_database_files(string infile, string datfile, string p3dfile,
                          int I, int J, double L, double PHI,
                          double a, double xi0,
                          double x0, double phi0, vector<double>& newblock_x);

double analytic_function(double C[8], double x){
  double retval;

  retval = C[0] + C[1] * pow(fabs((x + C[2])/C[3]),C[4]);
  if(C[7] != 0.0 && C[7] != 1.0){
    retval = C[5] + C[6] * pow(fabs(retval),1.0/C[7]);
  }

  return retval;
}

void get_xyz(double H, double W, double Z0, double N, double phi, double * x,
double * y, double * z){
  double sinphi,cosphi;
  double num,den,r;

  sinphi = -sin(phi);
  cosphi = cos(phi);
  num = (H*W)/2.0;
  den = pow((pow(fabs(H * sinphi),N) + pow(fabs(W * cosphi),N)),1.0/N);
  r = num/den;
  *y = r * cosphi;
  *z = r * sinphi + Z0;

}

void read_coefficient_block(double CH[], double CW[], double CZ0[], double CN[], 
ifstream& Cfile){
  int i,j;
  string line;
  stringstream ss;
  double Ctemp;
  string coeff_line;
  map<string, char> switchmap;
  switchmap["H"] = 'H';
  switchmap["W"] = 'W';
  switchmap["Z0"] = 'Z';
  switchmap["N"] = 'N';

  getline(Cfile, line);
  getline(Cfile, line);
  for(i=0; i < 4; i++){
    getline(Cfile, line, ' ');
    coeff_line = line;
    for(j=0; j < 8; j++){
      getline(Cfile, line, ' ');
      ss.clear();
      ss << line;
      ss >> Ctemp;
      switch(switchmap[coeff_line]){
        case 'H':
          CH[j] = Ctemp;
          break;
        case 'W':
          CW[j] = Ctemp;
          break;
        case 'Z':
          CZ0[j] = Ctemp;
          break;
        case 'N':
          CN[j] = Ctemp;
          break;
        default:
          cout << "Error, non matching coefficient line key '" << coeff_line << "'" 
<< endl;
          exit(1);
          break;
      }
    }
    getline(Cfile, line);
  }

  return;
}
```

```
void write_database_files(string infile, string datfile, string p3dfile,
                          int I, int J, double L, double PHI,
                          double a, double xi0,
                          double x0, double phi0, vector<double>& newblock_x){

  double this_x;
  double last_this_x = 0.0;
  double this_dx;
  double xi;
  double xin;
  double **x,**y,**z; // coordinates
  double phi;   // angle in super-ellipse
  double H,W,Z0,N;
  double CH[8];
  double CW[8];
  double CZ0[8];
  double CN[8];
  int i,j;
  double dx, dphi;

  dx = L/((double)I - 1.0);
  dphi = (PHI - phi0)/((double)J - 1.0);

  try{
    x = new double*[I];
    y = new double*[I];
    z = new double*[I];
    for(i=0; i<I; i++){
      x[i] = new double[J];
      y[i] = new double[J];
      z[i] = new double[J];
    }
  } catch(...) {
    cout << "Unable to allocate memory." << endl;
    exit(1);
  }


  ifstream Cfile;
  try{
    Cfile.open(infile.c_str());
  } catch(...) {
    cout << "Unable to open input file " << infile << " for coefficient input." <<
endl;
    exit(1);
  }

  read_coefficient_block(CH, CW, CZ0, CN, Cfile);
  for(j=0; j<J; j++){
    x[0][j] = x0;
    y[0][j] = 0.0;
    Z0 = analytic_function(CZ0, x[0][j]);
    z[0][j] = Z0;
  }
  xin = tanh(a * xi0)/(tanh(a*xi0) + tanh(a*(1.0-xi0)));
  for(i=1; i < I; i++){
    // two-sided hyperbolic tangent clustering to pack points near nose and tail
    xi = (dx * (double)i)/L;
    this_x = xin * ( 1.0 + (tanh(a*(xi - xi0))/tanh(a*xi0)) );
    this_x *= L;
    this_x += x0;
    this_dx = this_x - last_this_x;

