ABSTRACT

Title of Dissertation: On Overset Grids Connectivity and Vortex Tracking in Rotorcraft CFD

YikLoon Lee, Doctor of Philosophy, 2008

Dissertation directed by: Associate Professor James D. Baeder
Department of Aerospace Engineering

Two new numerical techniques that are useful in the simulation of high fidelity, vortical interactional aerodynamics for rotorcraft are developed. The first is a simple, general and radically different approach to the problem of overset grids connectivity in the numerical solutions of partial differential equations with arbitrary geometry. The second is an approach for addressing one of the most pressing issues in rotorcraft aerodynamics, the simulation of vortices and their interactions with the rotorcraft. It is a technique to automatically capture and track the vortices for subsequent interactions using very high-resolution helical grids. While the underlying problem in the former is fundamental and of general interest to computational science, the latter is specific to problems in rotorcraft aerodynamics, and is still in its infancy.

The overset structured grids connectivity developed in this thesis is based on a cell selection process instead of the traditional explicit hole cutting as an essential ingredient. Hole cutting is a byproduct of this process and the holes that arise are both automatically optimum and not affected by any imperfection on the wall surface. It can be regarded as a generalized method that can cut holes around both flow refinement and bodies indiscriminately. The greatest strength of this approach is the simplicity of its concept and the implementation, which results in a very elegant and compact algorithm.

The heart of the vortex tracking methodology is a set of long helical grids with high grid density at the central core where each vortex trajectory resides at the completion of the tracking process. Only the quasi-steady state problem is studied here. The technique is successfully tested on a half-span Quad Tilt Rotor in high speed forward flight. Several technical problems related to quasi-steady vortex tracking under certain flight conditions are also presented.

The two new techniques are also applied to the problems of 2D blade vortex interaction (BVI) and co-rotating vortices with encouraging results. Coupled with monotone cubic interpolation for the connectivity, these techniques used in 2D BVI are able to track the distorted vortex long after the interaction. These vortex tracking problems in rotorcraft CFD are difficult to handle without the use of IHC.

On Overset Grids Connectivity and Vortex Tracking in Rotorcraft CFD

by

YikLoon Lee

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Prof. James D. Baeder who, first of all, was willing to take the risk and accept a student with a background from an unrelated engineering discipline. His guidance, suggestion, patience and teaching throughout the course of the research work is greatly appreciated.

I would also like to thank my wife Molly Chen for her understanding and sacrifice throughout my graduate study. It was also fun explaining and discussing some of the more fundamental ideas in the thesis to someone completely outside the field and still getting interesting feedback.

Appreciation is also extended to my branch manager Dr. David Findlay and supervisor Mr. Mark Silva at Naval Air Warfare Center (NAWC) for their patience and support for the later part of this work.

I want to thank all my committee members for constructive criticism of this work, which is now more complete because of it.

Lastly, I benefitted greatly from discussion with people working with me, each of whom has a unique set of skills that I can sometimes get answers from. They include Drs. Jayanarayanan Sitaraman, Vinit Gupta, Karthik Duraisamy and our local LaTeX guru at NAWC Mr. Ryan Czerwiec.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

$minmod(x, y)$	median value of x, y and 0
$s_{i+\frac{1}{2}}$	linear slope between f_i and f_{i+1}
s_i	minmod($s_{i-\frac{1}{2}}$,$s_{i+\frac{1}{2}}$)
d_i	second difference at i, =($s_{i+\frac{1}{2}} - s_{i-\frac{1}{2}}$)/ ($x_{i+1} - x_{i-1}$)
$d_{i+\frac{1}{2}}$	minmod(d_i,d_{i+1})
$p_{i+\frac{1}{2}}(x)$	median of quadratic and linear curve between i and $i + 1$
s, t	linear slope to the left and right of i
$g(r)$	=$f_i'(s/t)$
ξ, η	curvilinear coordinates
Δx,Δs	uniform grid spacing
a_∞	free-stream speed of sound
c_{tip}	tip chord length
C^n	smoothness of function to the n^{th} derivative
C_l	sectional lift coefficient
C_m	sectional moment coefficient
C_p	pressure coefficient
M_∞	free-stream Mach number
r	radial position (0 to 1)

R	rotor radius
r_c	vortex core radius
x/c	non-dimensional blade chord coordinate
x_v	vortex position
y_v	miss-distance of vortex from blade
Γ	vortex circulation strength
μ	advance ratio ($V_\infty/\Omega R$)
ψ	azimuthal coordinate

Abbreviations

BVI	Blade Vortex Interaction
CAF	Co-Array Fortran
CFD	Computational Fluid Cynamics
IHC	Implicit Hole Cutting
VT	Vortex Tracking

Chapter 1

Introduction

The purpose of this thesis is to accurately model vortical flowfields of rotorcraft numerically for very large distances. Two new numerical techniques are developed to help achieve this. The first is a simple approach to the problem of overset grids connectivity for the numerical simulation of partial differential equations (PDE) with complex geometry and/or relative grid motion, and the second is a methodology for high resolution and automated tracking of helical vortices in rotorcraft computational fluid dynamics (CFD) using the above connectivity approach.

1.1 Physical Understanding

A large number of phenomena in science and engineering are accurately described by PDEs. Almost all of those that are of importance to scientists and engineers are either highly nonlinear, have complex geometrical boundary, or contain components with relative motion. These attributes necessitate the use of numerical methods on high performance computers to obtain any reasonable solution. This, in turn, requires the construction of a grid in the solution domain that serves to approximate the space continuum. The design of the grid (in particular the grid density) generally requires some prior knowledge of the solution before the PDE could be solved. In this respect, several general classes of grids have been developed for this purpose, e.g., structured, unstructured, adaptive Cartesian, etc. This work employs overset Chimera) structured grids exclusively for all numerical studies.

There exists a problem in numerical PDEs that is of fundamental and intense interest to the rotorcraft CFD community. This is the high resolution capturing of very long helically rotating blade tip vortices from first principles (i.e., using only conservation of mass, momentum and energy), and the simulation of the subsequent interaction with other blades, aircraft components and themselves. The two pictures in Fig. 1.1 show these helical vortices from propellers and a helicopter rotor made visible by the effect of natural condensation of water vapor in the atmosphere inside the low-pressure region within the vortex core. The importance of an accurate solution to this problem in a reasonable time frame to engineering and science is well documented.

The significance of the above problem to rotorcraft aerodynamics stems from the strong effect of the blade tip vortices on the performance and behavior of the rotor and airframe. Each vortex is generated at the tip of a blade due to the 'leaking' of air from high- (bottom of blade) to low-pressure (top of blade) region, setting it into circular motion. It is structurally similar to a miniature tornado or hurricane. Its core diameter is generally a few inches across and has a peak swirl velocity of about a couple hundred miles per hour. Besides causing vortical interactions, which induce noise, vibration, fatigue, etc., these energetic vortices represent a loss of useful energy at the expense of more fuel burn.

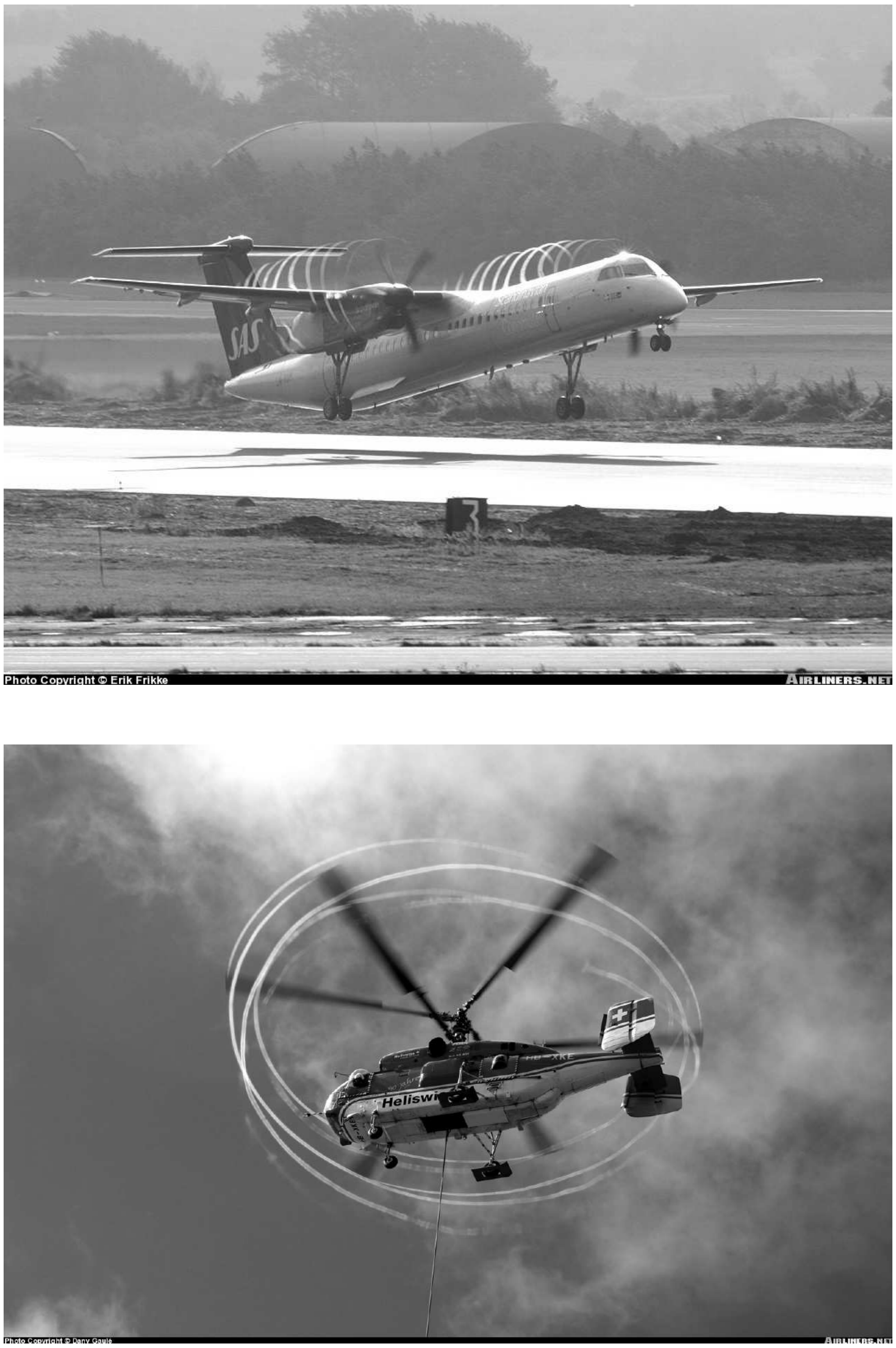


Figure 1.1: Helical vortices generated by propellers and a helicopter. (Courtesy of Erik Frikke (above) and Dany Gaule (below)

The outboard portion of a rotorcraft blade is where most of the aerodynamic lift is generated. Unfortunately the region in the vicinity is also where the flowfield is most intense and complicated. The flow regime encountered here is from high subsonic to transonic with the possible appearance of shock waves. A large velocity and pressure gradient exists in the flow field around the tip vortex. The solution at the vortex core has the form of a steep inverted spike for the pressure or density field and a peak/valley pair for the swirl velocity. This flow feature is maintained for a long distance as the vortex spirals downward (and backward in a forward flight). The mechanism for the generation of a tip vortex and its numerical investigation is studied in detail in Duraisamy [1] and will not be discussed here.

In light of the above, it is easily understandable why the prediction of the strength, size and trajectories of these helical vortices and the simulation of their interactions with anything in their path is of utmost importance for the design and analysis of a high-performance rotorcraft.

1.2 Motivation and Objective

This well-known aerodynamic problem has remained, to this day, generally unsolved despite years of intensive studies employing different methods and levels of approximation. It has long plagued the rotorcraft industry and research community alike. The numerical solution in the region surrounding the vortex can not capture the fundamental vortex feature described above for an extended period of time. The computational difficulty is essentially because of a large change in the solution within a small confined region. The solution gradient calculated using finite differences is always underestimated near the peaks or troughs. The situation is compounded by the insufficient grid resolution to resolve the sharp features. This leads to a gradual and steady ‘lowering’ of the peaks as the vortex convects downstream. Eventually the vortex is weakened to oblivion by energy dissipation through the action of viscosity.

The ability to numerically preserve the vortex structure for a long period of time is indispensable for the simulation of the aerodynamic interactions between the vortices and other aircraft components that are so crucial to the accurate prediction of a rotorcraft's behavior including the ”Brownout” phenomenon.

Two avenues are generally available to ameliorate the problem: 1) use of higher accuracy numerical methods and, 2) grid refinement. Although high-order accurate schemes can help alleviate the problem, it has limited potential by itself. Grid resolution is ultimately the most important issue. Grid refinement is, however, computationally very expensive. Various types of grids have been used by previous investigators, ranging from adaptive unstructured to overset structured to a combination of these. In recent years, overset structured grids seem to be making progress in tackling this problem.

While a few researchers suggest that at least some new mathematical ideas similar to that for the proper treatment of shocks is required, the results from this work reinforces the belief that all or most of the ingredients for this problem are already in place. The hurdle lies in the implementation of a seemingly simple vortex tracking methodology, which is depicted in Fig. 1.2 where specially constructed high core-resolution grids are used in an attempt to capture the tip vortices.

One of the objectives of this thesis is to compute, with unprecedented accu-

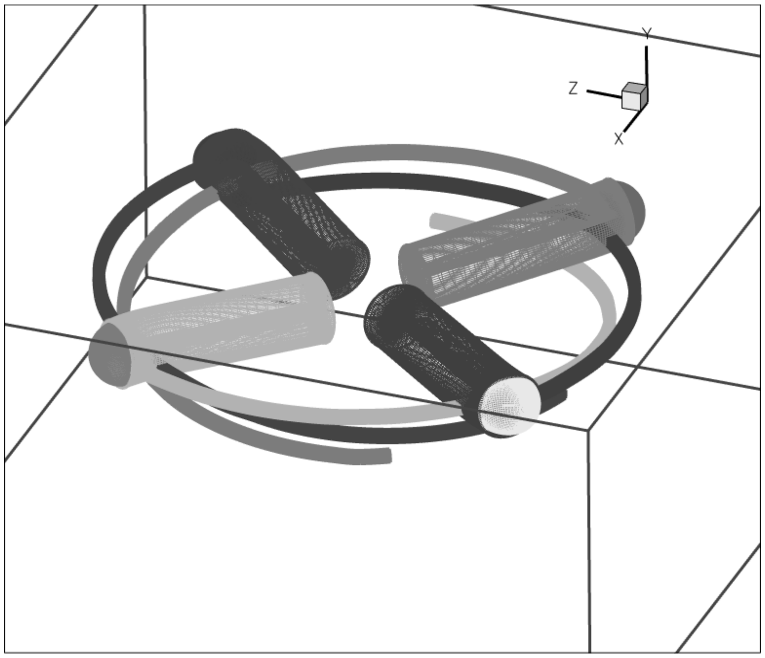


Figure 1.2: Helical vortex-tracking grids.

racy, economy, and from first principles, the properties of the vortices and their interactions using overset vortex tracking grids. The two pictures in Fig. 1.1 clearly depict the visualization of the desired flow solution.

1.3 Previous Work

1.3.1 Overset Grids Connectivity

The first paper on the methodology of overset grids most widely cited appeared in 1983 was published by Steger et al. [2] at NASA, for computations in aerodynamics, although the general idea most likely arose many years before. It has since been applied to different fields outside of aerospace such as biological flow, hydrodynamics, geodynamics [3], droplet dynamics [4], oceanic flow [5], etc. The

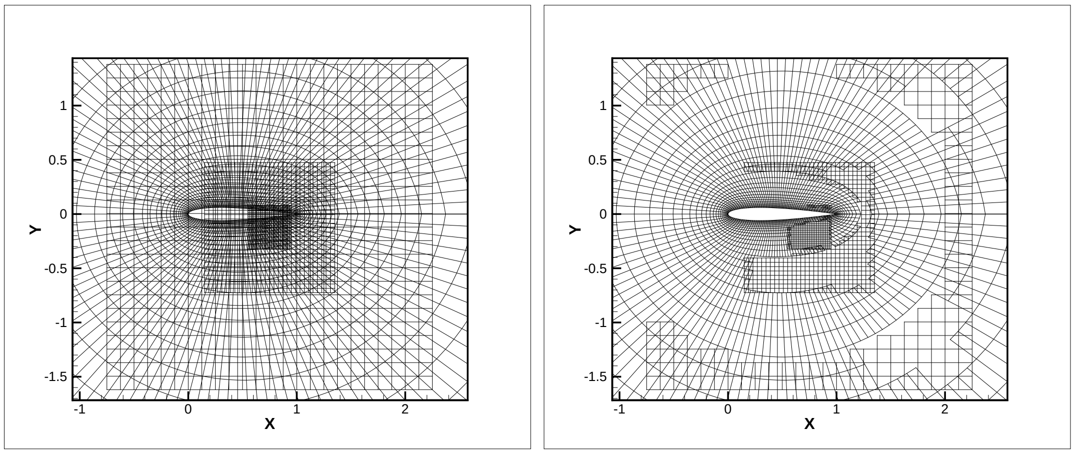


Figure 1.3: Overset connectivity in moving grids.

usage of overset connectivity is particularly important in problems with moving grids, such as the simple unsteady 2D airfoil vortex interaction in Fig. 1.3

There are likely many in house or commercial codes that have some limited capability in establishing overset grids connectivity to various degree of sophistication and geometry complexity. In this section, only those existing algorithms and codes that are more general in their ability to handle geometry with some degree of arbitrariness (at least in intention), have dedicated publications on the method employed and have been reasonably widely used and tested will be briefly reviewed. In recent years, existing connectivity algorithms are gradually maturing to the point at which further improvements are likely to be only incremental. The complexity of the algorithm is probably the last remaining area that needs significant improvement.

The most widely used overset grids connectivity code is probably PEGASUS [6], especially Version 4. The amount of user input required in this version is rather large. The latest Version 5 has been completely rewritten from scratch, with the main objective of automation and lower expertise requirement for the user. It has evolved from the first connectivity code in the early eighties. PEGASUS has always been a stand-alone code and not specifically associated with any solver, but is extensively used with OVERFLOW. It has been used to study some very complex geometry such as the Boeing 777 in a landing configuration. Despite the latest effort, however, it is still relatively slow and requires a lot of user input. It is thus not suitable for problems with moving grids that require connectivity recalculation every time step. PEGASUS employs the hole map method (as explained in Chapter 3) as its hole cutting technique. This method is very memory intensive especially for 3D problems.

DCF3D [7] is another NASA code that is primarily associated with OverFlow-D although it can also be stand-alone. It was designed from the beginning to be used for moving grids, which is why it has to be fast. Like PEGASUS, DCF3D is also widely supported. The older version uses basic analytic shapes to cut holes. The latest version, however, uses an innovative combination of hole map and ray casting techniques, called object X-ray, to significantly improve their weaknesses and combine their strengths. This technique is likely the most advanced among existing methods.

BEGGAR [8] is an Air Force code developed at Eglin Air Force Base specifically for store separation problems, although theoretically it can also solve other external aerodynamics problems. These are inherently unsteady problems with moving grids. This code is somewhat slower than DCF3D and has a limited user base because it was designed from the beginning for store separation. It is also a monolithic code and is integrated with a flow solver. For hole cutting, it employs the 'mark and fill' technique which, in different versions, is used in several other codes as well.

Another code that is mostly used for non-aerospace applications is Overture [9], which was developed at Los Alamos and later at CASC of Lawrence Livermore. It was evolved from an earlier work by Chesshire and Henshaw [10], which was published in 1990. It has been used for problems like internal combustion engines, trucks, submarines, some fundamental flow problems, etc. Overture, unlike other codes, is written in C++ using object oriented methodology. It is also capable of solving moving body problems. It is, however, not very widely distributed. The hole cutting technique used is a variant of the 'mark and fill' for BEGGAR.

While the four codes mentioned above are Government supported, FASTRAN [11] is a commercial software developed at CFDRC. Its hole cutting technique is yet another variant of 'mark and fill'. Although it was used in [11] for a 2D store separation problem, the example showed the code is barely fast enough for a fast moving body problem. Its suggested remedy was a recalculation every few time steps. This is surely not acceptable for a rotorcraft problem.

Xcog (2D) [12] and Chalmesh (3D) [13] were developed in Sweden in an academic environment as a stand-alone code. They have been used in hydrodynamics for propeller and ship design. The connectivity problem analysis seems to be given careful consideration. Moving body problems, however, were not demonstrated. Again, holes are cut with a variant of a two-step 'mark and fill' technique.

DiRTLib [14] and SUGGAR [15] are more general codes that can handle unstructured grids with any generalized cells. It also differs from the other codes in that it uses wall distance to grid points as part of the hole cutting tool. Distances, however, are quite costly to calculate especially because they may not always be required by the solver. This code is not yet widely distributed because it is relatively new. It is nevertheless quickly gaining acceptance.

Some of the main features of these codes at this writing are summarized in Table 1.1. The general logic of the algorithm behind each of the above existing codes is similar although their implementations in practice are very different. It primarily concerns the identification of those grid points that are inside multiple bodies of arbitrary shape and size. An algorithm of this nature is inherently very complex and error prone. The new approach given in this thesis completely does away with this concept. Another problem with the existing approach, besides its significant complexity, is that those points that are inside refinement grids (such as vortex grids in this work) need to be identified as well. This latter group of points are either ignored or treated separately. This problem is solved seamlessly in the approach given in this thesis with no distinction between the two groups.