    for(j=0; j < newblock_x.size(); j++){
      if(this_x >= newblock_x[j] && this_x < (newblock_x[j] + this_dx)){ // if this
x starts a new block, get new coefficients
        read_coefficient_block(CH, CW, CZ0, CN, Cfile);
      }
    }
    last_this_x = this_x;
    H = analytic_function(CH, this_x);
    W = analytic_function(CW, this_x);
    Z0 = analytic_function(CZ0, this_x);
    N = analytic_function(CN, this_x);
    for(j=0; j < J; j++){
      x[i][j] = this_x;
      phi = dphi * (double)j + phi0;
      get_xyz(H, W, Z0, N, phi, &x[i][j], &y[i][j], &z[i][j]);
    }
  }
  Cfile.close();

  ofstream outfile;
  try{
    outfile.open(p3dfile.c_str());
  } catch(...) {
    cout << "Unable to open output file." << endl;
    exit(1);
  }
  outfile << 1 << endl; // one block
  outfile << I << " " << J << " " << 1 << endl; // output block sizes
  outfile << setprecision(16);
  int linebreak_counter;
  linebreak_counter = 0;
  for(j=0; j<J; j++){
    for(i=0; i<I; i++){
      outfile << x[i][j] << " ";
      linebreak_counter++;
      if(linebreak_counter % 10 == 0 && linebreak_counter != 0){
        outfile << endl;
      }
    }
  }
  linebreak_counter = 0;
  for(j=0; j<J; j++){
    for(i=0; i<I; i++){
      outfile << y[i][j] << " ";
      linebreak_counter++;
      if(linebreak_counter % 10 == 0 && linebreak_counter != 0){
        outfile << endl;
      }
    }
  }
  linebreak_counter = 0;
  for(j=0; j<J; j++){
    for(i=0; i<I; i++){
      outfile << z[i][j] << " ";
      linebreak_counter++;
      if(linebreak_counter % 10 == 0 && linebreak_counter != 0){
        outfile << endl;
      }
    }
  }

  outfile.close();

  try{
    outfile.open(datfile.c_str());
  }catch(...){
    cout << "Unable to open output file." << endl;
    exit(1);
  }
  outfile << I * J << endl;
  for(i=0; i<I; i++){
    for(j=0; j<J; j++){
      outfile << x[i][j] << " " << y[i][j] << " " << z[i][j] << endl;
    }
  }
  outfile << endl;
  outfile.close();

  for(i=0; i<I; i++){
    delete[] x[i];
    delete[] y[i];
    delete[] z[i];
  }
  delete[] x;
  delete[] y;
  delete[] z;

}

int main (int argc, char * const argv[]) {
  double L; // maximum x-value
  double PHI; // maximum phi, angular coordinate around x axis in super-ellipse
equations
  int I,J; // number of divisions along x axis, polar axis
  double a,xi0; // parameters for hyperbolic tangent clustering of x points - a:
coefficient of clustering ratio, xi0: center point
  double x0, phi0; // starting x and polar coordinate for geometry
  vector<double> newblock_x; // vector of starting x coordinates for each new block
of coefficients

  I = 1000;
  J = 1000;
  x0 = 0.0;
  L = 2.0 - x0;
  phi0 = 0.0;
  PHI = M_PI;
  a = 10.0;
  xi0 = 0.5;
  newblock_x.clear();
  newblock_x.push_back(0.4);
  newblock_x.push_back(0.8);
  newblock_x.push_back(1.9);

  write_database_files(body_coordinate_file, body_dat_file, body_plot3d_file, I, J,
L, PHI, a, xi0, x0, phi0, newblock_x);

  I = 500;
  J = 1000;
  x0 = 0.4;
  L = 1.018 - x0;
  phi0 = 0.0; //-M_PI*1.1/2.0;
  PHI = M_PI; //M_PI*1.1/2.0;
  a = 10.0;
  xi0 = (L/2.0);
  newblock_x.clear();
  newblock_x.push_back(0.8);

  write_database_files(pylon_coordinate_file, pylon_dat_file, pylon_plot3d_file, I,
J, L, PHI, a, xi0, x0, phi0, newblock_x);

  return 0;
}
```

VITA

Adam Cofer was born in Oak Ridge, Tennessee on January 16, 1980. Most of his primary education was received in the public Oak Ridge school system. Following this, Adam attended the University of Tennessee at Chattanooga (UTC), graduating as a Brock Scholar with Departmental Honors in Computer Science (Scientific Applications) in May of 2003. Adam spent five years working as a Software Developer/Analyst for CH2M HILL, Inc. (now Critigen) before joining UTC's SimCenter: National Center for Computational Engineering in August of 2008. Adam is now a Computational Engineer for SimCenter Enterprises, Inc.