1.3.2 Rotorcraft Vortical Flow

True high-fidelity rotorcraft vortex capturing and tracking to a large wake age and the subsequent blade or airframe-vortex interaction in CFD using Euler/Navier–Stokes equations alone (without external wake models, vorticity confinement or excessive expense) is still evading researchers. Numerous studies and attempts have been made to achieve, at least partially, this objective in the past two decades. Most of these are first targeted at the hovering rotor, because it is the most fundamental of all rotorcraft vortex interaction problems. It is also the only one that can take advantage of the axisymmetric nature of the flow, and thus tremendously reduces the problem size of a multi-bladed rotor to a single blade with a periodic boundary condition. This smaller problem size allowed

Code name	Organization	Method	Comments	Widely used?
PEGASUS 5	NASA original, N. Suhs	Hole map	slow, not for moving grids, tested with large problems	yes
DCF3D	NASA Ames, R. Meakin	Object X-ray	Fast, widely used with OverFlow-D, significant user-input	yes
BEGGAR	Air Force Eglin R. Maple, D. Belk	Direct cut	large code, for store separation, reported unreliability	no
OVERTURE	Lawrence Liver. W. Henshaw	Direct cut	C++ for non-aerospace application	no
DirtLib/ SUGGAR	U. of Alabama R. Noack	Octree	multi-resol., structured /unstruct. grids, serial	yes
FASTRAN	CFDRC, Z. J. Wang	Direct cut	not widely demonstrated	no?
ChalMesh/ Xcog	Chalmers U. of Tech., Sweden, A. Peterson	Direct cut	some similarity with OverTure	no

Table 1.1: Some main features of existing connectivity codes at this writing.

researchers to study the problem of vortex capturing and diffusion with the high performance computers available at the time. The first such study using the inviscid Euler equations was published by Roberts and Murman [16] in 1985, while Wake and Sankar [17] provided the first results using the Navier−Stokes equations in 1989.

After years of improvement to the underlying numerical technique, Wake and Baeder [18] presented the then most complete results and comparison with experimental data for such a reduced hovering rotor problem in 1996 using the TURNS code. This code was first published in Srinivasan and Baeder [19]. More recently in 2001, Strawn and Djomehri [20] provided a very detailed study (and comparison with experimental data) of this well-investigated problem for UH-60 rotor concentrating their efforts on, among others, the grid-independence issue. In addition, Potsdam and Strawn [21] provides a good reference on the half-span and whole-aircraft V-22 tilt rotor hover problem in which the download and aerodynamic interaction with the wings were considered and extensively investigated. However, only the usual force-related quantities were presented without any of the vortex related quantities. Both stationary and rotating isolated rotor were also studied. These last two works were conducted using NASA's overset grids Navier−Stokes flow solver OverFlow-D.

A very detailed study of both fixed and rotary wing tip vortex formation and evolution was presented by Duraisamy in [1] using a modified version of the TURNS code. One- and two-bladed hover cases were modeled using a single blade with a non-nested vortex grid and a highly refined background grid respectively. For the two-bladed case, the tip vortex was tracked up to two revolutions. Extensive comparison with experimental data was conducted which was generally favorable. This study was also extended to include flow control with blowing at the blade tip.

Recently, rotor hover calculations have also been conducted outside the United States. Kang and Kwon [22] at KAIST used solution adaptive mesh refinement in unstructured grids to study the Caradonna and Tung rotor. In Europe, a hover study of the BO-105 and model 7A rotors using two numerical codes (FLOWer and CANARI) and including aeroelastic effects was published in Beaumier et al. [23]. It also accounts for a laminar-turbulent boundary layer instead of a fully turbulent one. This study was performed in ONERA and DLR using a more conventional single-grid method. Another paper on model 7A rotor hover problem was written by Benoit and Jeanfaivre [24] using overset grids similar to [20] in the framework of the European CHANCE (Complete Helicopter Advanced Numerical Computational Environment) project. Unlike the other studies mentioned above that use symmetry condition, all the blades in the rotor were modeled here. The intention of this study was the investigation of meshing strategies for overset grids and thus no experimental data were shown. Another European project with a much wider collaboration known as EROS [25] was initiated with the objective of a common European Euler code for rotorcraft aerodynamics.

Another fundamental problem case in the study of rotorcraft vortex interaction is the unsteady 2D blade vortex interaction (BVI). Rai [26] published one of the first results for this problem. While the emphasis in the latter half of this thesis is on general 3D vortex tracking and interaction, this 2D BVI problem provided a good initial testing ground for the general 3D case.

Many researchers have investigated this problem in the years since. Recently it was studied using adaptive unstructured grid with 2nd-order accuracy by

Oh, Kim and Kwon [27], [28]. Results compared well with experiment reported by Lee and Bershader in [29]. High-resolution results of both the vortex and the acoustic wave were obtained using adaptive cells, but at a cost of 32 cells across the vortex core. This cost, which is mostly from the unstructured nature, might be unbearably high for some complicated 3D problems using present-day computers. This is compared with only four cells used in the present work, and Tang [48], while having similar rate of vortex diffusion.

Vortex tracking grids are very simple and logical and certainly not entirely new, though not very common either. They have been used by a group of researchers [30] to track non-rotating vortices for fixed wing aircraft. In this paper by Hariharan and Sankar [30], only two overset grids were used – one wing grid and another stretched vortex grid (not nested). The latter was placed close to the actual vortex trajectory. This ensures that the grid is not too skewed relative to the vortex after adaptation (because the grid stations are not rotated) to ensure orthogonality. A captured vortex distance of about 50 chords was achieved using a 7th-order ENO scheme.

Spall [31] also used the Euler equations with the more common 2nd-order scheme to numerically study wing tip vortex preservation. However, grid clustering around the vortex was achieved by using patched grids instead of the more versatile overset grids. A convection distance of at least 13 chords was achieved.

Helical vortex grids were also used by Egolf et al. [32] to capture vortices from a 4-bladed rotor. Each of the non-nested grids extends about 80° from the tip trailing edge, ending just before the next blade. Thus no blade vortex interaction was simulated. Grid points concentrated at the center of the boundaries of the stretched grids were fanned out to reduce cell aspect ratio there, and so increase grid quality. But cells at the corners were significantly skewed as a result. Nevertheless encouraging results were obtained for the short length of vortices.

One adaptation technique in overset structured grids that can be used to track flow features [33] is the Adaptive Spatial Partitioning and Refinement method developed in Overflow-D. In this method, successive levels of off-body Cartesian grids are generated, adapted, and refined around flow features, based on solution error estimation. This technique has all the advantages associated with simple Cartesian grids. The grids, however, are not necessarily aligned with the dominant direction of the flow features, and thus can be relatively skewed with respect to the direction with the largest solution gradient. This method has been successfully applied to many practical problems containing shocks. Overflow-D was also used in [20] to capture a hovering rotor wake. The above-mentioned adaptation technique, however, was not utilized.

Vortex methods are another class of numerical techniques for rotor wake analysis [34] [35] [36] [37] [38] [39] [40]. They are relatively inexpensive compared to most CFD methods and have proven to be very useful in predicting vortex positions, vortex roll-up, load distributions over the blades, and rotor performance. There is no artificial diffusion and the governing vorticity transport equation is solved using finite-difference approximations to the space and time derivatives. Some empiricism is assumed in the initial strength, core size, release location and diffusion rate of the vortex. The induced velocity field can be reconstructed from the application of the Biot-Savart law once the position and strength of the vorticity is computed at each time step. These free vortex methods have proven to be particularly useful for a broad range of practical problems in rotorcraft aerodynamics, aeroelasticity, and acoustics, and are at the heart of nearly all comprehensive forms of helicopter analyses.

The aircraft platform used for the development of vortex tracking method in this work is the Quad tilt Rotor (QTR). Some of its background information and history can be found in Snyder [41], Corrigan et al. [42] and in *Vertiflite* [43]. Several authors have conducted numerical studies of the QTR aerodynamics. Most of these concentrate on the fountain flow phenomenon in hover. The numerical flow field and general behavior of this aircraft was extensively investigated in [44]. Some of the problems studied are download in hover and low speed forward flight. The rotors in [44] are modeled as an actuator disk. Sitaraman and Baeder [45] investigated the fore/aft rotor interaction using coupled CFD/free wake analysis. Further references on QTR can be found in these two references.

Despite numerous studies of rotorcraft vortical flow using different techniques applied to Euler/Navier−Stokes equations as outlined in this section, vortex preservation and the simulation of their interactions remain a very challenging task. The vortex tracking technique developed in this thesis provides the latest tool for an attempt to significantly improve upon existing methods. It automatically tracks the location of the vortices and accordingly positions the helical grids that are both locally high-resolution and economical.

1.4 Outline of Thesis

The work reported in Chapters 2, 3 and 4 is geared towards the eventual objective of developing a general vortex tracking methodology. The new concept underlying the overset grids connectivity method as given in Chapter 3, however, is generally applicable to any problem in numerical PDEs with an arbitrary geometry.

Chapter 2 concentrates on two topics of relevance to overset grids, namely interpolation and data communication. These two issues directly affect the accuracy and efficiency of the flow solver. Their underlying mathematics and technology are not new, and mostly developed somewhere else. Some of the viewpoints and details, however, seem to be new and unavailable elsewhere, e.g., exact coordinate transformation, the curse of dimensionality in high-order interpolation, one-sided vs. two-sided data communication for interpolants in overset grids, etc. This chapter concisely summarizes some of the important results relevant to the objective of this work.

In Chapter 3, the exposition of a new and surprisingly simple approach to the central problem of overset grids connectivity in numerical PDE with complex geometry and/or moving components is undertaken. This approach is completely general and is fundamentally different from existing methods that are publicly available. It has several prominent advantages over the existing technology. Of these, the simplicity of its concept is considered the most significant. This is especially true considering the logical complexity of existing codes and high level of experience and knowledge (depending on which code) expected of the users.

One manifestation of the extreme simplicity of this concept is the total transparency of this approach to the user, and the fact that the primary operation is the comparison of two numbers (the cell sizes). No additional information is required as input besides those needed by the solver. This is in stark contrast to existing codes.

It is not within the scope of this thesis to discuss at length the different implementation methods and strategies employed in existing codes. They would be briefly discussed as appropriate in serving to more clearly explain and differentiate the new approach outlined here.

In Chapter 4, attention is turned to the problem of high fidelity vortex tracking. The main theme here is the use of overset vortex tracking grids to capture the tip vortices of rotorcraft (QTR is considered as a test case) blades with very high precision and their interactions with other aircraft components. These are flexible, adaptive, long and slender nested helical grids that completely embed the vortex trajectory. They have very high resolution at the inner core, and yet contain only a small number of points.

Chapter 2

Computational Methodology

This chapter concentrates on two topics that are especially important to overset grids technology: interpolation and data communication in parallel computation. They address respectively the issues of accuracy and efficiency of the flow solver. For other numerical techniques such as unstructured grids, the former topic is mostly irrelevant while the latter is significant in a less complicated sense. Next section briefly reviews the solver employed here before embarking on these issues.

2.1 Flow Solver

The underlying flow equations in the solver used throughout this work are derived from the three conservation laws for mass, momentum and energy. This vector equation is generally expressed as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = 0 \tag{2.1}$$

where Q is the conservative variable vector $\{\rho, \rho\mathbf{V}, \epsilon\}$ of mass, momentum and energy density and F, G, H are the inviscid flux vectors, e.g., $F = \{\rho u, \rho u^2 + p, \rho v, \rho w, (\epsilon + p)u\}$. It is extensively discussed in the classic text of Hirsch [47].

The solver was developed in house at University of Maryland for the purpose of solving rotorcraft CFD problems although it is general enough for other compressible flow applications as well.

This code is generally third-order accurate in space, time as well as inter-grid interpolation. The vortex-preserving scheme for the spatial discretization is similar to those employed by Tang [48]. In this scheme, the 4^{th}-order difference scheme is employed for the first two derivatives of a 3^{rd}-order piecewise quadratic variable extrapolation. Without limiting, this results in a vortex diffusion of about 10% after a free convection distance of 10 chords for a Cartesian grid with four cells across the vortex core. For limiting, the BAP (biased averaging procedure) proposed in [49] is employed for the grids to suppress any possible oscillation around shocks at the blade tips. This limiter is not strictly TVD (total variation diminishing) and thus still produces some small oscillations. It might also be very computationally expensive if the biased averaging functions chosen are trigonometric, e.g., tangent or hyperbolic tangent.

The time integration scheme employs explicit 3-stage TVD Runge−Kutta (RK) scheme as given in Shu and Osher [50] for non-body-conforming grids. It consists of three stages of Euler forward difference and is a third-order RK method with the optimal CFL. It is a low storage scheme compared with some classical RK methods because it does not need to store any intermediate results. The first-order Euler implicit time integration scheme is used for all body-conforming grids.

2.2 Interpolation in Overset Grids

In the many diverse applications of computational science in which discrete data are common, such as solution reconstruction in CFD, solution communication between overset grids, experimental data interpolation (e.g., equation of state data), computational graphics/geometry etc., the following problem is frequently encountered, which has been stated in a simple one-dimensional form:

Given a set of discrete data $\{f_i\}$ that represent samples of some unknown function on the real line (equally spaced or not), estimate the set of slopes $\{f_i'\}$ at the same points subject to some conditions (monotonicity-preserving, convexity, order of accuracy etc.) on the final results.

Both $\{f_i\}$ and $\{f_i'\}$ are then used to perform subsequent computations. Note that without the stipulated monotonicity-preserving condition, the problem is easily solved with conventional finite difference or spline methods (in fact, the slope estimated from these methods is, in one type of interpolation as discussed later, the first of a two-step procedure). This is sufficient when the data are smooth. When the data gradient is large, as is often encountered in all the applications mentioned above, these simple slope estimates are no longer accurate and create spurious oscillations in the final results, whatever the application. Monotonicity-preserving conditions are then required to restore physical reality and sometimes prevent negative quantities from appearing. They can take the form of constraints on the slopes, slopes that are non-oscillatory a priori (i.e., limiter functions), increasing the degree of the interpolating polynomial, adding new mesh points etc.

The problem statement above is simple and clear. It is also such an important subject that this section is allocated to summarize some of the important results relevant to CFD of hyperbolic conservation laws and overset grids. The emphasis here is intentionally heavy on fundamentals and less on the seemingly countless number of methods for interpolation. Naturally, there are many different ways to estimate f_i', each producing different final results. This gives rise to a rich research area on a seemingly simple subject that has attracted some of the best minds in the field and resulted in many research papers over the years. It is also important to note that this subject involves just discrete mathematical analysis and contains no physics.

In the context of CFD using overset structured grids, two applications of this subject are of interest here: i) piecewise solution reconstruction within a grid cell, and ii) solution interpolation from another grid. While reconstruction, which is fundamentally an extrapolation, is one of two steps for flux evaluation in the flow solver, interpolation is used in overset method for solution communications among grids. Both determine the spatial order of accuracy of the computation and are required to yield results that are oscillation free for monotone but discontinuous data. A large literature exists for the reconstruction process ([51], [52], [53], [54]) and will not be further discussed here. This section will concentrate on high-order (cubic) interpolation of solutions between overset grids. This is still not very common among existing overset-grid CFD codes but is essential for certain problems that need interpolations from large gradient region (such as 2D BVI problems in the presence of a shock as discussed in Chapter 4). However, methods developed for interpolation can also be applied to reconstruction in conservation laws and vice versa. Practical differences between them will be mentioned at the end of this section after monotonicity constraint is discussed.

2.2.1 Preliminaries

A straightforward but essential definition of monotone data is needed for the following discussion and for the high-order extensions of the monotonicity constraints: $\{f_i\}$ are said to be locally monotone in the interval $[x_i, x_{i+1}]$ if the data are increasing or decreasing at the four points $i-1, i, i+1$ and $i+2$, i.e., if $f_{i-1} \le f_i \le f_{i+1} \le f_{i+2}$ or $f_{i-1} \ge f_i \ge f_{i+1} \ge f_{i+2}$.

By considering all possible relative positions of the four points, we can classify the data at these points, for the purpose of monotonicity analysis, into three types, shown in Fig. 2.1a, as i) locally monotone data, ii) local extremum, and iii) oscillating data which we assume here is either spurious or due to insufficient mesh resolution. This last type will not be further considered here. Type i) and ii) can be further divided into one with and without an inflection point as shown in Fig. 2.1b. Oscillatory data of type iii) always have an inflection point. The presence or absence of an inflection point is an important consideration in monotonicity constraint such as MP5 [58] and M3 (as detailed later).

For the purpose of discrete analysis, a general function can be said to consist of regions with 3 different features: monotone regions, extrema and discontinuities. A discontinuity, however, cannot be represented unambiguously using discrete data, i.e., it is impossible to classify a data set as having a discontinuity, but only as a data set with large gradient. Having noted that, we show in Fig. 2.2 that it takes at least five data points to be able to distinguish between an extremum and a possible discontinuity. The set of 3 data points with circles in both figures have the same values. Yet they can contain either a discontinuity or an extremum depending on the neighboring values. It is important to be able to detect the latter because the continuation property inherent in Taylor's

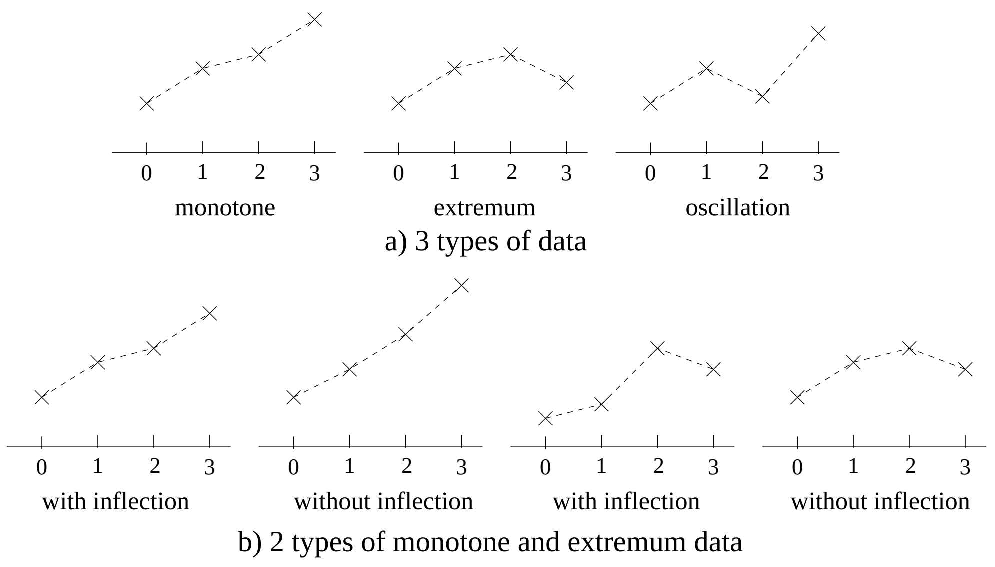


Figure 2.1: Data classification for monotonicity analysis purpose.

expansion, and thus finite differences, is broken across a discontinuity.

Two concepts are central in the discussion of monotonicity constraints for the estimated slope f_i': median and minmod. The median of three numbers, median(x, y, z), is the one that lies between the other two. The minmod of two numbers minmod(x, y) is the median of x, y and 0. Effectively, minmod returns the smaller of the two numbers in absolute term if both have the same sign and returns zero otherwise. For programming purposes, they can be expressed mathematically as

$$\begin{aligned} minmod(x,y) &= \frac{1}{2}[sgn(x) + sgn(y)]min(|x|,|y|) \\ median(x,y,z) &= x + minmod(y-x, z-x) \\ &= y + minmod(x-y, z-y) \end{aligned}$$

We make a few observations concerning the properties of a median. First, the median lies in the interval defined by any two of the three arguments. This is quite obvious. Second, the constraining of a number to lie within an interval

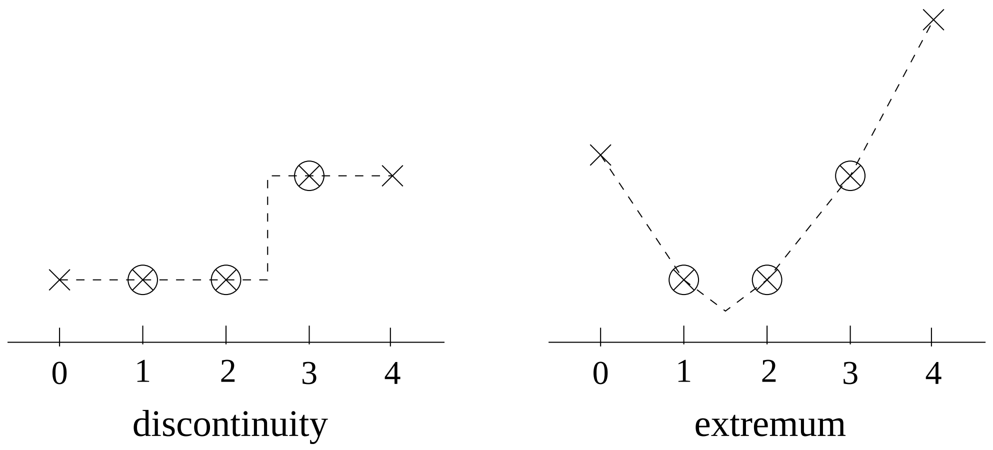


Figure 2.2: At least 5 points are required to distinguish between an extremum and a possible discontinuity.

can be enforced by using the median function. Third, if any two of the three arguments of a median, say x and y, are accurate to a certain order, then the median is also accurate to the same order because of the first property above and the fact that any value between x and y has the same order of accuracy as x and y. Minmod, on the other hand, arises naturally in monotonicity constraint to be discussed later.

It is also observed here that while minmod(x, y) is a stable function, maxmod (x, y), similar to minmod except for the change from $\min(|x|, |y|)$ to $\max(|x|, |y|)$, is unstable (like the popular ENO scheme [53] in CFD) since there is a large change of returned value when x approaches 0^- and y is positive fixed or when x approaches 0^+ and y is negative fixed. In practice, however, x and y usually represent two neighboring slopes, and a discontinuity is likely the reason when this happens. A limiter or a constraint, if present, then takes effect to prevent a large change and returns a stable value. 'Superbee' limiter is such an example:

$$final\ slope = minmod(maxmod(x, y), 2minmod(x, y))$$

The final slope is stabilized by the term 2minmod(x, y).

2.2.2 Piecewise Cubic Hermite Interpolation

For interpolation at a point between i and $i+1$, the simplest and most natural interpolant that utilizes both f_i and f_i' is the piecewise cubic Hermite interpolation, which can be written as a polynomial

$$f(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 \tag{2.2}$$

where

$$\begin{aligned}
0 < x < \Delta x (= x_{i+1} - x_i) \\
c_n = c_n(f_i, f_i', f_{i+1}, f_{i+1}') \\
c_0 = f_i,\ c_2 = \frac{3s_{i+\frac{1}{2}} - 2f_i' - f_{i+1}'}{\Delta x} \\
c_1 = f_i',\ c_3 = \frac{f_i' + f_{i+1}' - 2s_{i+\frac{1}{2}}}{\Delta x^2} \\
s_{i+\frac{1}{2}} = \frac{f_{i+1} - f_i}{\Delta x} = linear\ slope
\end{aligned}$$

It can also be rearranged into a different form:

$$f(x) = f_i H_1(x) + f_{i+1} H_2(x) + f_i' H_3(x) + f_{i+1}' H_4(x) \tag{2.3}$$

which is commonly used and more suitable for coding purpose, especially for bi- and tricubic interpolant. In contrast, Lagrange interpolation refers to any method that utilizes only f_i without f_i'. The second-order and C^0 smooth linear interpolation that is common among most existing overset-grid CFD codes is the simplest example.

The cubic Hermite interpolant in (2.2) has a continuous first derivative, i.e., it is C^1 smooth, because of the f_i'. It may be C^2 depending on how $\{f_i'\}$ are obtained. It is n^{th}-order accurate ($n \leq$4), if f'_i and f'_{i+1} are (n-1)th-order. Thus, the original slope estimate $\{f'_i\}$ does not need to be more accurate than third-order, since this would produce a maximum of 4th-order interpolant. However, a third-order finite difference stencil is non-symmetric. In order to maintain symmetry, the 4th-order central difference with a 5-point stencil is usually employed. Since the stencil for the computation of certain types of constraint intervals (to be discussed later) is also 5-point, this does not enlarge the stencil.

Near a discontinuity, however, the accurate slope above has the wrong sign and becomes grossly inaccurate (see Fig. 2.3). This happens because this 4th-order slope of a quadric actually consists of two linear slopes, one for the 3-point central and another for the 5-point central, with weights of $\frac{4}{3}$ and $-\frac{1}{3}$ respectively, i.e.,

$$f'(x_i) = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12} = \frac{4}{3}\frac{f_{i+1} - f_{i-1}}{2\Delta x} - \frac{1}{3}\frac{f_{i+2} - f_{i-2}}{4\Delta x}$$

It is the latter linear slope crossing a discontinuity that gives an unrealistically large negative contribution that results in the wrong sign. The problem is serious because it indicates the data is decreasing instead of increasing or vice versa. Incidentally, it does not arise if both linear slopes cross the discontinuity, i.e., if the discontinuity is between i-1 and i+1. This, however, would still result in a very inaccurate slope. The problem of having the wrong sign can be avoided by requiring the 4th-order slope to lie within a certain interval using the median function. This would be discussed in the next subsection.

Localness in the determination of f'_i and thus the interpolant (2.2) is important when storage requirements are critical, e.g., on parallel computers with local memory, in multidimensional problem or very large data sets. In contrast, the complete spline interpolant, using f'_i that makes (2.2) C^2 smooth, is not local.

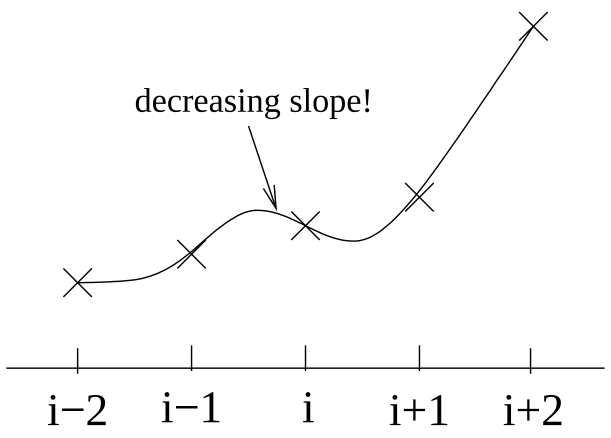


Figure 2.3: Slope of a quadric with wrong sign at point i near a discontinuity.

For interpolations in CFD, localness is favored at the expense of C^2 property, which is not required by Euler's equation.

2.2.3 Monotonicity and Loss of Accuracy at Extrema

The interpolant (2.2) may not be monotone between i and $i+1$ even for locally monotone data. To preserve monotonicity of the data, f'_i and f'_{i+1} must be constrained to lie in the interval

$$f'_i, f'_{i+1} \in [0, 3s_{i+\frac{1}{2}}] \tag{2.4}$$

where $s_{i+\frac{1}{2}}$ is given in Eq. (2.2). This is equivalent to $f'_i \in [0, 3s_i]$ where $s_i = \text{minmod}(s_{i-\frac{1}{2}}, s_{i+\frac{1}{2}})$. The interval $[0, 3s_i]$ expressed using minmod is the intersection of the two intervals $[0, 3s_{i-\frac{1}{2}}]$ and $[0, 3s_{i+\frac{1}{2}}]$. Minmod thus arises naturally from the constraint of monotonicity interval. This sufficient condition was first observed by de Boor and Swartz [55], and is a simplified version, or a subset, of the more accurate version shown as the shaded region in Fig. 2.4. This region is the necessary and sufficient condition for monotonicity preservation. It was independently found by Fritsch and Carlson [56] and Ferguson and Miller

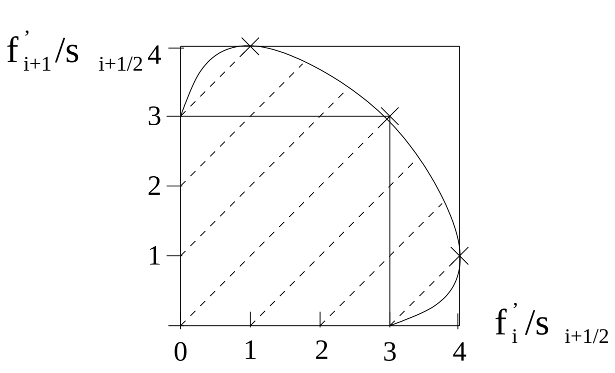


Figure 2.4: Necessary and sufficient condition for monotonicity preservation.

[57]. However, the use of this full region, instead of the simpler square subset, makes the interpolant non-local because of the elliptical region outside the square and thus not easily coded into the algorithm.

The monotonicity preserving (or MP) interval given in Eq. (2.4) employs the minmod function, which returns zero in the case of local extremum where the two linear slopes are of opposite sign. This is necessary to produce a monotone interpolant across the extremum. The problem is, of course, that the data at local extremum are non-monotone to begin with. The insistence on monotonicity of the interpolant for non-monotone data is inconsistent and has the effect of "clipping" or flattening the peak (of a vortex for example) as shown in Fig. 2.5. It degenerates f_i' there to first-order and the interpolant accuracy to second-order (same as linear interpolation). This results in a globally first-order L_∞ norm of the slope.

This phenomenon is especially undesirable in the simulation of vortical flow when a vortex convects into another grid and interpolation near the peak is unavoidable. It causes significant numerical diffusion of the vortex. Deactivating the constraint at the extremum is a possible solution but is unstable in the sense that it might cause a sudden change in the slope. Thus we have a situation in

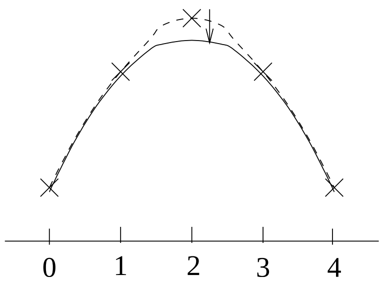


Figure 2.5: Clipping of an extremum due to monotonicity preservation.

which preventing oscillations near a discontinuity creates an accuracy problem near an extremum. This dilemma is also present in a solution reconstruction.

2.2.4 M3 Interpolation

The above problem of conflicting requirements at discontinuities and extrema is caused by a very small or zero s_i acting as one of the two bounds in $[0, 3s_i]$ of (2.4). It can be overcome by enlarging this small MP interval near an extremum. This allows more "room" in the interval for an accurate slope. Several methods are available to restore this accuracy. Here the 'M3' method proposed by Huynh in [59] is chosen for its simplicity. Like the UNO scheme in [54], it is constructed from consideration of the straight line and the two parabolas (through points $i-1, i, i+1$ and $i, i+1, i+2$ respectively) between i and $i+1$ as depicted in Fig. 2.6 (dashed lines). The median of these three curves is defined as the curve that lies between the other two and shown as a solid line in Fig. 2.6. It is always the straight line for data with an inflection point (Fig. 2.1b). Equivalently, the median curve can be used to identify if the data has an inflection point.

The properties of this median curve is worth reviewing before proceeding. The most important is the fact that the median is strictly monotone when the

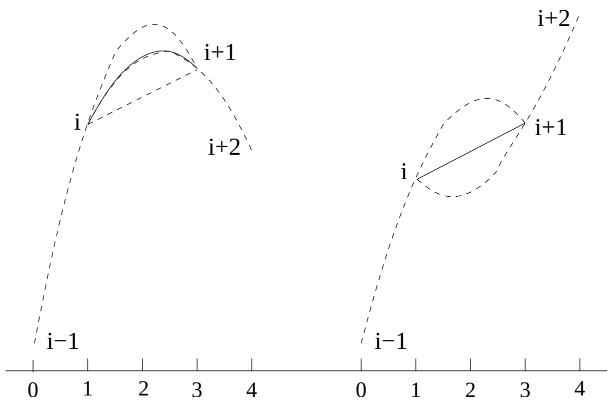


Figure 2.6: Definition of the median (solid line) of two parabolas and a straight line between i and $i+1$ as the middle of the three lines.

data are monotone and thus its slope always lies inside the MP interval of Eq. (2.4). In contrast, neither of the two parabolas from which the median is constructed is necessarily so. The straight line is, of course, always monotone. In fact, none of the two first-order slopes ($s_{i-\frac{1}{2}}$ and $s_{i+\frac{1}{2}}$) and the three second-order slopes for point i always result in a monotone interpolant. This fact can be proven as follows. The equation of the median and its first derivative can be expressed as

$$p_{i+\theta}(x) = f_i + s_{i+\frac{1}{2}}(x - x_i) + d_{i+\theta}(x - x_i)(x - x_{i+1}) \tag{2.5}$$

$$p'_{i+\theta}(x) = s_{i+\frac{1}{2}} + d_{i+\theta}(2x - x_i - x_{i+1}) \tag{2.6}$$

where $\theta = \frac{1}{2}$, $x_i \leq x \leq x_{i+1}$ and

$$d_{i+\frac{1}{2}} = minmod(d_i, d_{i+1})$$
$$d_i = \frac{s_{i+\frac{1}{2}} - s_{i-\frac{1}{2}}}{x_{i+1} - x_{i-1}}$$

These same equations also describe the two parabolas when $\theta = 0, 1$ respectively. Considering increasing data only ($s_{i+\frac{1}{2}} \geq 0$) on a non-uniform mesh, the maximum value d_i can attain is $d_i < s_{i+\frac{1}{2}}/\Delta x_{i+\frac{1}{2}}$ since $s_{i-\frac{1}{2}} \geq 0$ for monotone data and $x_{i-1} < x_i$. Similarly, the minimum value d_{i+1} can attain is $d_{i+1} > -s_{i+\frac{1}{2}}/\Delta x_{i+\frac{1}{2}}$ since $s_{i+\frac{3}{2}} \geq 0$ for monotone data and $x_{i+2} > x_{i+1}$. Therefore we obtain, from the definition of $d_{i+\frac{1}{2}}$,

$$\frac{-s_{i+\frac{1}{2}}}{\Delta x_{i+\frac{1}{2}}} < d_{i+\frac{1}{2}} < \frac{s_{i+\frac{1}{2}}}{\Delta x_{i+\frac{1}{2}}} \tag{2.7}$$

Multiplying by $(2x - x_i - x_{i+1})$ throughout, one gets a new inequality with a larger bound

$$-s_{i+\frac{1}{2}} < d_{i+\frac{1}{2}}(2x - x_i - x_{i+1}) < s_{i+\frac{1}{2}} \tag{2.8}$$

since $|2x - x_i - x_{i+1}| \leq \Delta x_{i+\frac{1}{2}}$. Finally, adding $s_{i+\frac{1}{2}}$ to the inequalities to obtain, using $p'_{i+\frac{1}{2}}(x)$ from Eq. (2.6)

$$0 < p'_{i+\frac{1}{2}}(x) < 2s_{i+\frac{1}{2}} \tag{2.9}$$

which shows the slope is always positive. This implies that $p_{i+\frac{1}{2}}(x)$ is strictly increasing (thus monotone) and that the MP condition Eq. (2.4), i.e.,

$$p'_{i+\frac{1}{2}}(x_i), p'_{i+\frac{1}{2}}(x_{i+1}) \leq 3s_{i+\frac{1}{2}} \tag{2.10}$$

is always satisfied. The same derivation for a uniform mesh results in a tighter bound

$$\frac{1}{2}s_{i+\frac{1}{2}} < p'_{i+\frac{1}{2}}(x) < \frac{3}{2}s_{i+\frac{1}{2}} \tag{2.11}$$

The slopes of the median curve at i and $i+1$ are then evaluated as $f'_{i+\frac{1}{2}}(x_i)$ and $f'_{i+\frac{1}{2}}(x_{i+1})$. These two slopes can be considered as the second order extension of the first-order slopes $s_{i-\frac{1}{2}}$ and $s_{i+\frac{1}{2}}$. The minmod of the two resulting slope values at each point i,

$$t_i = minmod(f'_{i-\frac{1}{2}}(x_i), f'_{i+\frac{1}{2}}(x_i)) \tag{2.12}$$

which can be proven to be $O(\Delta x^2)$ accurate compared with s_i, is then used to extend the monotonicity interval Eq. (2.4) to

$$f_i' \in [0, 3s_i, \frac{3}{2}t_i] \tag{2.13}$$

This extended interval preserves extrema and takes effect only on extrema where data are supposed to be non-monotone. The factor $\frac{3}{2}$ is chosen as the upper limit in (2.11). These derivatives are then used as the final values for the cubic interpolant in (2.2). We note here that t_i is the final UNO slope first proposed in the landmark paper by Harten et al. [54] to achieve uniform 2^{nd}-order accuracy. The above 'M3' method is also used for solution reconstruction in [60] after modification of the coefficients.

2.2.5 Limiter Functions

Methods to compute slopes that satisfy the constraint (2.4) are generally divided into two types: i) a slope computed first from finite difference and then combined with a constraint (as described in the previous section) and ii) a nonlinear averaging function (limiter function) of the first-order linear slopes on the left and right of the point i, $s_{i-\frac{1}{2}}$ and $s_{i+\frac{1}{2}}$, computed in a one step procedure that automatically satisfies the constraint. This was first presented in [61] and [62]. It was shown by Hunyh in [59] that there is no linear formula for f_i' above first-order that also automatically satisfies (2.4). There are, however, many nonlinear formulas that do. They are called limiter functions and their properties will be briefly discussed in this section.

Since a limiter function just averages $s_{i-\frac{1}{2}}$ and $s_{i+\frac{1}{2}}$ and does not use any finite difference formula, it differs from the finite difference plus constraint method in several aspects: i) there is complete freedom in designing a formula for the slope, giving rise to many new limiters over the years, ii) its order of accuracy is not immediately obvious, but it is at least first-order and at best second-order accurate (the necessary condition will be proven shortly), iii) it must possess certain necessary properties (that do not arise in the finite difference formula) as given below:

a) From the requirement of unit independence, it must be homogeneous of degree one, i.e.,

$$f_i'(ks, kt) = kf_i'(s, t)$$
$$f_i'(s/t, 1) = f_i'(s, t)/t$$

or

$$f_i'(s, t) = tf_i'(r) = tg(r) \tag{2.14}$$

where $k = 1/t$ is the unit conversion factor, $r = s/t$, $s = s_{i-\frac{1}{2}}$ and $t = s_{i+\frac{1}{2}}$. This requirement implies that a limiter is really a function of one variable, i.e., ratio of the two linear slopes, or $g(r)$.

b) f_i' lies between $s_{i-\frac{1}{2}}$ and $s_{i+\frac{1}{2}}$, or $g(r) \in [1, r]$. This ensures f_i' to be at least first-order and thus the interpolant (2.2) at least second-order accurate.

c) f_i' is symmetric in s and t:

$$f_i'(s, t) = f_i'(t, s)$$
$$tg(r) = sg(1/r)$$

Thus

$$g(r) = rg(1/r) \tag{2.15}$$

The variable $1/r$ of g implies that $g(r)$ for $r > 1$ is defined by $g(r)$ for $0 \leq r \leq 1$ and thus only $g(r)$ in the interval $0 \leq r \leq 1$ needs to be defined. To examine the slope of (2.15) at the symmetric point $r = 1$, (2.15) is differentiated (assuming g is C^1) to yield

$$g'(r) = g(1/r) - \frac{1}{r}g'(1/r) \tag{2.16}$$

Substituting $r = 1$ and since $g(1) = 1$,

$$g'(1) = \frac{1}{2} \tag{2.17}$$

While this symmetry property is not absolutely essential nor necessarily true for the function f, nevertheless, the above analysis shows that it gives rise to desirable properties.

d) f_i' satisfies the MP constraint (2.4) or $g(r) \in [0, 3minmod(1, r)]$.

Since the best accuracy attainable from the combination of two first-order slopes is second-order, a limiter function is at best second-order accurate. The necessary condition to achieve this is shown next. Since the second-order accurate slope of f_i is $f_i' = (s + t)/2 = t(r + 1)/2$, or

$$g_2(r) = (r + 1)/2$$

its derivative $g_2'(r)$ is also $\frac{1}{2}$ anywhere on $0 \leq r \leq 1$. For any other limiter $g(r)$ with the same slope at $r = 1$, i.e., $g'(1) = \frac{1}{2}$, the order of accuracy of the difference $g(r) - g_2(r)$ is $O((1 - r)^2)$ at worst (which corresponds to a quadratic $g(r) - g_2(r)$). Since $1 - r = (t - s)/t = O(\Delta x)$ or first-order accurate, we obtain

$$g_2(r) - g(r) = O(\Delta x^2)$$

Thus, $g(r)$ is second-order accurate if and only if $g'(1) = \frac{1}{2}$.

Some of the more well known limiters are given in [59] and will not be further elaborated here.

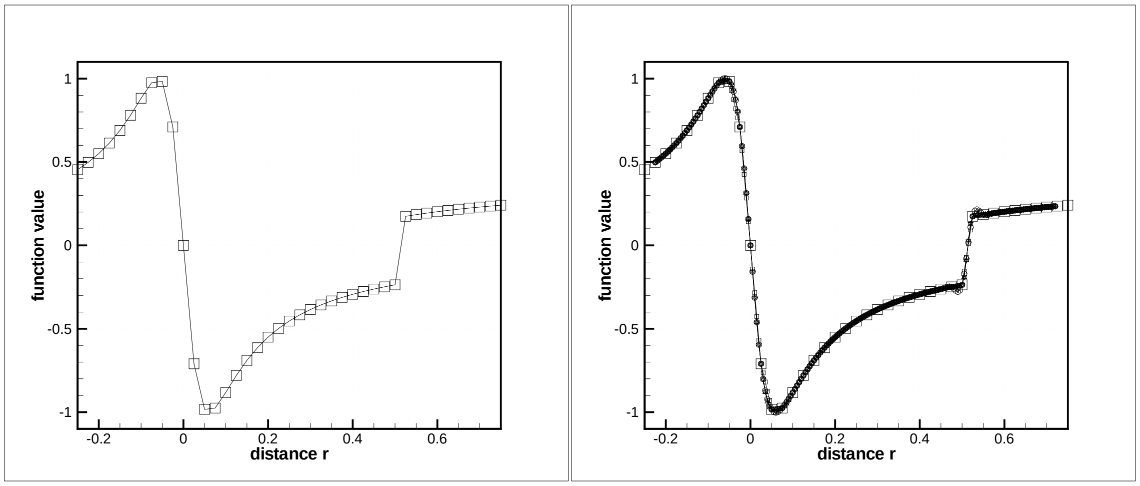
2.2.6 Interpolation Example

An example of interpolation of a specially constructed function will be given in this section. Fig. 2.7a) depicts a function of the velocity profile of a Scully vortex model $v_\theta = \Gamma/2\pi\mathrm{r} \cdot r^2/(r^2 + r_c^2)$ with a discontinuity towards the right. Equally spaced sample points are marked in squares. Here, Δx =0.025 and r_c is set to 0.06 so that the peaks are not on the points. Fig. 2.7b) show the interpolated values from these samples for four different interpolants. The linear case, although satisfactory at discontinuity, is inaccurate and monotone at extremum. The unlimited cubic has the opposite problem, i.e., oscillation at discontinuity. This is solved by the monotone cubic, which eliminates the oscillation but, like the linear, introduces a new monotonicity ('clipping') problem at the extremum. Finally, this latter problem is significantly remedied by the M3 cubic without disturbing the desirable traits at the discontinuity. The 5-point 4^{th}-order finite difference formula is used as the original f_i' for the 3 cubics above.

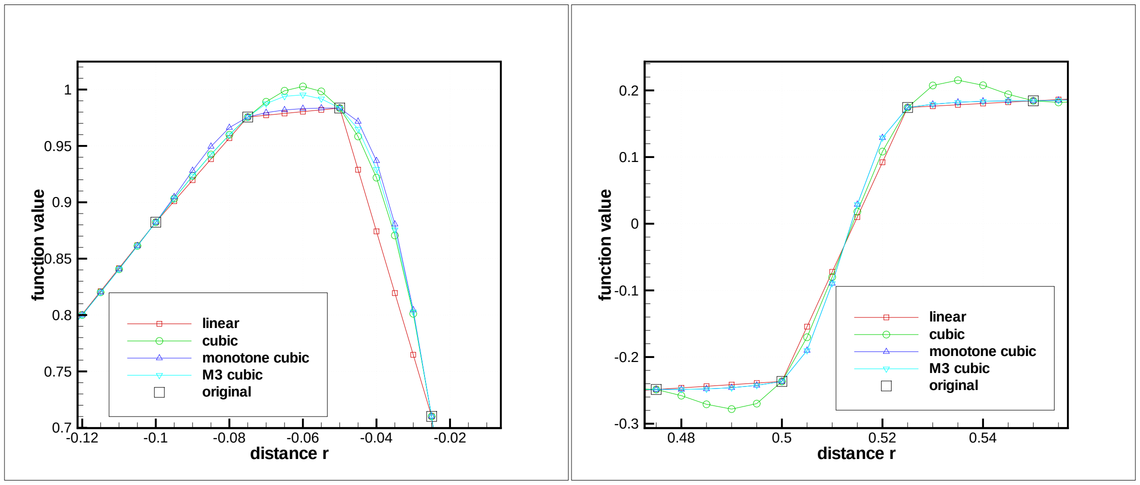
2.2.7 Bicubic Interpolation

Interpolations for multidimensional problems can be calculated by either repeating the 1D procedure in every dimension, or extending it to an inherently multidimensional bi- or tricubic interpolation. The latter is generally preferred for economic reason if the stencil size is large. The essence in its application to overset grids is summarized in this subsection.

A bi- or tricubic interpolant [63] [64] contains 4^{ndim} terms and, since it is inherently multidimensional, contains many cross terms. The definition of function monotonicity is less clear and more complicated. In addition, the cross derivatives also need to be constrained. It is based on another constraint for



a) A function with two extrema and a discontinuity.



b) Magnified views at an extremum (left) and the discontinuity (right).

Figure 2.7: An interpolation example using four different interpolants (one linear and three cubics).

the cubic function - that f does not change sign on $[f_i, f_{i+1}]$. For an increasing function, we have

$$-\frac{3f_i}{\Delta x} \leq f'_i \qquad \text{and} \qquad f'_{i+1} \leq \frac{3f_{i+1}}{\Delta x}$$

The proof is given in Carlson and Fritsch [63]. If f above is replaced by another cubic $f_y(x)$, the two inequalities become a constraint for f_{xy}.

This solver employs piecewise Hermite cubic monotone interpolation for both coordinates and field solutions. This is shown to be important in chapter 4 when interpolating from a high gradient region. Besides accuracy, smoothness of coordinate transformation and consistency with grid metrics in the flow solver are two important reasons for using Hermite cubic rather than linear interpolant.

The coordinate transformation of a curvilinear grid can be exactly and analytically defined when its primitive metrics and cross derivatives are taken into consideration. This piecewise analytical expression is precisely the piecewise Hermite tricubic interpolant and is used in the code for coordinate calculations everywhere. Contrary to its name, this interpolant produces exact (not approximate) and consistent coordinates. It results in a C^1 smooth transformation function and is the lowest order interpolant consistent with the grid metrics used by the solver. The more popular decoupled cubic interpolant is free of cross derivatives but is not C^1. The commonly used trilinear interpolation is not consistent because it ignores the metrics and, furthermore, is only C^0 continuous.

Cubic interpolation has an added benefit of allowing a larger spacing amplification factor between two interpolated grids (usually about 2:1 ratio). Its biggest drawback is the tremendous cost disadvantage associated with an enlarged stencil. This will be further elaborated in the section "Curse of Dimensionality" later.

For the interpolation of 2D coordinates in overset grids, the bicubic can be functionally written as

$$\mathbf{x}(\xi, \eta) = fn(\mathbf{x}_{i,j}, \mathbf{x}_{\xi i,j}, \mathbf{x}_{\eta i,j}, \mathbf{x}_{\xi\eta i,j}; \xi, \eta) \tag{2.18}$$

where $0 < \xi, \eta < 1$, i, j represents all the corner points of a cell, $\mathbf{x}_{i,j}$ are the grid coordinates, $\mathbf{x}_{\xi i,j}, \mathbf{x}_{\eta i,j}$ are the primitive grid metrics and $\mathbf{x}_{\xi\eta i,j}$ is a cross derivative defined to nullify the metric invariants in the source term of the transformed Euler equations, i.e. $(\mathbf{x}_\eta)_\xi - (\mathbf{x}_\xi)_\eta = 0$. These are also all the grid related quantities that appear in the flow solver (together with the metrics on the mid-point of each cell edge for the Riemann solver). The bicubic is thus the lowest order coordinate transformation consistent with the solver if the same grid metrics in the transformed flow equation are used in (2.18). The commonly used bilinear function is not consistent because it ignores the metrics. In this work, interpolations of solution and coordinates employ bi- or tricubic interpolant. The solver, however, provides another option to use cubic interpolant for boundary conditions.

2.2.8 Reconstruction vs. Interpolation

We are now ready to note some of the practical differences between reconstruction and interpolation in CFD as mentioned in a previous section. Reconstruction, or extrapolation, is usually required for all data points whereas interpolation is usually needed only for a few selected points away from the boundary. This subtle difference discourages the use of non-local methods such as compact scheme or spline methods to compute the slopes $\{f'_i\}$ for interpolation. Compact scheme has the advantages of better spectral resolution, smaller stencil for the same order of accuracy and smaller or no biased stencil at the boundary. The fact that these selected interpolation points are away from the boundary also implies that no special one-sided formulas are needed. Reconstruction, on the other hand, is required near the boundary.

Another difference is that the use of both f_i and f_i' in reconstruction yields second-order accurate solution, whereas the same data and slopes would result in a fourth-order interpolant. That is why linear Lagrange interpolation which does not use f_i' and employed in many existing overset grid solvers is sufficient to match the solver's second-order accuracy in reconstruction that uses $\{f_i'\}$.

Furthermore, reconstruction in a cell is generally only piecewise continuous and extends from $i-\frac{1}{2}$ to $i+\frac{1}{2}$ while an interpolation is at least globally C^0 and extends from i to $i+1$.

Finally, the coefficient for the monotonicity constraint in (2.4) is 3 (cubic) for interpolation while it is 2 (quadratic) for reconstruction. This latter coefficient of 2 is due to the requirement that the reconstructed value from $i-\frac{1}{2}$ to $i+\frac{1}{2}$ using f_i and f_i' must satisfy $f_{i-1} \leq f(x) \leq f_{i+1}$.

2.2.9 Curse of Dimensionality in High-Order Interpolation

Consider the increase in computational cost per point per variable when a problem is extended from 2D to 3D. It is 50% for an interior point, that is, from two fluxes to three. For an interpolated point, however, it is about n times, where n is the number of stencil points in each direction ($n = 6$ for monotone cubic interpolation). To be exact, the increase is n times for coupled and from $n+1$ to n^2+n+1 for decoupled interpolation in each direction respectively.

This is many times larger than the 50% for an interior point. This also comes in addition to the usual "curse of dimensionality" for total number of points in a problem extension from 2D to 3D. For a steady state case, this problem can be partly avoided by turning to high-order only when close to convergence.

The problem can also be partly overcome if the interpolation is coupled in the three directions, i.e., tricubic. In this case, derivatives can be precalculated on all donor cells. This saves cost for nearby interpolated points with many common stencil points. This is not as effective for decoupled interpolation, since new sets of points are created after each direction.

The number of derivative evaluations for a 3D tricubic and decoupled cubic interpolation is $4^3 - 2^3 = 56$ and $2(n^2 + n + 1)$ respectively. It is a variable in the latter case and is equal to 14, 42 and 86 for 2-, 4- and 6-point stencils. We note that 32 of the 56 derivatives in the former case are cross derivatives.

In light of the above, it is recommended that a single layer of grid outer boundary points be interpolated instead of the more commonly used two layers. This also minimizes the required overlap region besides the high cost problem mentioned above. Tricubic is preferred to decoupled cubic due to its more rational derivation and lower cost for large n. Biased stencils for first interior points are necessary if 5-point interior stencils are to be maintained. They are, however, not necessary if a 4^{th}-order compact scheme is used. For hole fringe points, however, the penalty for using as many layers as possible is usually not as high since the number of hole points is usually much smaller. In addition, a multi-layer hole fringe has the advantage of better insulation against contamination from invalid hole solutions as explained in the last section.

2.3 Computing Environment

Since the introduction of the Earth Simulator supercomputer in Japan in 2002, there has been a revival of interest in the architecture of vector processors in the United States. Unlike traditional vector machines with only a small number of processors, modern vector architecture is scalable and massively parallel. Besides being an impressive 5 times faster than its closest competitor in its debut, the most surprising feature about Earth Simulator is the very high sustained speed (anywhere from 30% to 88% of peak) achieved in some real world applications. This is significant compared with the average of about 10% achieved in the much more popular cluster architecture built with commodity superscalar cache-based processors. While Japanese vendors have never really abandoned vector architecture, heavy emphasis on acquisition cost in the U.S. has resulted in the present situation in which Cray Inc. is the only U.S. vendor still making custom vector processors. Its latest model is the Cray X1 [65] with its new distributed shared-memory architecture. Although the initial cost per processor of a Cray X1 is several times that of a typical cluster, total life cycle cost of both systems are comparable [66].

Some of the advantages of a vector processor are that it needs less data dependency checking, lower processor complexity and thus power consumption, fewer instructions (for arrays) and more processor parallelism compared with a scalar processor. Performance evaluation of the Cray X1 has been published by several groups of researchers ([67], [68], [69]).

2.3.1 Code Migration to the Cray X1

One advance feature of the Cray X1 that is of general importance to the work in this thesis (and to overset grids technology in particular) is the built-in hardware support for one-sided communication between processors, or global address space (GAS, to be elaborated in the next section), with its Co-Array Fortran (CAF) implementation [70]. CAF is a relatively new parallel Fortran first proposed by Prof. Numrich [71] with a small language extension using co-arrays to facilitate data passing in a parallel code. This language feature is slated to become part of the next Fortran standard. CAF gives the Cray X1 a significant advantage over other massively parallel systems, as its compiler is not generally available. A CAF compiler for distributed memory architecture in clusters is still in its early stage and is even less generally available (see [72] and the references therein).

Parallelization on the Cray X1 is performed in a hierarchical manner by the compiler and the programmer. At the lowest level is vectorization. This is performed by the compiler on vectorizable loops (such as the 1D loops in the modified subroutines of the Riemann solver, variable extrapolation, solution update, etc.). Data are fed into the two vector pipelines in each processor. Frequently used hard-to-vectorize loops (such as the tridiagonal solver) impose significant performance penalties on the code. The next level up is multi-streaming. This is optional and activated/deactivated through the compiler command. This option performs limited parallelization and distributes every 1D loop in a structured grid to each of four processors which together comprise a multi-streaming processor (MSP). The highest level of the hierarchy is inter-processor communication (using models like MPI, CAF, OpenMP, etc.) and is the responsibility of the programmer. This includes communication across the 4 MSPs in a node, between nodes in a cabinet and between cabinets, with increasing physical distance.

Migration of a serial scientific code to a parallel-vector machine requires some modification for optimal performance. This involves a reassessment of the code for both single-processor and parallel (or communication) performance.

2.3.2 Single Processor Optimization

Single processor optimization for Cray X1 primarily implies vectorization of frequently used loops. Certain existing numerical schemes and functions are inherently non-vectorizable and thus perform poorly on a Cray X1 processor. These include Gauss-Seidel (GS)-like implicit time integration, alternating indexing in MultiGrid, almost the entire process of overset grids connectivity, etc. Vectorization of other segments of a structured-grid CFD code oftentimes increases the length of the code because of the need for the creation of more vectors (or arrays) and loop fission, or breaking up of a loop into several smaller ones and storing temporary variables in arrays. This is the case in the Riemann solver and the solution updating subroutines.

As an example of the performance penalty for unvectorized code, approximate solution of a 1D block tridiagonal system using LUSGS implicit time integration scheme is about half as costly as a 1D inviscid flux evaluation in a superscalar processor. On the scalar unit of a Cray X1 processor, this figure skyrockets to about 200%. This penalty is similar for the solution of a scalar tridiagonal system for calculating derivatives using compact schemes. One way to minimize this is to convert the system into a vector tridiagonal system. Initial evaluation of the (unmodified) connectivity code as given in this thesis on a Cray X1 also confirms a severe penalty of the order of >100%.

2.3.3 Processor Communication

As the number of processors and/or overset grids increases in a computation, interpolation of boundary values become increasingly dominant in the overall CPU time. While the former scenario necessitates more communication between processors, the latter increases the number of interpolated points. Interpolations may require solutions that reside in remote memory, and their access can be expensive due to the required communication overhead and distance the messages need to travel. Thus the parallelization technique and its relation to system architecture can become very important in efficiency optimization. This section discusses some significant communication issues of particular relevance to overset grids.

In the context of numerical PDE, parallelization can be implemented in two ways, closely corresponding to the two methods of processor communication, i.e., i) one-sided or ii) two-sided. These two methods give rise to very different codes, programming efficiency and communication efficiency. Whereas the latter is commonly implemented with the industry standard MPI library, the former is relatively new and has important advantages over MPI. Its implementation, such as that in CAF, is based not on library subroutines but on a simple language extension to arrays called co-array. The most significant difference between these two methods is that two-sided communication requires coordination, or handshaking, with the remote processor in order to establish a data communication channel while the other does not. This mode of communication is invoked by a send and a receive pair of subroutine calls from the two processors. In other words, remote data access in one-sided communication does not involve the remote processor, at least in concept. In a Cray X1, this is supported in hardware with its global memory address space. This difference in concept has especially important implications for overset grids methodology.

With the above in mind, it is instructive to recognize that two types of data communication scenarios are frequently encountered in the parallel execution of numerical PDE: a) at the common boundaries between any two neighboring subdomains (assigned to two different processors) of a decomposed structured or unstructured grid where an extra layer of cells are duplicated for holding received data, and b) at the outer boundaries and hole fringes of overset (overlapping) grids where interpolations are required for the supply of boundary conditions. Both type a) and b) can use either one-sided or two-sided communication methods mentioned above. It is also possible to have both types of communication scenarios in an overset grids CFD code in which the decomposed subdomains of some grids in a) are distributed among the processors just like other independent grids of the mesh in b). But the following discussion is restricted to communication problems in type b) only.

While both the size of the passed message and the identity of the communicating processors are known a priori in the former type, neither of these is generally known in the latter. In addition, these two unknowns in type b) may vary in the course of the computation for grids with relative motion. A consequence of this is that two-sided communication protocol is not, for programming purposes, as convenient and natural as one-sided since no processor would have any knowledge when to expect a data request, if there is any at all, from any other processor. In other words, a requesting processor can not simply call a 'receive' subroutine whenever it needs to access remote data because the remote processor does not know when this might happen. While it is, in principle, possible to set up the code to call a probing subroutine to check for any such request after a preset period of time, this is impractical and time consuming at best. None of these is a problem for type a) since everything about the data exchange is known and remains constant during computation.

2.3.3.1 Two-sided Communication in Overset Grids

For the data passing scenario in overset grids of (type b above), the usual method to access remote data in a two-sided communication is to pass one very long message between any processor pair in a bulk communication once and only once at the end of every time step instead of many short messages each for every interpolated boundary point whenever it is needed. Because of this, every processor must first calculate and gather the local data (interpolated solutions from all donor cells that reside locally) that it owns and required by all other processors. It must then store them in an array `Qsend(:,ndonor,nproc)`, where `nproc` indicates the receiving processor ID and `ndonor` counts local donor cells destined for `nproc`. The information required in the gather operation of this step, `iBndPtDonor(:,nbndpt,ngrid)` (and `iHolePtDonor(:,nholept,ngrid)`) where `nbndpt` and `nholept` count receiver points in grid `ngrid`, is obtained from the output of the connectivity subroutine (see Chapter 3). Once all the data are prepared, a synchronization barrier in the code is executed to ensure all processors have finished the interior computations and ready to receive boundary values before communications are initiated. Data received are first stored in a separate array `Qrecv(:,ndonor,nproc)`, where `nproc` is the sending processor ID, and then scattered, or distributed, to their respective receiver points.

Solution for the next time step can begin immediately in any processor as soon as its scatter operation is complete without another synchronization barrier since no more local data are needed by any other processor. This is an advantage compared with one-sided communication, and a result of the fact that interpolations are calculated remotely in the processor of the donor grid and only the interpolated solutions are passed. It enables combined load balancing of the solver and the interpolation / communication steps.

The difficulty with this method lies in the fact that all the remote interpolated data required by the local grids in any processor are possibly scattered around in multiple non-contiguous blocks in different processors. In other words, `Qrecv(:,ndonor,nproc)` is not stored in the same order as the original `iBndPtDonor(:,nbndpt,ngrid)`. This requires additional book-keeping, receiver point- and donor cell-counting, new arrays (with varying memory size for problems with moving grids) and most importantly, complex coding logic. Thus the implementation of this gather/scatter operation and optimization in the code can be very lengthy. The resulting code is also much less readable and understandable. In short, the different order of `Qrecv()` (gather/scatter operations) and `iBndPtDonor()` results in extensive code modification because of the required bulk communication, which in turn is due to the nature of two-sided communication.

2.3.3.2 One-sided Communication in Overset Grids

One-sided communication as implemented in CAF, in contrast, is free of this difficulty. It requires very minimal modification to only one statement (besides the co-array declarations and load balancing subroutine), incorporating a co-array to the solution array in the body of the serial code where the stencil data is accessed, i.e., from `Q(nDonGr)%node(:,i,j,k)` to `Q(nDonGr)[nproc(nDonGr)]%node(:,i,j,k)`.

There is, however, a trade-off for this programming simplicity. It results in a lot more data passing between processors and, indirectly, a necessary second synchronization barrier before the next time step that would increase processor idle time for any load imbalance in the interpolation / communication step. The former results from the fact that the data for the entire stencil in the donor grid (which in this work contains 4^3=64 points) are passed to the requesting processor where the interpolant would then be calculated. The interpolant is not calculated remotely because this would then require the remote processor to gather the local interpolated data in a different order than the original one in `iBndPtDonor(:,nbndpt,ngrid)`. In other words, passing only the computed interpolants instead of the whole stencils would require the remote processor to go through the same tedious procedure as described above in the 'two-sided' subsection.

This problem of a much larger message size seems to affect only overset grids methodology because of the need to interpolate from a stencil. Unstructured grids or any domain decomposition methods that do not need interpolation should be free of this difficulty.

Because the interpolations are calculated locally, interior solver for the next time step can not begin until all interpolations have been computed. Otherwise the stencil data for the unfinished interpolations of the previous time step might be overwritten. This necessitates a second synchronization barrier that prevents combined load balancing of the solver and the interpolation / communication steps. The load imbalance of the latter step for linear interpolation and small number of processors in close proximity may be tolerable. It can, however, become very expensive if this is not true. This is especially so for the very costly cubic interpolation (as explained in the section on Curse of Dimensionality in this chapter). The built-in hardware support for global address space in Cray X1 reduces the impact of this penalty.

One can of course choose to adopt the same tedious procedure that the two-sided model requires but use co-array to facilitate the bulk communication. This would, however, significantly nullifies the inherent advantage and defeats the purpose of one-sided model in overset grids.

2.3.3.3 Interim Summary

A brief summary of the pros and cons of one- and two-sided communication models in their applications to parallel numerical PDE using overset grids as detailed above is appropriate. The two-sided model is inefficient from the programmer's point of view, but it results in a much smaller message size and needs only one synchronization which in turn enables the combined load balancing of computation and communication. One-sided model, in contrast, has exactly the opposite of the above three features.

Chapter 3

A New Approach to Overset Grids Connectivity

Numerical simulations of partial differential equations (PDE) for complex geometries, such as those encountered in computational aerodynamics for aerospace or marine applications, can be approached with any of several types of grids such as unstructured or overset structured. This chapter is concerned with the latter, which can be dated back to the 1983 paper by Steger et al. [2].

In this chapter, the inner workings of a fundamentally different approach to structured grid connectivity in overset method is presented and several related concepts introduced. This approach is based on a cell comparison process instead of the traditional hole cutting as an essential ingredient. The final results of conventional hole cutting is a byproduct of this process and the holes that arise at the end are both automatically optimum, in a sense to be explained later, and not affected by any complication or imperfection on the wall surface. Furthermore, the solver does not need to blank out invalid hole points for solution update at every time step. The prevention from contamination of the field points is naturally provided by the mathematical property of hyperbolicity. Neither does this approach require any kind of tree data structure to expedite the donor search process. Its efficient execution relies on the physical proximity and continuity of the test points (boundary and hole fringe points) and the detection of non-overlapping grid-pairs. Only simple stencil walking with an approximate but fast cross and dot product inside/outside test is used. This approach can also be regarded as a generalized method of cutting holes around both flow refinement grids and walls indiscriminately. The greatest strength of Implicit Hole Cutting is the simplicity of its concept and implementation which results in a very natural and compact algorithm.

In addition to this new approach, this chapter also discusses the tricubic interpolation used in the algorithm. This interpolation provides an exact description of the coordinate transformation that is both C^1 smooth and consistent with grid metrics used in the solver. These properties are not found in the more commonly used linear interpolation. Unfortunately it also increases the cost of the execution quite significantly.

Overset grid methodology is well suited to problems involving relative motion between body components. Some applications that are of practical interest to the aerospace community include rotorcraft, turbomachinery, aircraft store separation, rocket stage separation, air drop, pilot ejection, etc. The method is especially important for rotorcraft and other problems with high speed grid motion. Problems with lower speed motion (e.g. store separation) have been attempted, without using overset grids, with a series of steady state problems each having a different relative position. The method has also seen wide applications in other areas such as, ships/submarines, road vehicles, biological flow, environmental flow, oceanography, etc.

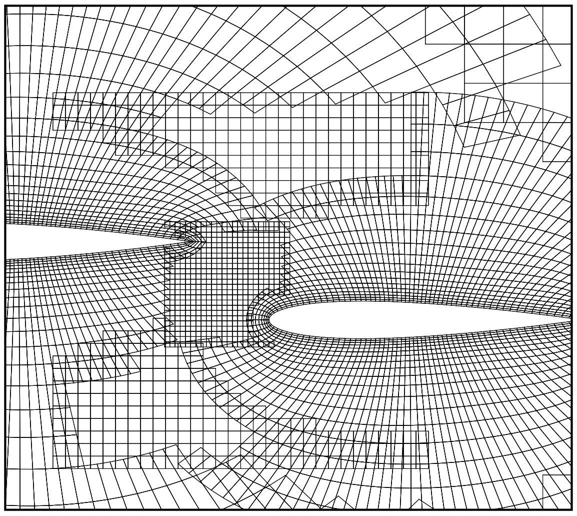


Figure 3.1: A composite of five grids (two curvilinear for airfoils and three nested Cartesian for a vortex) each with multiple holes cut out by both vortex grids and airfoils. New boundaries are frequently generated around holes and cuts.

3.1 Introduction to Overset Grids Method

Overset grids methodology is based on an elementary requirement in the theory of PDE, i.e., boundary condition is required everywhere along the boundary of the domain, including any holes cut out inside the domain (Fig. 3.1). New boundaries around holes and cuts are frequently generated in overset grids, especially for grids with relative motion. For a hyperbolic PDE, the boundary condition is, of course, subject to the constraint of characteristic conditions. For the sake of simplicity, however, most codes do not strictly satisfy this.

Consider an assembly of overlapping structured grids around an aircraft. The function of a grid connectivity algorithm is to establish a connection (through interpolation of boundary values) between all these independent and arbitrarily arranged grids so that they behave like a single grid, or more appropriately, like an unstructured cluster of structured grids interwoven with overlapping boundaries such as that shown in Fig. 3.1, which shows a composite of five grids. All redundant points that are not part of the overlapping boundaries are discarded as hole points. Each grid in the figure has multiple holes and some of them intersect.

The output from the algorithm is a list of these boundary interpolation points (their indices and grid numbers) and their donor cells (indices of one corner point, grid numbers and locations of the points in the cells in curvilinear coordinates). Once connectivity is established, solutions at interior points of all grids are computed independently as usual except that hole points are not updated, while those at the overlapping boundaries are interpolated so that all grids can feel the presence of all the others and behave as one. This assembly of either structured or unstructured grids provides tremendous flexibility for problems with very complex geometries and moving components. It also allows independent 'object-oriented' component grid generation that drastically simplifies the whole process.

Grid connectivity as described above is an expensive and daunting task, although some recent versions of connectivity codes have attempted to ameliorate this. These codes are also long with complicated logics, at least when compared with that of a flow solver, which has a much more standardized and straightforward logic.

Grid connectivity for traditional unstructured or finite element grids, on the other hand, is built-in and completely natural. No computation of any kind is required other than having two sets of numbering systems, one each for the cells and the nodes around them, and an array for the cells indicating the indices of the nodes around them. This array establishes full connectivity of the grid. However, since the cells have connections that are rigidly water-tight rather than arbitrarily overlapping, unstructured grids are not as flexible when components move relative to each other and regriding becomes necessary. In the same situation, the same set of structured grids can be used in overset method without regriding. Only their connectivity needs to be re-established. Overset structured grids are really unstructured globally with arbitrarily overlapping connections but structured locally. They thus enjoy both global geometric flexibility and all those advantages associated with structuredness. They may however sometimes suffer from local geometric inflexibility.

Some existing and well-known structured grid connectivity codes for solving PDE are given in chapter one. These are DCF3D [7], Overture [9] [10], PEGASUS [6], Beggar [8], ChalMesh [12]/Xcog [13] DiRTLib [14]/SUGGAR [15] and FASTRAN [11], all of which differ considerably in implementation, technique and detail but similar in approach, that of using walls to cut holes. This is generally outlined in the next section.

3.2 Existing Connectivity Approach

The approach commonly used in overset method for connectivity requirement between grids is divided into two steps. First, a "Chimera" hole-cutting technique is chosen and used to identify those points that are inside a given hole region with any arbitrary shape (that describes an aircraft surface geometry for example). These points are blanked out, i.e. identified in an array i_{blank}, which indicates the inside/outside status of all grid points for all given hole region. The points at the fringe of this initial minimum-size hole are not suitable to receive hole boundary values because of the large difference in grid resolution. The hole is then expanded or resized so that a better grid overlap is achieved. From this, a list of hole fringe points that require information from other grids to serve as boundary conditions can be easily extracted. Together with grid outer boundary points, this intergrid boundary point (IGBP) list includes J,K,L indices and X,Y,Z coordinates of the points and their grid numbers. The i_{blank} array is also used in the flow solver for the purpose of preventing updating of solutions for points in the holes. In this step, the major difficulty lies in the determination of i_{blank}. See Meakin [7] for the latest technique using object x-rays and a summary of other alternative methods such as mark-and-fill, surface normal vector, ray casting and hole-map method. Object x-rays is an ingenious combination of hole map and ray casting methods, exploiting the merits of each while avoiding their disadvantages for 3D objects.

Next, for each point in the IGBP list, a search is conducted to find an optimum donor cell from all potential grids. The three major problems in this step are the determination of i) the inside/outside status of a point for a cell, ii) a search path from a starting cell (may be the cell on the outer boundary closest to the point) to the donor cell (one that encloses the point) and iii) the computational coordinates $\mathbf{s}$ (between 0 and 1) of the point in the cell (usually second-order linear coordinate transformation is assumed). The method for the point status in i) is much less involved than that used in the first step because of the much simpler shape and edges of a cell compared with that of an arbitrary region. Reference [73] lists a number of these 'domain connectivity' methods. Some kind of tree data structure, such as polygonal mapping tree, alternating digital tree etc., is also commonly used in the preprocessing to expedite the search process in step ii). It is interesting to note that it is possible to solve all three problems including the search direction in ii) using only **s**, provided **s** is not too large or the point not too far away. This step is only required when the donor grid is curvilinear. The case for Cartesian donor grid is trivial and the indices of the bottom left vertex of the donor cell are simply

$$J, K, L = INT\frac{\overrightarrow{x_p} - \overrightarrow{x_o}}{\Delta\overrightarrow{x}} + 1 \tag{3.1}$$

where $\Delta\overrightarrow{x}$ are the grid spacings and $\overrightarrow{x_o}$ the coordinates of the bottom left most grid point.

3.3 Implicit Hole Cutting (IHC) - A New Approach

The idea of Implicit Hole Cutting was first reported in [77] for 2D problems and applied to 2D airfoil/shock vortex interaction. This chapter concentrates only on 3D cases although the two main subroutines can be used for both 2D and 3D problems. Figures to help explain the text are mostly 2D for clarity purpose.

Several previously unreported concepts combine to make Implicit Hole Cutting possible, but two issues are central in its understanding. The first is that hole points inside a solid body do not need to be identified and blanked out in the solver in order to perform a valid computation without contamination. The only requirement is that the hole fringe layers be ‘thick’ enough and completely enclose the body. This will be further elaborated in the next section. This implies that the same flow solver for a single structured grid can be used in an overset method with no modification other than increasing the dimension of the relevant arrays by one to accommodate multiple grids, e.g. `P(i,j,k)` to `P(i,j,k,ngrid)`. However, the i_{blank} array is still required for three other auxiliary functions, namely, contour plotting, blanking out hole points for convergence checking purposes in the calculation of residuals for steady state problems and residuals for implicit Newton subiterations. In other words, unsteady problems with explicit time integration will have no use for i_{blank} except for contour plotting.

Secondly and more importantly, the IGBP list and i_{blank} array can be obtained in an alternative method without explicitly knowing where the holes are, cutting them out and expanding them. Neither is it necessary to use any kind of tree data structure in a preprocess to help search for donor cells. This method is a cell selection process (as opposed to a wall cutting process) based on the main criterion of cell size (and may be some other cell properties), and hole cutting around bodies is a byproduct of this operation. Cell size, or inverse Jacobian, is a parameter also required by the flow solver. Thus no additional calculation is needed. It can cut holes around bodies without knowing the exact location of the walls because the 'hole-cutting effect' of the body can be felt through either one of the following two mechanisms, progressively smaller cell size towards the wall, or grid topology with always a fixed index for walls such as K=1. For almost all practical problems, the former alone is sufficient and the latter is usually not needed. This is better understood if one realizes that somewhere embedded in the cell size array is the information of the approximate location of the body. This information is all that is required to cut the optimal holes around the body.

Because a wall is not used as a cutter, hole cutting failures in some existing methods caused by imperfections or difficulties at the wall (e.g. thin bodies, mismatch/openings or leaks on wall surface, etc.) are irrelevant. One example as shown in Fig. 3.2 is an opening/gap on the surface that causes the sweep over all surface cells to proceed through the opening to the inside surface of the body and emerges from the same opening after sweeping through all the cells on the inside surface. Extra code logic is needed to prevent this from happening.

The absence of such failure also eliminates the possibility of orphan points or leakage. Orphan points caused by insufficient overlap, though, must still be checked.

The hole thus obtained from this method is also automatically optimum in the sense that the match in cell size between the first layer of hole fringe points and their donors is always the best. There is no need for a separate step for hole optimization or overlap minimization with user specified or calculated parameters such as offset value, wall distance or donor health/potential. Nor does it require the grids to be ordered by priority or grouped by hierarchy.

Offset value is a user-specified integer indicating the number of cells a hole is to be enlarged. This can be used to correct the status of points inside sharp bodies (e.g. trailing edge) that are mistakenly labeled ‘normal’ because of failure in the chosen hole-cutting method. This inevitably introduces arbitrariness and increases user workload and expertise requirement.

Another problem is the use of distances between points and the wall in some existing methods (especially for overset unstructured grids) for cutting holes. This criterion has the undesirable effect of cutting the hole at the same location regardless of possibly large difference in grid resolution. This could, in some cases, potentially result in hole fringe points interpolating from donors whose cell

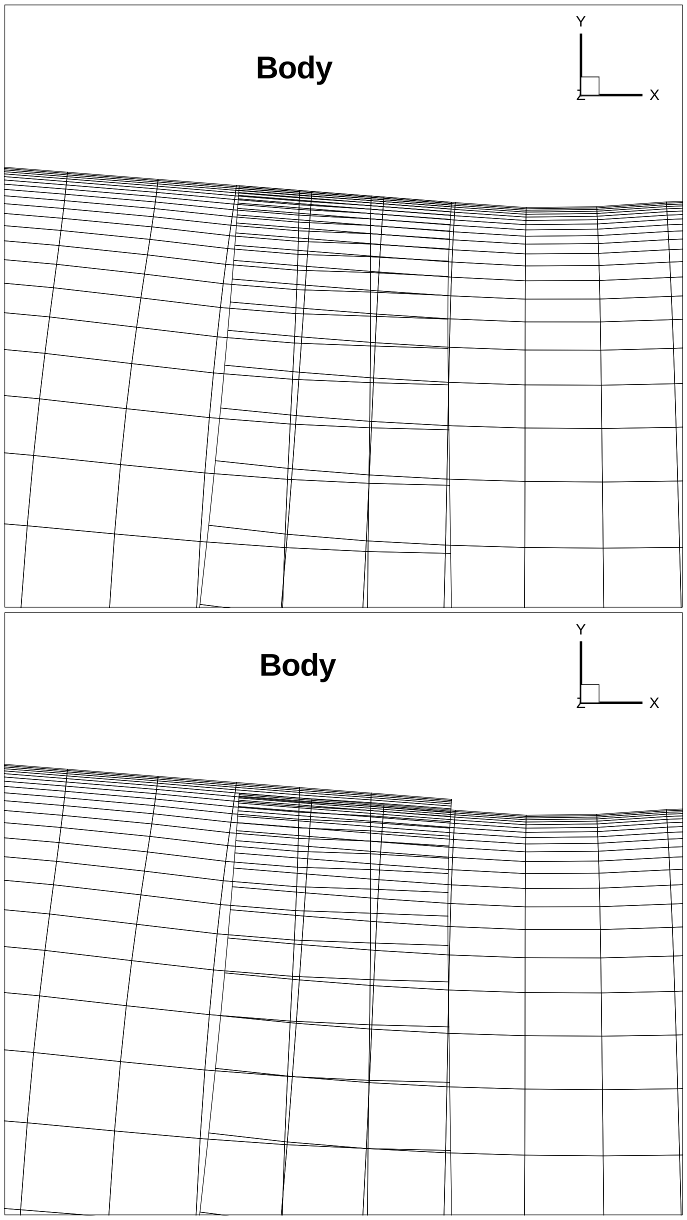


Figure 3.2: Overlap region of two grids on wall surface. Top – Perfect match. Bottom – A mismatch/gap in the two overlap surface on the wall is an example where extra code logic is needed in existing approach to prevent hole cutting failure.

volumes are drastically different from those of the receivers, thus deteriorating accuracy of the interpolation.

This cell selection process can also be regarded as a generalized method for cutting holes around both bodies and some flow features, e.g. a vortex, shock or any refinement grid, indiscriminately and simultaneously. This can be seen clearly in Fig. 3.1 where multiple holes are cut out by the presence of both vortex grids and bodies, and similarly later in Fig. 3.16 for a 3D case. Existing methods use only bodies as the cutter and thus need another procedure to cut holes around these other flow features.

This approach results in a very simple and compact algorithm with only three main subroutines. This is principally due to the absence of any traditional hole cutting using walls as the cutter, hole optimization or tree data structure. The code is completely integrated in the flow solver with no additional input besides those required by the solver. It can also act as a stand-alone code if a main routine is provided. The algorithm is best explained and understood by means of a pseudo code.

In a nutshell, the IHC method routes through every point in the grid system to test and select the best quality cells in a multiply overlapped region, i.e., search for all donor cells of every point in all overlapped regions and keep only the best, leaving the rest as hole points. All points are divided into two types : boundary or interior, with obvious meaning. Boundary points of a grid are first tested before the interior points. The reason for this division will be clear in the next section. Thus, we have the following in the main routine of the solver to start off the connectivity code:

```
DO nRecvGr=1,NG
```

```
   CALL BoundaryConnect(...)
   CALL InteriorConnect(...)
ENDDO
```

It is also possible to first test boundary points of all grids before the interior points as follows:

```
DO nRecvGr=1,NG
   CALL BoundaryConnect(...)
ENDDO
DO nRecvGr=1,NG
   CALL InteriorConnect(...)
ENDDO
```

The grid information needed as input to the subroutines is a bare minimum: 1) dimensions, 2) coordinates, 3) cell sizes (inverse Jacobian) and 4) topology. But cell sizes are calculated in the solver and need not be prepared separately by the user. In addition, special topology such as periodic cut, wake cut, etc. can theoretically be detected in the code. Whether or not those grids with wall boundary are self-protective, i.e., cells on the wall surface are the smallest in the entire grid system, no input is required from the user once a set of grids with reasonable quality are generated. This is thus the most transparent a connectivity code can ever get, and requires no user experience and knowledge of overset method.

In contrast, the corresponding procedure used for the existing approach, which is generally divided into 2 steps, is broadly outlined below. 1) Hole-cutting - a test is performed on every grid point to determine its status of inside/outside of all enclosed bodies (called holes). 2) Expand the resulting 'minimum' hole and find donors for its fringe points - both the expansion and donor search are linked and performed simultaneously. This is more clearly explained in the pseudo-code below.

```
DO ngrid=1,Ng
    DO, for every point of ngrid
        CALL hole-cutting routine (i.e. determine status of point and store in Iblank)
    ENDDO
ENDDO
Next, identify hole fringe points from Iblank
DO, for every hole fringe point
    DO, for every other grid
        Find donor stencil
        Assign a suitability index to donor (could be from hierarchy or priority list; index
            can be distance from surface, cell volume, etc.)
    ENDDO
    If index is 0, mark point as hole and change fringe point (expand hole)
    Store receiver and donor information
ENDDO
```

Naturally, a larger portion of the CPU time is expected to be spent in `InteriorConnect()` because of the larger number of interior points. Detection of non-overlapping grid-pairs as discussed in the next section would tremendously reduce this CPU time.

It is important to note that the IHC approach cannot, by itself, identify points inside the body. It can only identify the hole fringe region surrounding the body. This is, however, sufficient for IHC because firstly, the hole (body) is now completely isolated and cutting it out in a separate process does not pose a challenge and is not costly. But more importantly, as explained in the *On contamination and hyperbolicity* subsection, the points inside the body should not affect the solution and thus are not essential.

It is clear from the description of the two approaches above that the sequence of their processes are opposite of each other. The existing approach is an inside-out process in which the minimum hole is first cut and then expanded, whereas the outside-in process of the IHC locates the hole fringe region first and then cuts the remaining isolated minimum hole. The outside-in process is one reason why location of the body is not required in IHC to locate the hole and why any hole-cutting difficulty at surface is irrelevant.

The simplicity of the IHC approach is also reflected in the fact that there are only three subroutines in the algorithm, namely, `BoundaryConnect()`, `InteriorConnect()` and `FindDonor()`. These will be discussed in the following sections.

3.3.1 `BoundaryConnect()` and `InteriorConnect()`

Both of these subroutines are very similar in content. `InteriorConnect()` is more straightforward than `BoundaryConnect()` however. Other differences will be highlighted in the following discussion.

```
SUBROUTINE BoundaryConnect(...)
DO, for every continuous bound. pt. of grid nRecvGr
   RecvCellVol = 1.e30
```

```
    IF(point is close to wall) ILowOrder = .TRUE.
    DO, for every overlapping or untested grid nDonorGr
        IF(nDonorGr is Cartesian) THEN
            IF(donor Io(n) is beyond 3rd cell from boundary) Io(1) = -1
        ELSE
            Io(n) = iStartCell(n,nDonorGr,nRecvGr)
            CALL FindDonor(...)
            iStartCell(n,nDonorGr,nRecvGr) = Io(n)
        ENDIF
        Check for overlapping with untested grid


        IF(possible donor found) THEN
        IF(donor smaller than receiver) THEN
            Store nDonorGr, donor and receiver indices
            Store interpolation coef.
        ENDIF
        ENDIF
        ENDDO
    Check for overlapping with no mutual cutting
ENDDO
Test one interior point of each non-overlapping grid to check for total embedment
RETURN




SUBROUTINE InteriorConnect(...)
```

```
DO, for every continuous interior point of grid nRecvGr
    RecvCellVol = InvJacob(i,j,k,nRecvGr)
    DO, for every overlapping grid nDonorGr
        IF(nDonorGr is Cartesian) THEN
            IF(donor Io(n) is beyond 3rd cell from boundary) Io(1) = -1
        ELSE
            Io(n) = iStartCell(n,nDonorGr,nRecvGr)
            CALL FindDonor(...)
            iStartCell(n,nDonorGr,nRecvGr) = Io(n)
        ENDIF


        IF(possible donor found) THEN
        IF(donor smaller than recv. .or.  forced donor)
            Store nDonorGr, donor and receiver indices
            Store interpolation coef.
        ENDIF
        ENDIF
    ENDDO
ENDDO
RETURN
```

1) *On using cell size to cut holes around solid bodies and refinement grids.* For every point of a grid, all other grids are searched for having any potential donor cell for this point. (An interpolated boundary point is one whose boundary type is not specified and not updated in the flow solver. A donor cell is one that

encloses this point). In the case of multiple donor cells, the one with the smallest cell volume, i.e., inverse Jacobian, may be selected as the optimum donor. Other geometric criteria, e.g. cell aspect ratio, orientation etc, may be important in a high gradient region. They are, however, not as straightforward to use and will not be considered here (see [6] for the use of cell difference parameter CDP). The cell volume associated with a boundary point is assigned infinity regardless of its original value, since this point must always have a donor cell even if its cell volume is already the smallest among all possible donor cells. In contrast, cell volume associated with an interior point is its original value. If this value is already the smallest, then no donor cell is qualified and this point is a normal field point.

2) *On non-overlapping grids and why boundary points of a grid are tested first before interior points in order to eliminate unnecessary tests.* Many component grids are non-overlapping and far apart in most applications. One can take advantage of that to eliminate many of the tests on the much more numerous interior points by testing boundary points first. This tremendously reduces CPU time spent in testing interior points. Non-overlapping grid-pairs can be detected from the following two conditions: i) none of the boundary points on the grid being tested (grid A in Fig. 3.3 left) has any potential donor cell from the partner grid (B), and ii) the partner grid (B) is not fully embedded inside the grid being tested (grid A) (embedment is shown on the right in Fig. 3.3). Only one point from grid B needs to be checked for condition ii) once i) is satisfied, since satisfying i) implies grid B is either completely outside or completely embedded in grid A. This can be theoretically any point but preferably close to the midpoint of the grid since boundary points of an embedded grid may be so close to

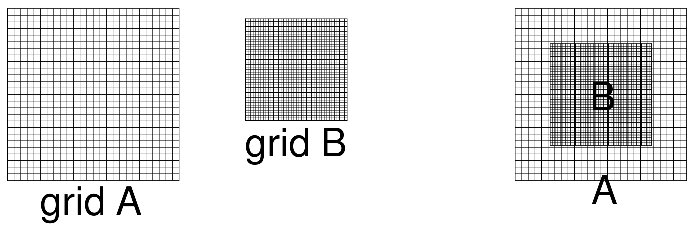


Figure 3.3: Non-overlapping grid-pair and embedded grid-pair.

the boundary of grid A that an outside status is returned because of insufficient stencil size (see *Stencil size* of next section). This condition can be easily determined for a Cartesian grid A (upright or inclined). For a curvilinear grid, a donor search must be performed. If both conditions are satisfied, the interior points of the test grid (A) will not be tested and `Ioverlap(nRecvGr,nDonorGr)` is assigned 0 (1 for overlapping pair). The matrix `Ioverlap(nRecvGr,nDonorGr)` is symmetric and the diagonal elements are irrelevant. Hence, only the upper triangular elements need to be determined. The lower triangular elements must also be filled because they will be accessed in the subroutine.

Since many boundary points need interpolation anyway, tests on these points first yield the additional benefit of detecting non-overlapping grids. Without this, the increase in cost with number of grids, NG, would have been boundless (for the same number of points). Non-overlapping grids help keep the cost flat with increasing NG for interior (but not boundary) points.

Interior points of two overlapping grids, A and B, that share the same part of a solid wall (this can be either O-H, C-H or 1-wall grid) as sketched in Fig. 3.4 do not need to be tested for cutting each other's hole since it is assumed here that cell quality (i.e., volume) of each grid is good enough for the region each covers

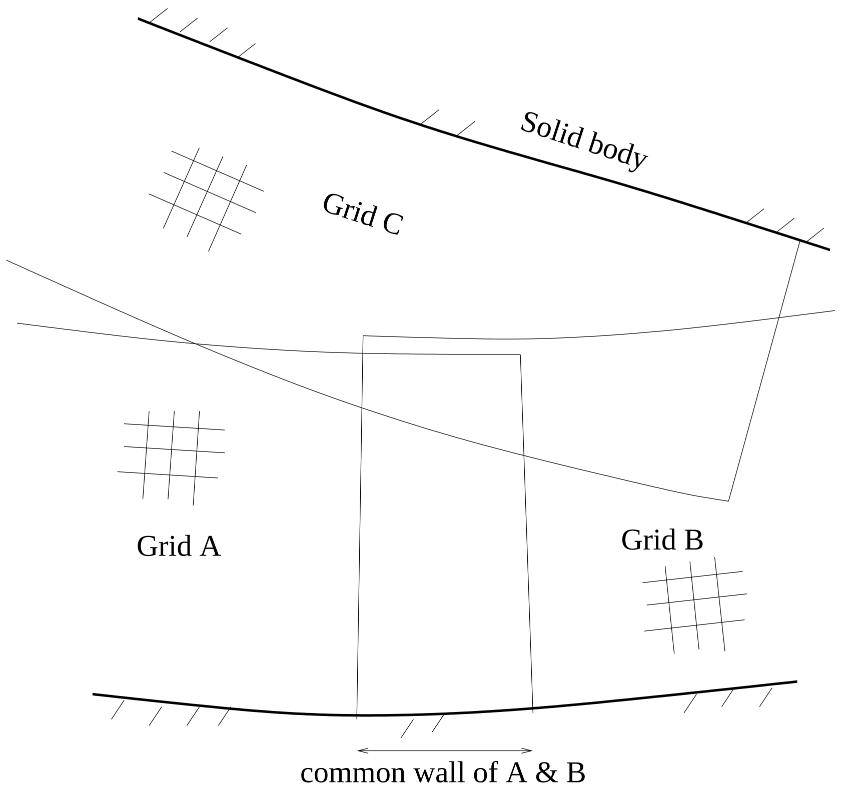


Figure 3.4: No mutual cutting between grids that share part of a wall (A and B). Possible cutting between A and C, or B and C.

even though quality of the neighbor grid may be even better in the overlap region. This is done to both expedite the connectivity process and reduce the number of interpolation points since, as explained later, interpolation is an expensive process especially if it is high-order. `Ioverlap(nRecvGr,nDonorGr)` is assigned -1, not 0, in this case because a zero value copied to the lower triangular element would later cause its partner grid `nDonorGr` to mistake this grid pair as non-overlapping and ignore even the boundary point interpolation. The way to detect such 'no mutual cutting' overlapping pair is to identify the grids of the donor cells for the boundary test points just above the wall surface.

For an O-H or C-H test grid A, condition ii) can be relaxed so that the partner

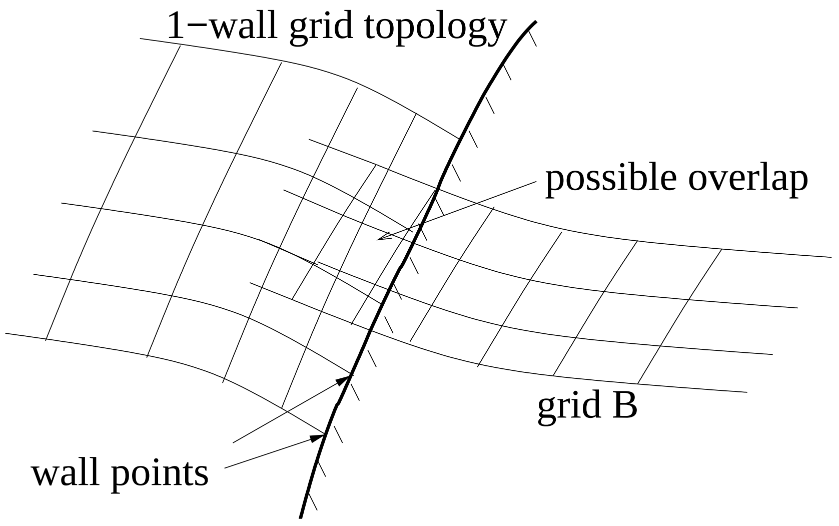


Figure 3.5: Wall points on one-wall grid topology must also be tested for having potential donor cells, not for interpolation, but for detection of overlapping grids coming in from the wall.

grid (B) is not fully embedded inside the outer boundary of grid A, since the wall and wakecut boundary are inside this outer boundary. It is not necessary to test points on the wall and wake cut. For a grid with one-wall topology, however, points on the wall are tested, not for interpolation purpose, but for the detection of the unusual event schematically shown in Fig. 3.5 in which it overlaps with the partner grid (B) coming in from the wall boundary. This grid (B) could be a long and skinny vortex grid. Should this happen, test on interior points will be carried out.

3) *On continuity of test points (boundary or interior) in order to shorten search path without using tree data structure.* The starting cell for a search in a particular grid is either i) the donor cell of the previously tested point (if there was indeed a donor cell), or ii) the last cell next to the grid boundary in the previous search path before the search steps out of that boundary

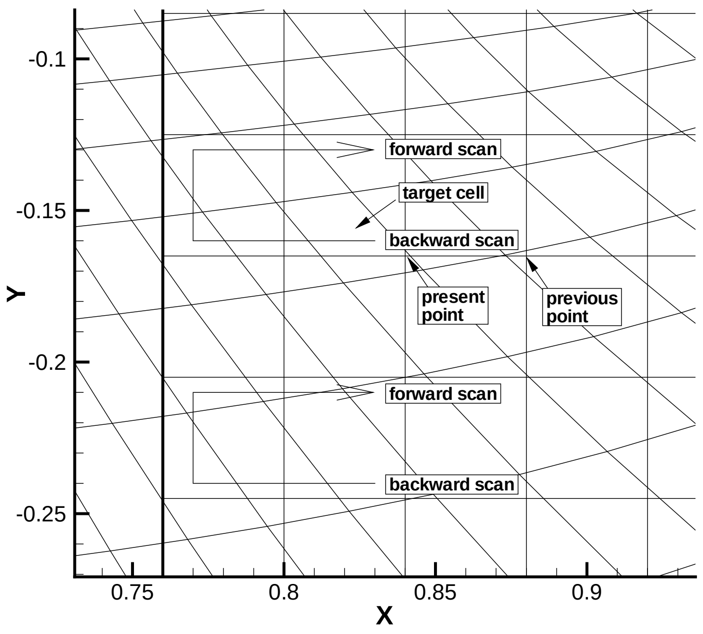


Figure 3.6: Illustration of alternating forward / backward scan to ensure continuity for interior test points.

if the point is outside the grid (in which case indices of this last cell are made negative to indicate unsuccessful donor search). Due to the proximity of the present and the previous test points in case i) if test point continuity is maintained, the donor cells of the two points are also in close proximity. This would considerably shorten the search path. No tree data structure whatsoever is required to expedite the search. The three indices of the starting cell are stored in `iStartCell(n,nDonorGr,nRecvGr)` where n=1,2,3. This storage is required for each grid pair and the matrix is not symmetric in `nDonorGr` and `nRecvGr`.

Lacking a previous point for guidance, the first test point of a grid would arbitrarily start the search in other grids from the cell with the lowest indices, i.e., (1,1,1). This naturally results in a long search path but is limited to only the first boundary and first interior point of every grid in the first time step. Employment of any accelerated search technique here (e.g. some tree data structure) is not really warranted, as this would also unnecessarily complicate the algorithm for a negligible gain in efficiency.

Continuity of interior points can be achieved quite effortlessly by running the I,J,K DO loops for the test in alternating forward and backward direction as symbolized by the arrow in Fig. 3.6. For a 3D problem scanning in the K,J,I direction sequentially, the K,J and I loops are swept backwards for odd istep, even K+istep and odd J+K+istep respectively. Here istep is the time step iteration number.

Continuity of boundary points is much more difficult to maintain. Nevertheless, it can still be achieved for the O-H (3 sides), C-H (5 sides) and one-wall (5 sides) grid topology with additional logic in the code. A free grid without any solid wall, however, will have one discontinuity in traversing through all the boundary points. Occasional discontinuity in test points would not affect the speed of the algorithm appreciably, but the resulting long search path at the discontinuity may pose difficulty along the path if it has to go through some complex topology.

In order to avoid test repetition at the eight corner points and twelve boundary lines of a grid if each of the six sides is scanned from first to last point, the dimensions of the two sides normal to each of the I, J and K direction used in the test are (JMAX-2)*(KMAX-2), IMAX*KMAX and IMAX*(JMAX-2) respectively as illustrated in the diagram in Fig. 3.7. The savings achieved both here and during solution interpolation is not negligible especially for long and

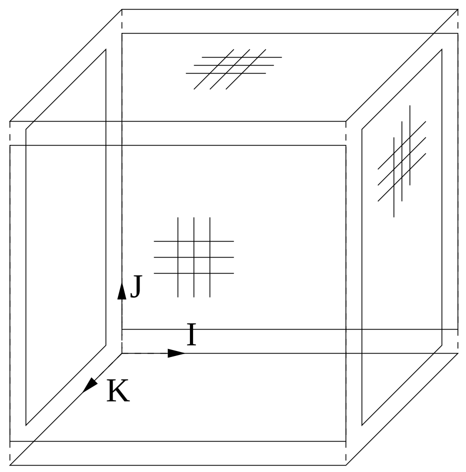


Figure 3.7: Test repetition at boundary lines and corners is avoided by using interior dimensions.

skinny grids (e.g. vortex grids) and/or when the disproportionately expensive high-order interpolation is used (as will be explained later).

4) *On contamination and hyperbolicity.* All test points, whether they fall inside or outside a solid body, are treated indiscriminately. If a point inside the body finds a donor, it will be interpolated even though neither it nor the value it gets is valid. This interpolation is of no consequence and does not contaminate the rest of the grid because invalid points are always fully insulated from the normal field points by the layers of hole fringe points that will be interpolated from valid donor cells from other grids (as will be explained in the last two paragraphs). In view of the inconsequence of this action and the very small anticipated number of such points, additional code logic to avoid this is not needed. The grid of these donor cells is usually, but not necessarily, the boundary conforming grid of the body inside which the above mentioned points fall. The distance of the first fringe layer from the surface depends on the relative cell volumes of the receivers and the donors and is always optimum.

Grids are almost always designed such that cells are smaller the closer they are to the wall surface. This ensures the availability of donor cells for the hole fringe points and thus that the 'hole-cutting effect' of the wall is felt. In the unusual situation in which even the smallest donor cell on the surface is bigger than all of the receiver volumes, hole fringes can still be formed by the criterion that cells with index in the direction normal to wall less than or equal to `jHoleImmune` (user specified) must be donors regardless of volume (as exemplified Fig. 3.8 where airfoil grid is donor). Such a grid system, however, has doubtful overall quality. `jHoleImmune` also serves a similar purpose for a receiver grid. Points with index in the wall-normal direction less than or equal to `jHoleImmune` can not be receivers regardless of volume (see Fig. 3.8 again where the airfoil grid is now receiver) and are not being tested at all. This enforces the sanctity of a wall boundary condition. The value of `jHoleImmune` can be different for this forced non-receiver and the previous forced donor case.

Immunity against being tested and becoming receivers also helps reduce the number of test points and CPU time especially for viscous grids where a large number of points in the boundary layer can be immunized. Indeed, with a much larger `jHoleImmune` for viscous grids, the cost of connectivity is not very much higher than that for inviscid grids that have much smaller number of points. A note of caution is appropriate here. `jHoleImmune` is not the `OFFSET` variable in PEGASUS 5 or similar parameters in other codes for hole expansion purpose. Hole expansion is irrelevant in this approach.

So far, all holes cut are assumed to be away from the surface. That is what `jHoleImmune` enforces. However, it is sometimes more convenient for a near-body (untrimmed) grid to cut through the body and build another (or several)

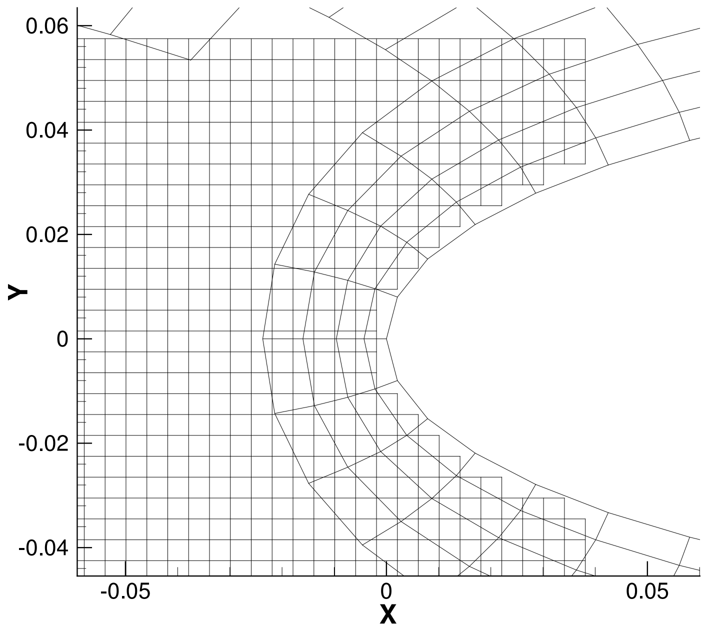


Figure 3.8: An unusual situation in which the smallest cell on an inviscid grid wall is larger than all receiver cells. `jHoleImmune = 5` immunizes the first 5 grid lines from the wall against receiving or force them to be donors.

partner grid(s) for the wall that it misses. Examples include collar grids and grids that cover secondary surface features. Untrimmed grids and their partners can be topologically represented by the schematic shown in Fig. 3.9. This topological arrangement also applies to surface refinement grids with no cut through the body (Fig. 3.10) - in fact this is a special case of Fig. 3.9. `jHoleImmune` for the untrimmed grid as a receiver must be set to 1 when testing interior points against its partner grids so that even its wall surface can feel the presence of the missed wall. The automatic and unambiguous detection of untrimmed grids and their partners is difficult. They can be visually identified, however, and specified manually.

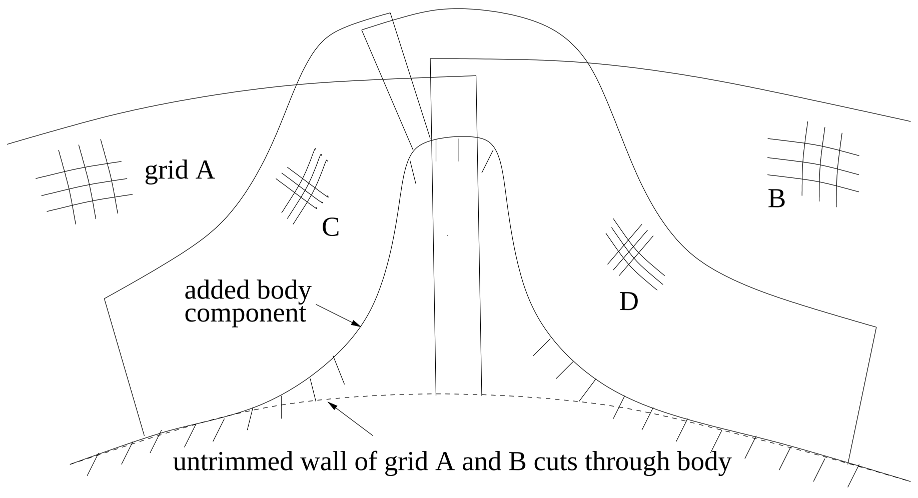


Figure 3.9: Sketch of multiple untrimmed grids (A,B) with multiple partner grids (C,D).

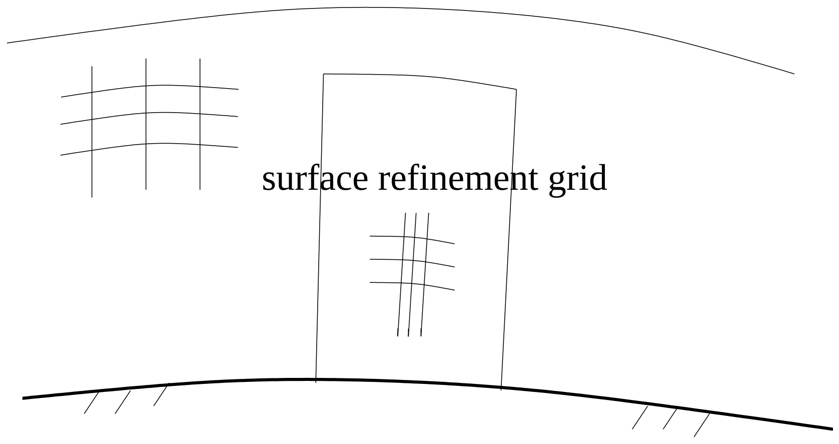


Figure 3.10: Similar treatment for surface refinement grid as for untrimmed grids.

Theoretically, contamination of normal field points by the invalid hole points is avoided because of the hyperbolic nature of the fluid equations in time. Hyperbolic solutions are 'local' and have finite speed of information propagation. This implies the region of influence of any invalid point is limited and depends on the time step size Δt specified by the user. The hole fringe layers that separate the field points from the invalid hole points act like a cushion of insulation against contamination. The thicker the layers, the longer time the invalid solutions need to cross it in order to contaminate. If the distance traveled by the information from any invalid point at the end of Δt is less than the thickness at the thinnest section of the fringe, contamination would be theoretically impossible since the solutions at all fringe points (some of which are already contaminated) would have been replaced with valid solutions from other grids through interpolation. This is why blanking out hole points in the solver to avoid contamination is not necessary. This situation with an overset hole is somewhat similar to a black hole in which outside information can fall in but information inside the hole can never escape and be felt outside. This remains true as long as the numerical schemes employed to solve the equations do not violate the propagation properties dictated by the mathematics of hyperbolic PDE.

Furthermore, because of the above reason, there is no difference between a hole point inside the flow field and one that is inside a body. Hence the exact shape of the wall surface has lost its significance as a hole cutter and any surface grid imperfection can be forgiven. This tremendously reduces the complexity of the connectivity algorithm since using walls to cut holes is no longer required. Thus the name Implicit Hole Cutting. However, for the purpose of calculating residuals and plotting contours, as mentioned before, hole points inside a body must be identified.

The above theory results in a rather unusual situation in which fringe thickness, information propagation speed and more importantly Δt are now theoretical determinants of possible contamination. Fortunately, first-order checks of many cases indicate that commonly encountered Δt is usually well within a contamination limit. Nevertheless it is advisable to make frequent checks whenever possible.

Numerically, the issue of contamination can be explained as follows. Methods for time integration can be classified as either explicit or implicit. From the mathematical behavior of partial differential equations, one can say that an explicit method mimics the 'local' behavior of hyperbolic equations while an implicit method mimics the 'global' behavior of parabolic equations. Both methods, however, have been used successfully for either equation despite obvious physical violation for the wrong combination of method and equation. This can be explained by the fact that distant information does not contribute much to the solution at a point, whether it is an implicit method applied to a hyperbolic equation or an explicit method applied to a parabolic equation. The inclusion or exclusion of this information in the numerics contributes very little. Thus, it is safe to say that an implicit method applied to a hyperbolic equation for a grid with a hole does not significantly, and theoretically should not, cause contamination especially if Newton subiterations are used, provided the fringe thickness condition in the previous paragraph is satisfied. Of course, for this to be true, values at the invalid points must not be very large. Otherwise even a small fraction of it can make a significant contribution (or contamination) to a distant point. This can usually be satisfied by the finite values assigned as the initial condition. In addition to this, attention must also be given to the situation in which only a few points span across a hole caused by a thin (or small) body, i.e. where grid resolution difference is too great.

3.3.2 Subroutine `FindDonor()`

This subroutine is called by both `InteriorConnect()` and `BoundaryConnect()` to find donor cells.

```
SUBROUTINE FindDonor(...)
DO
    Get starting cell (previous donor) Io(n) for the search
    Cross+dot product for inside/outside cell test
    IF(inside of cell) jump out of DO loop
    Check for moving back to a previously tested cell
    Check for cut across concave boundary
    Check for crosses into solid body
    Check for cut across O- and C-grid periodic boundary
    IF(point is outside boundary) THEN
        negate Io(n)
        GOTO the end and return
    ENDIF
ENDDO

21 IF(point is beyond first interior point .and.  not LowOrder case) THEN
    negate Io(n)
    GOTO the end and return
ENDIF

Get coordinates of stencil points for interpolation
Calc. initial guess of dIo(n) (use linear interp.) or set to .5

niter = 0
DO WHILE(residual > tolerance)
    niter = niter+1
    Solve for interp. coef. dIo(n) with Newton iter.
    IF(dIo(n) outside range .and.  niter.ge.3) THEN
        Move to neighbor and GOTO 21
    ENDIF
ENDDO

IF(point is beyond second interior point .and.  not LowOrder case) THEN
    negate Io(n)
ENDIF

RETURN
END
```

-*A quick first-order inside/outside cell test.* This test of whether a point is inside or outside a cell is required in every step of the search. A quick cross and dot product test suffices for a first-order estimate. This is much faster than

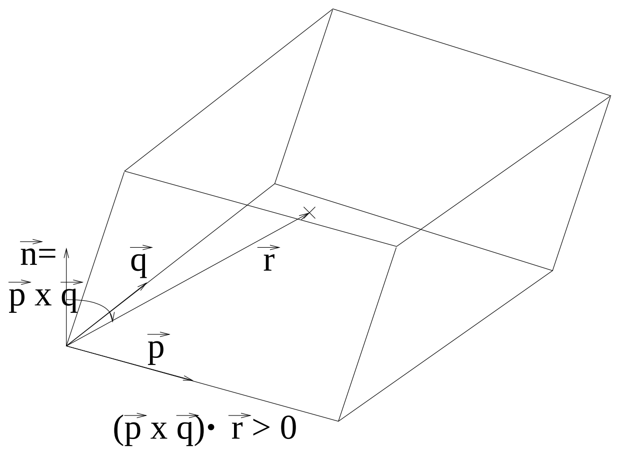


Figure 3.11: A first-order cross and dot product test for the inside/outside status of a point.

calculating interpolation coefficients using Newton iterations though not very accurate for points close to the boundary (if the boundary face is highly curved). For a point to be inside a cell whose boundary faces are initially assumed to be planar, all six dot products of the inward normal vector at any point on each face, $\overrightarrow{n_i}$ in Fig. 3.11, and the vector between the point P and the base of the inward normal vector, $\overrightarrow{r_i}\cdot(\overrightarrow{n_i}\times\overrightarrow{r_i})$, must be positive. This is equivalent to requiring the point to lie on the 'in' side of each boundary face. If the point is outside, the direction of the next move in the search is indicated by the faces with negative dot products. This test requires that the three axis in computational space follow the directions of the right hand rule. Otherwise, it would lead the search away from the donor.

-*Indefinite search.* The possibility of a search going in a loop indefinitely, i.e., repeatedly coming back to previously tested cells, arises because of the simplifying geometric assumption underlying the above test. This can happen when a point is close to a cell boundary face that is not exactly planar and the

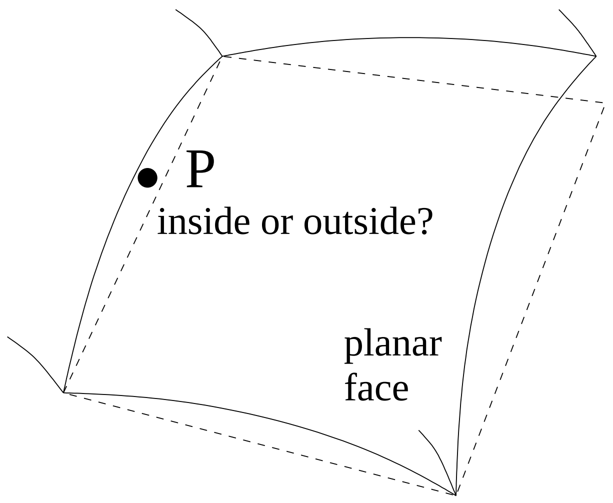


Figure 3.12: Inside/outside status of a point close to a truly curved cell boundary can not be determined from cross and dot product test.

test does not reliably indicate its inside/outside status (Fig. 3.12). This can be solved by storing the current test donor cell indices after a preset number of searches. Every subsequent test cell would be checked against this for any repetition. Once repetition is first detected, a donor is assumed found regardless of its correctness. The subsequent cubic coordinate interpolation will be able to accurately determine the correct status and move accordingly. Checks are also easily provided for search across the wake cut in O-H/C-H grids and for stepping out of the grid boundary.

-*Grid topology*, besides indicating physical and interpolated boundaries, is required for a search in determining the action to be taken when it steps out of the grid boundary in computational space. Four types of topology are common and often sufficient for aerospace applications : O-H, C-H, free and one-wall. The wake cut boundary of O-H and C-H grids would move the search back into the grid. Other topology types such as grids with opposite or adjoining walls can be accommodated without significant alteration to the logic of the code.

-*Stencil size.* The Hermite cubic monotone interpolation employed in the solver for flow solutions requires a 6-point stencil in each direction, or 5 points for each of the two center points where first and cross derivatives are required. This derivative is required to be at least 3rd-order (4-point) in order to achieve full accuracy of the cubic interpolant. A larger 5-point 4^{th}-order stencil is used for reasons of symmetry. Unlike solution reconstruction in the solver, a 4^{th}-order compact scheme can not be used for interpolation as explained in chapter 2. A 6-point stencil implies that the first two cells from each grid boundary are not qualified as donors. However, the second cell must not be ruled out after the first approximate test since it might be so close to the third cell that the more accurate test later may reveal that the point is indeed in the third cell. Note that for coordinate interpolation, we use a smaller 4-point tricubic stencil for the reason of consistency with the metrics in the solver as mentioned before.

-*Implicit interpolation.* This problem of mutual coupled interpolations between two boundary points with 6-point stencil and minimum overlap (Fig. 3.13) is not as severe as in linear interpolation with a 2-point stencil because it is assumed that the first and the last point in the 6-point stencil do not contribute significantly to the interpolated value. We can thus justifiably avoid the more expensive implicit interpolation, easing the already very costly interpolation that we employ. Furthermore, this problem does not exist when one of the two points is a hole fringe and the other a grid boundary point if the latter is interpolated first. This is due to the fact that solutions at the hole fringe points which lie on the first fringe layer have been updated by the solver and are valid to provide interpolation to the grid boundary point. Likewise, neither does the problem

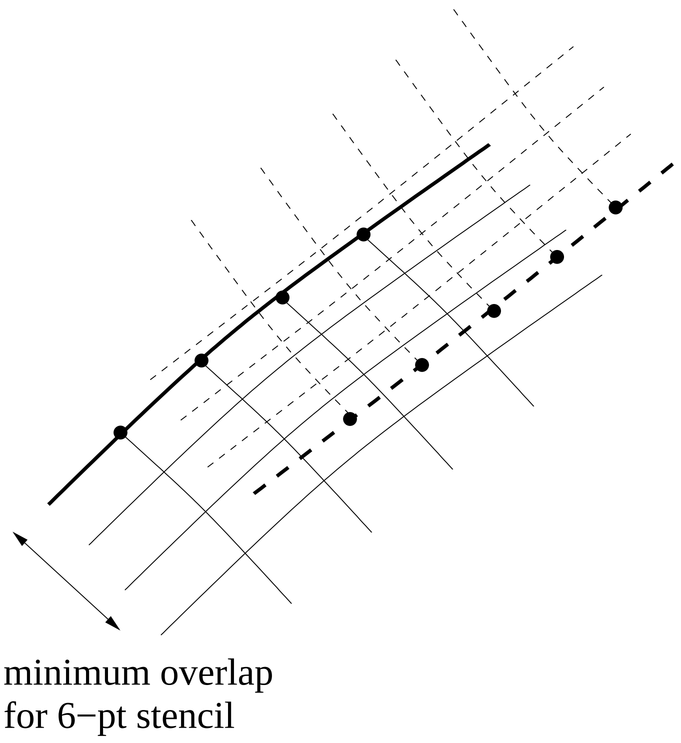


Figure 3.13: Implicit interpolation for two outer boundary points with minimum overlap and 6-point stencil is not used.

arise when both of the points are hole fringe points. Thus in the solver, all grid boundary points are interpolated first before the hole fringe points.

-*Low-order linear interpolation* is unavoidable for the situation illustrated in Fig. 3.14. The boundary points immediately above the wall side of a grid will have donor cells with insufficient stencil width. Normally these cells would have failed the test for this reason. But in this situation they must be accepted and a lower-order 4- or even 2-point stencil be employed here. These points are identified in subroutine `BoundaryConnect()` as J<n (where n is a small user specified number, e.g. 4, and J is normal to wall) for O-H and one-wall topology.

-*Initial guesses of the interpolation coefficients* for Newton iterations can be

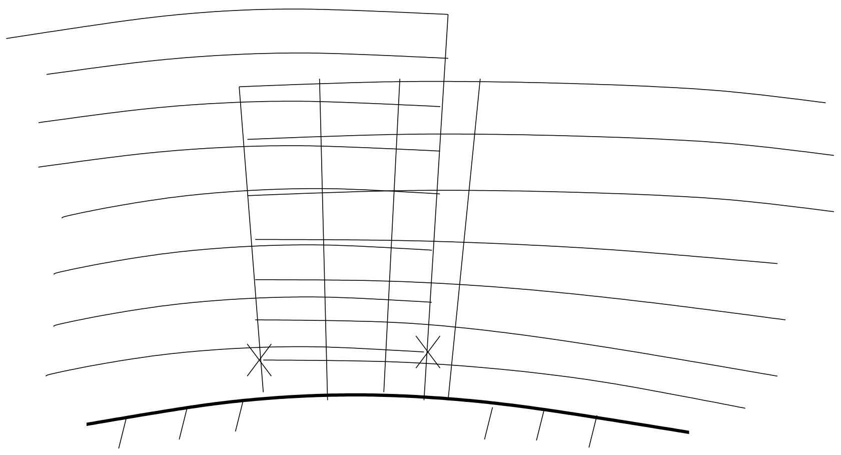


Figure 3.14: Small stencil low-order interpolation is unavoidable for points close to wall marked X.

either approximate solutions from trilinear interpolation or simply set to 0.5, i.e., midpoint assumed. Tests have shown that the former is more efficient for high-order interpolation since it provides better estimates for fewer Newton iterations which consume a large percentage of CPU time. However, the latter is favored for coarser multigrid levels when linear interpolation is used.

-*Concave boundary.* Detection of a concave boundary is important when the search steps out of the boundary. It is also one of the most difficult tasks in this approach. Failure in the detection might result in a point being mistakenly assigned as outside the grid when it is actually inside. Visual identification usually works but is a crude method.

-*Indefinite toggling.* Even with accurate interpolation, it is still possible for the final donor cell to toggle indecisively between two neighboring cells if the point is very close to the border. It was decided that a move to the neighbor for an out of bound point ($\mathbf{s}$<0. or $\mathbf{s}$>1.) is made only at the end of the third Newton iteration since it usually takes about 3 iterations to converge.

-*Multigrid.* In a multigrid situation where every alternative grid line is successively removed, the third cell becomes the second (first) cell in the first (second) coarser level and the stencil must be reduced to 4-point (2-point).

-*Unstructured grids.* The main principle behind Implicit Hole Cutting, i.e., using cell size comparison to locate hole fringe and cut hole, is valid whether the grids are structured or unstructured. The implementation, however, is different for the latter in the donor search because of the unstructured coordinates and volume data arrays. Cross and dot product in/out test for stencil walking and wall boundary crossing in `FindDonor()` would have to be modified due to loss of data structuredness.

3.4 Advantages of 'IHC'

The new approach used in 'IHC' results in numerous significant advantages not realizable in the conventional approach. Some of these are summarized below:

a) No need to specifically cut holes around arbitrary bodies. This is the most difficult step in the conventional approach.

b) No hole cutting failure due to imperfections/difficulties at the wall (e.g. mismatch/opening, thin bodies, surface discontinuities, etc.) because these surface features are irrelevant in a cell comparison process. In fact cells in the vicinity of the wall are never subject to the comparison process.

c) No hole optimization step needed. Holes that arise at the end of the process are always automatically optimum because the criterion for cell comparison and optimization are the same.

d) No need for user to manually order grids by priority or group by hierarchy.

e) No extra step needed to cut holes around wall-less refinement grids (e.g. PEGASUS 5 Level 2 Interpolation). Holes around bodies and refinement grids are formed indiscriminately.

f) Many points in and near the boundary layer can be immunized, further speeding the code for viscous grids.

The most significant of these advantages, however, is undoubtedly the elegance and simplicity of this approach which results in a very compact algorithm. The algorithm is implemented with less than a thousand lines of Fortran90 code, or about an order of magnitude smaller than that using existing approach.

3.5 Parallel Implementation

Load balancing in the parallel execution of overset connectivity codes using existing approach is much harder than that of the flow solver due to the unknown and varying (in problems with relative motion) number of fringe points and the amount of work needed to find all donors. This problem is critical for cases with moving components, such as rotorcraft, since the connectivity needs to be updated every time step and the CPU time for connectivity is a significant fraction of the total time. While the number of points in a grid is a reasonable criterion for load distribution in a solver, it is not a valid prediction of connectivity cost. Since hole cutting and donor search are the most expensive operations in conventional connectivity, the most accurate way to balance load is to use the number of cells on the hole-cutting surface and the number of IGBPoints that need interpolation as the criteria for work distribution. They are considered the smallest work units in connectivity. This is, however, also the source of most of the difficulties in the connectivity algorithm.

The parallel implementation of the Beggar code is detailed in [8] in which four phases are implemented in increasing order of efficiency. In the first three phases, the connectivity code is executed in the background on separate processors with increasing balance of the load distribution. Only the last phase is considered a truly parallel attempt, executing both the solver and connectivity on the same processors and taking the smallest work units into consideration.

Several references, e.g. [74], [75], are available on the parallel implementation of the adaptive OverFlow-D code. In [75] a dynamic load balance scheme is outlined in which the work load of donor search and solver are balanced concurrently across the processors, using the number of IGBPoints and grid points respectively for this purpose. While the number of IGBPoints is allowed to vary in the long term because of moving holes, hole cutting is not considered in the balancing criterion.

Since PEGASUS [6] is stand-alone and not associated with any particular CFD code, its parallelization can not be combined with that for a solver. Furthermore, it is not designed for problems with moving components.

Load balancing in Implicit Hole Cutting, on the other hand, is considerably simpler. Since the two main loops in 'IHC' are over grids, work load can be distributed grid-by-grid just as in the coarse-grain parallelization of the solver. This is a direct consequence of the approach advocated in the 'IHC' algorithm. In a coarse-grain implementation, all coordinates and cell volume arrays, unlike the solution arrays, are stored in every processor. This increases memory requirement for these two arrays but simplifies the job since no massive communication is necessary. This is however not recommended for fine-grain parallelism.

If the parallelism is implemented using Co-Array Fortran, cell volume and coordinates arrays need not be stored in every processor since the one-sided communication (no hand-shaking required) to access remote memory where the donor cell resides is easy to implement using co-array. Typically only a small number of donor cells are needed.

The CPU time of the preceding time steps for every receiver grid in the subroutines `BoundaryConnect(...)` and `InteriorConnect(...)` can be used as the criterion to distribute workload. This is the most accurate coarse-grain load balancing criterion and it works equally well on heterogeneous clusters. Some time saving features are more costly in parallel mode (e.g. double work in the detection of non-overlapping grids for some grid-pairs in the `BoundaryConnect(...)` loop) due to their sequential nature. Iterations in the `InteriorConnect(...)` loop are, however, independent. Every work package in these two loops can be assigned to different processors every time step since each processor has all the information it needs to work on any package. Load balancing then becomes straightforward and can even be combined with that for the solver.

3.6 Examples

Compared with that of a flow solver, the validation of a connectivity code is much simpler and less uncertain. Good correlation of a few sets of results from a solver with experimental data for a particular flow condition does not automatically verify the correctness of every numerical scheme in the code for all flow conditions. Confidence in the solver to produce acceptable results is gradually gained over time after repeated applications by a wide group of users. This is

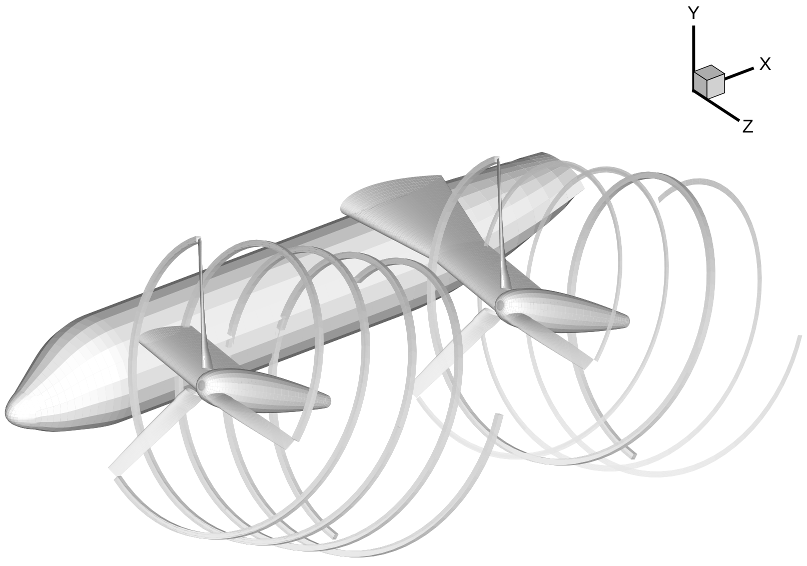


Figure 3.15: Half-span QTR-like model.

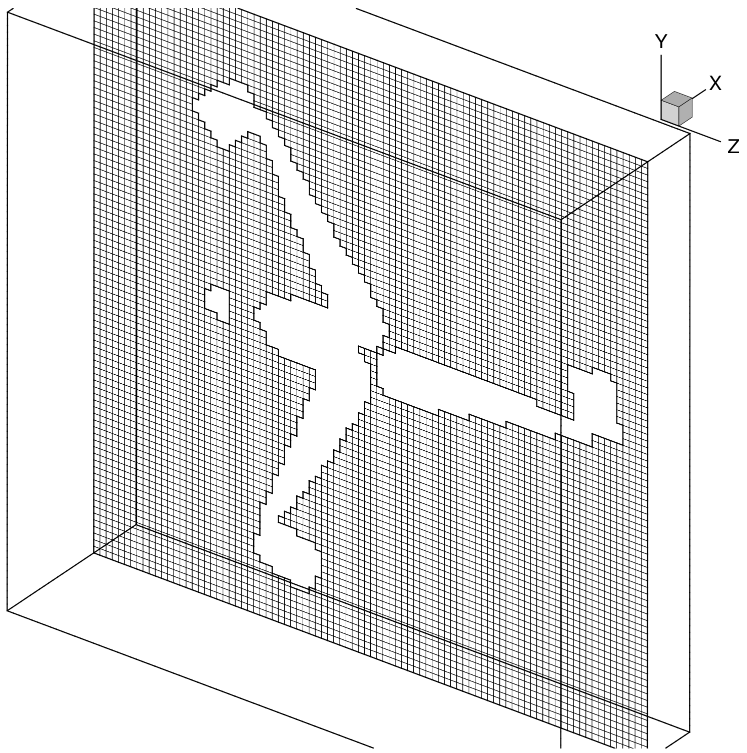
also one reason why the lifespan of a software is many times that of the hardware that it runs on. It is not inconceivable that some of the numerical schemes are not suitable for certain flow conditions or body configurations and that a time-tested code produces disappointing results under these conditions.

The verification of a connectivity code, on the other hand, is much more certain if the body geometry and configuration is general enough. It is easy to determine visually if a hole is incorrectly cut or if a receiver point has the wrong donor cell. In other words, the range of conditions within which the results are right or wrong is more narrow.

Some 2D test problems on blade vortex interaction will be given in the next chapter on vortex tracking (see Fig. 3.1 for an example). Only 3D test problems will be considered in this section. The first example is a Quad Tilt Rotor (QTR) like configuration with blades, nacelles, nested front vortex-tracking grids, wings, fuselage and background grids (Fig. 3.15). The grid system consists of 39 grids and about 3.1M grid points for a half-span model with moderately coarse spacing (except for inner vortex grids). Figure 3.16 (top) shows a constant stream-wise section and the holes in the background grid cut around the rotors, a wing and the vortex grids. The bottom frame shows the negative 3D image of the same holes. These holes are appropriately cut and serve as verification of the validity of this new approach.

Required CPU time on an Alpha 21264 processor at 667 MHz to generate all necessary parameters for the solver using the more expensive cubic interpolation was about 35 seconds. The use of linear interpolation reduces the time by about $20-25\%$. This significant reduction is not surprising considering the expense of cubic interpolation as explained earlier.

Figure 3.17 depicts three snapshots of a movie showing holes in a background grid for a half-span QTR with vortex grids. The blocks at the tip of the blades are refinement grids added to increase the resolution in the region immediately beyond the blade tip grids, since the background grid has a rather coarse spacing of 0.45 c_{tip} and the flow field at the tip region is intense. Each of these grids contains only about 17K points but has high grid density especially in the radial direction in which the density matches that of the blade grid. Hence, two types of refinement grids are present in this example, blade tip refinement and vortex grids. It should be noted again that the figure does not represent any aircraft components. It shows only the holes caused by the components in the background grid shown at the top of the figure. This grid system and connectivity



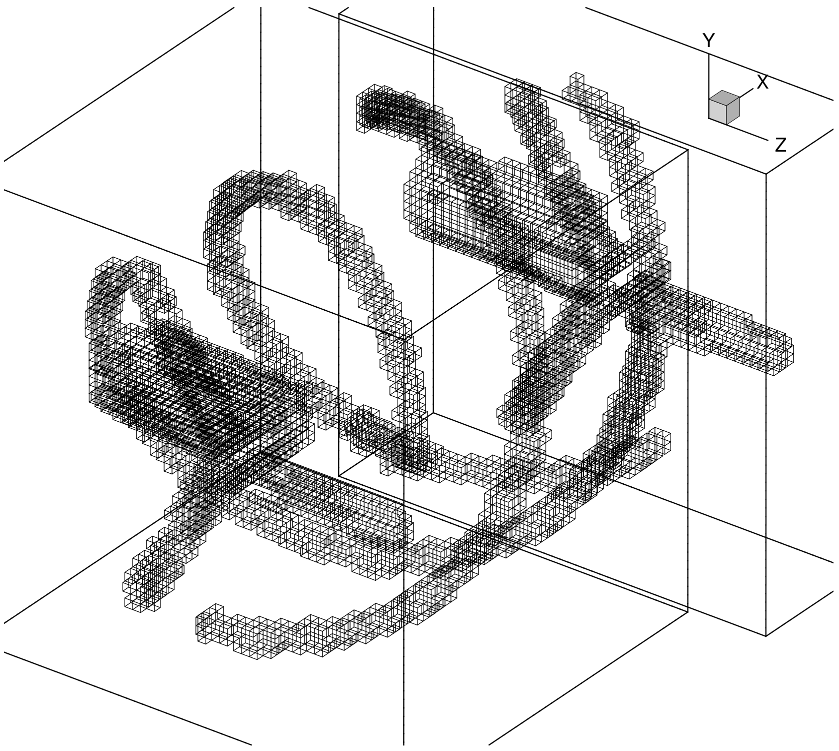


Figure 3.16: Holes caused by blades, vortex grids, wings and engine nacelle (for the bottom figure).

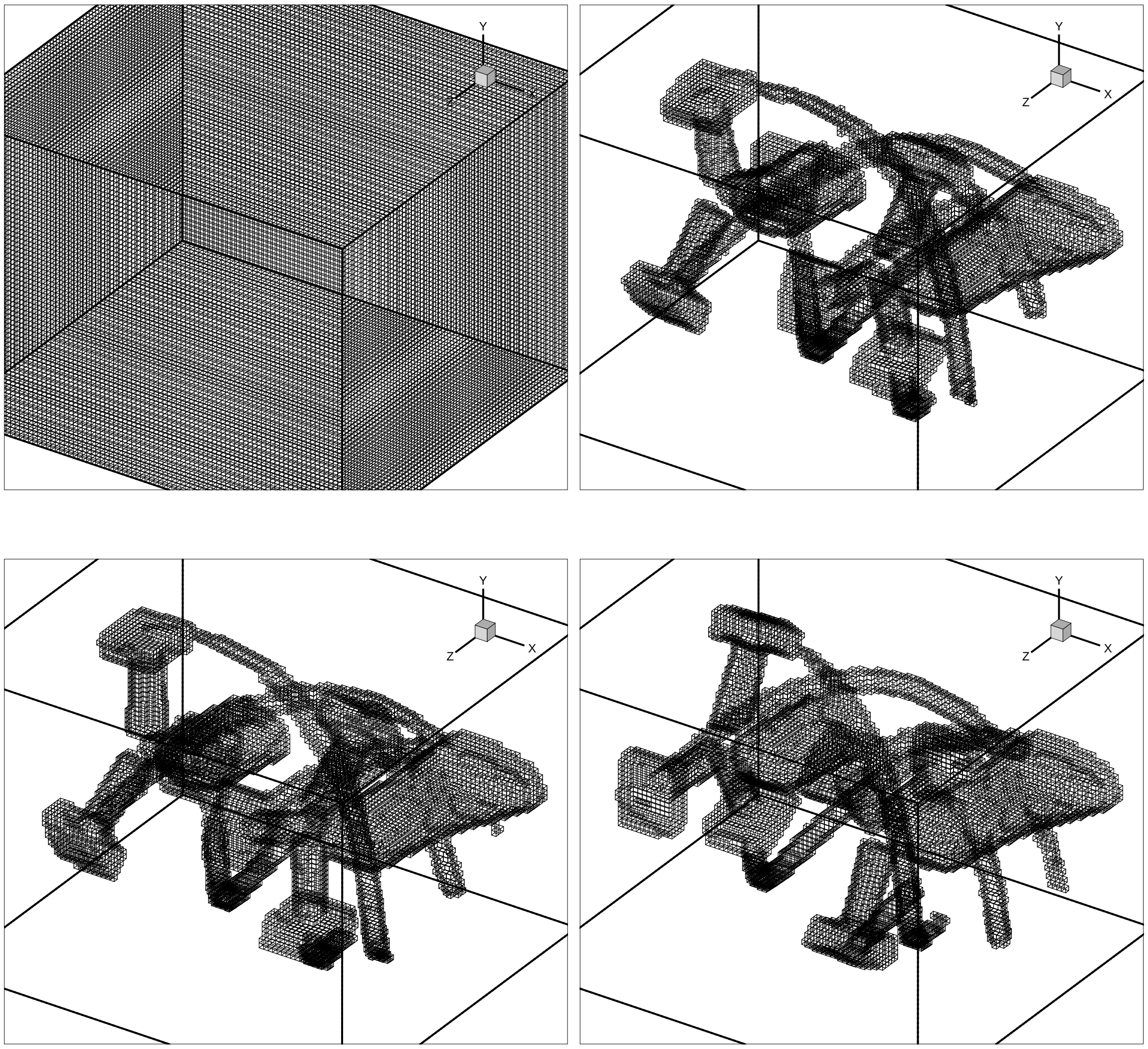


Figure 3.17: Three snapshots showing holes in the rectangular background grid cut out by a half-span QTR with rotating blades. Note that these are not aircraft body grids.

are used in the next chapter for the development of vortex tracking methodology.

To the user, the addition of these refinement grids is completely effortless (apart from the effort needed to generate them) and the establishment of their connectivity with other grids is totally transparent. This demonstrates one of the advantages and flexibilities of overset grid methodology in general and Implicit Hole Cutting in particular (in that holes around bodies and refinement grids are indiscriminately cut as previously explained). This flexibility, of course, can be considered to be gained at the expense of the requirement of a connectivity code.

The third example considers a multi-aircraft configuration and is intended to demonstrate the capability of this new approach to handle very general and arbitrary geometry with relative motion among the components. Figure 3.18 depicts snapshots of a movie showing holes in the three near-field Cartesian background grids cut out by an approaching QTR overtaking a stationary QTR aircraft. This situation is frequently encountered on the deck of an aircraft carrier. Two types of motions are present in this problem- rotation of the blades and translation of the aircraft. The 3 grids have a total of 9M points, while the entire grid system contains 112 grids and 14M points. In this case computation was performed with every other line removed.

The fourth and last example focuses on the static grids of the V-22 rotor/nacelle. The difficulty in this case lies in the close proximity of the components and the fact that the shape of the holes cut out by each other are not easily discernible and thus not easily specified by the user. Fig. 3.19a shows the surface and volume grids of the rotor/nacelle together with the outline of two background uniform Cartesian grids. Fig. 3.19b are the holes in the background grids resulting from existing approach and IHC respectively. The user-specified

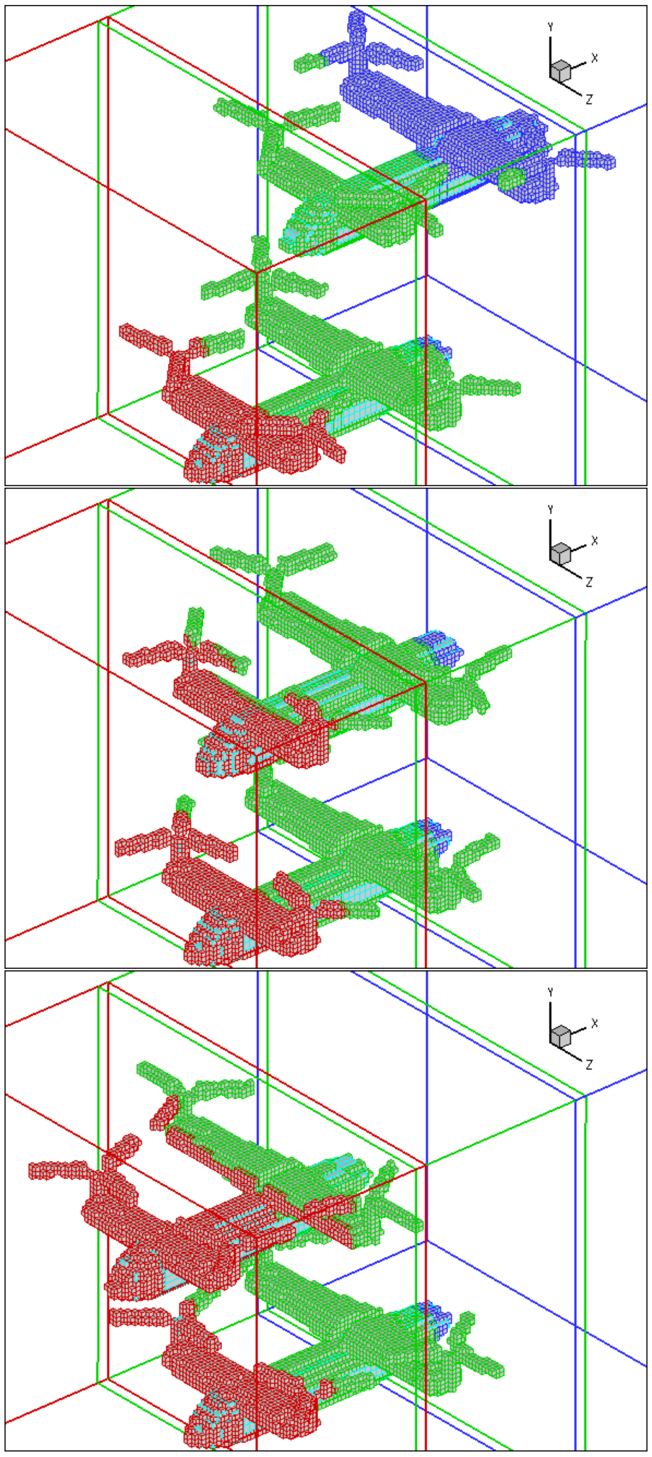
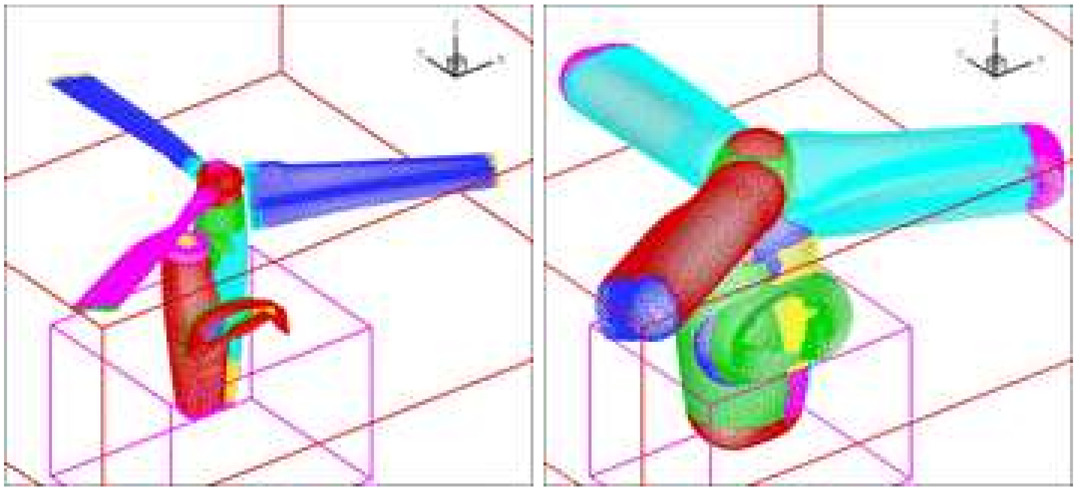


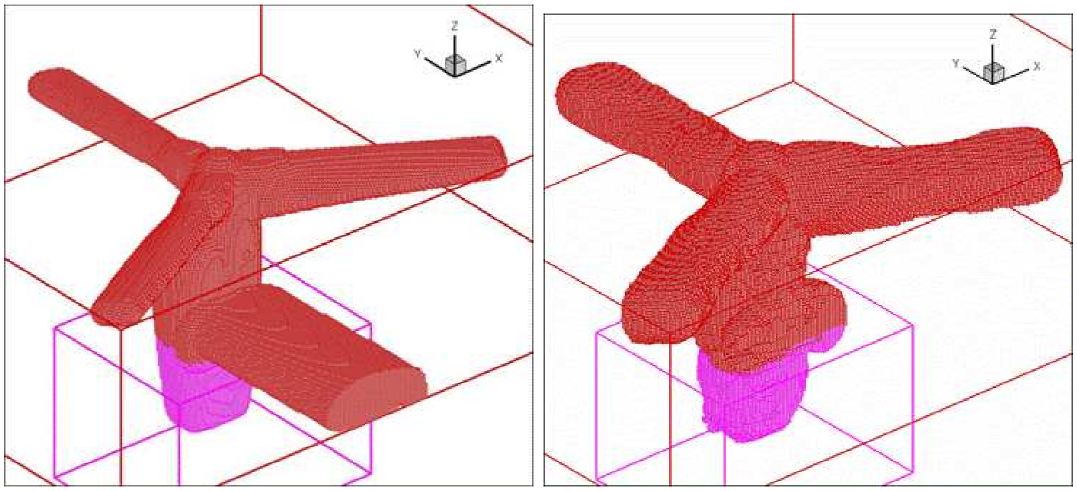
Figure 3.18: Three snapshots showing holes in the three background grids cut out by an approaching and a stationary aircraft underneath with rotating blades (note that these are not aircraft body grids).

hole (left) is not necessarily optimum even though its straight cut may look better than the wavy cut (right) from IHC, whose optimum hole shape is really a function of grid resolution.

Fig. 3.20 is an example in which the hole cut in the hub component grids by another close-by component grid may be more complicated in that the shape may not be easily discernible and thus may be hard for the user to specify. Here again in Fig. 3.20b the optimum hole in the hub grids is completely dependent on the resolution of the volume grids.

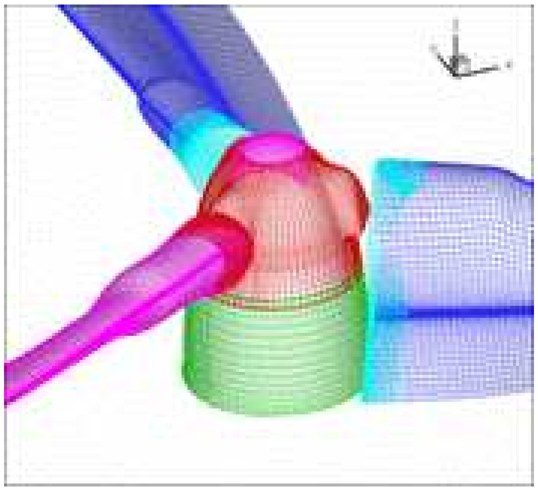


a) Surface (left) and volume (right) component grids.

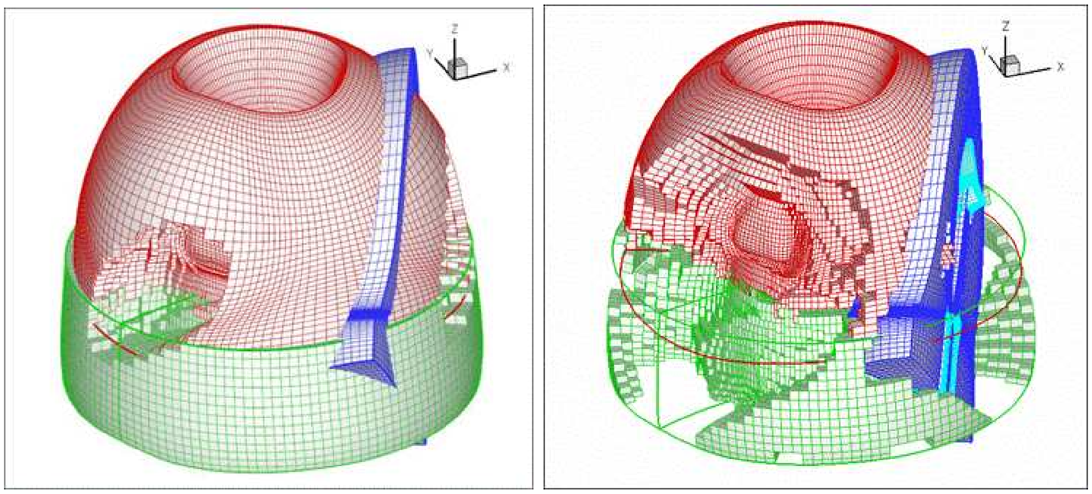


b) Holes in the two background grids using existing approach (left) and IHC (right).

Figure 3.19: V-22 rotor/nacelle assembly. Grids courtesy of NASA Ames.



a) Surface grids of blades and hub.



b) Holes in the two hub grids using existing approach (left) and IHC (right).

Figure 3.20: V-22 nacelle hub and blade. Grids courtesy of NASA Ames

Chapter 4

Automated Vortex Tracking in Rotorcraft CFD

High-fidelity, far-field and economical blade vortex capturing, tracking and subsequent interaction is one of the most important aerodynamics problems in rotorcraft. Its solution is also one of the most difficult to obtain. This chapter documents the use of multiple helical vortex tracking grids and the development of an automated tracking technique to achieve this objective. The technique is developed and tested first in 2D then on the interactional aerodynamics of a generic Quad Tilt Rotor in airplane mode and low speed forward flight. Initial results suggest the feasibility and potential of such an approach.

4.1 Introduction

It is a well known fact in the rotorcraft CFD community that high fidelity vortex capturing and the subsequent body or blade vortex interaction is of utmost importance and one of the most challenging aerodynamics problems in the design of future advanced rotorcraft. The essential problem is one of numerical vortex diffusion. Unlike numerical shock smearing or oscillation, which only spreads out or oscillates the shock but leaves the jump magnitude relatively unchanged, numerical diffusion can make the vortex disappear entirely. In addition, a shock is nonlinear and has a built-in steepening mechanism while a vortex is, like a contact discontinuity, linear with no such mechanism. Mathematically, a shock can be thought of as resembling a step function and a vortex as resembling a delta function. While the problem of high resolution non-oscillatory shock capturing has been basically solved since the 70's, the same can not be said for low-diffusion vortex capturing.

This chapter outlines the development of a true rotorcraft vortex capturing method and presents results of fore-aft rotor and wing interaction for a Quad Tilt Rotor (QTR) configuration. For simplicity, only the front and aft rotors and rear wing are represented here to showcase the essential elements of the technique. Future studies will include other secondary body components, which are non-essential for the present purpose. Figure 3.15 in the last chapter shows a half-span QTR-like model and its helical vortex-tracking (V.G.) grids used for computational purposes while the top and front views in helicopter and propeller mode are shown in Fig. 4.1. A possible overset grid system for the front and aft rotors is also shown in Fig. 4.2. The rotors and wings (outer section of rear wing) of this particular QTR are identical in scale to those of a V-22 tilt rotor.

A major aerodynamic issue of the QTR in forward flight is the interaction of the aft rotor with the wake of the front rotor which could potentially result in significant performance degradation and large vibratory loads on the aft rotor and wing.

The work documented in this chapter has two objectives, i.e., to conduct a

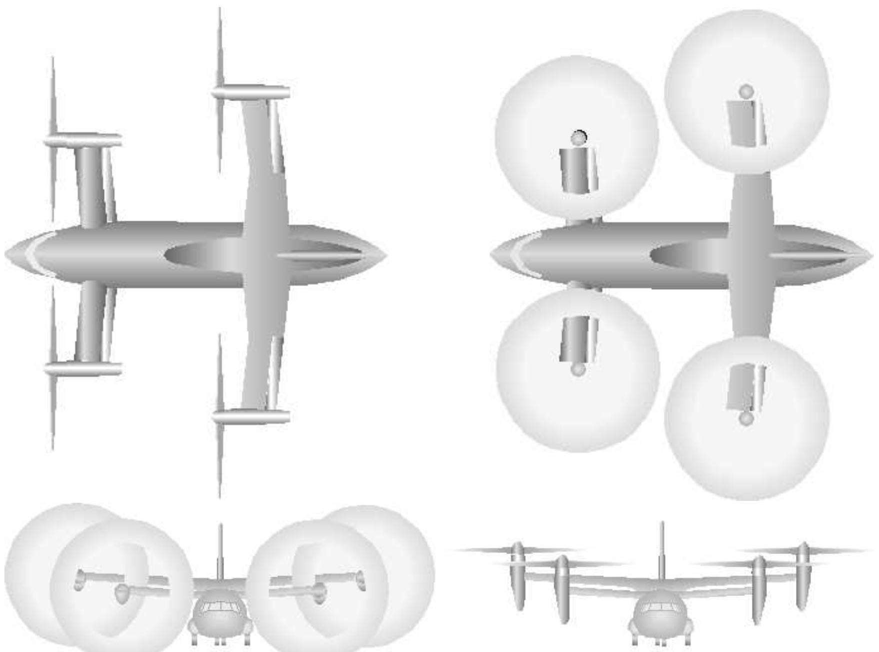


Figure 4.1: Top and front view of the proposed QTR.

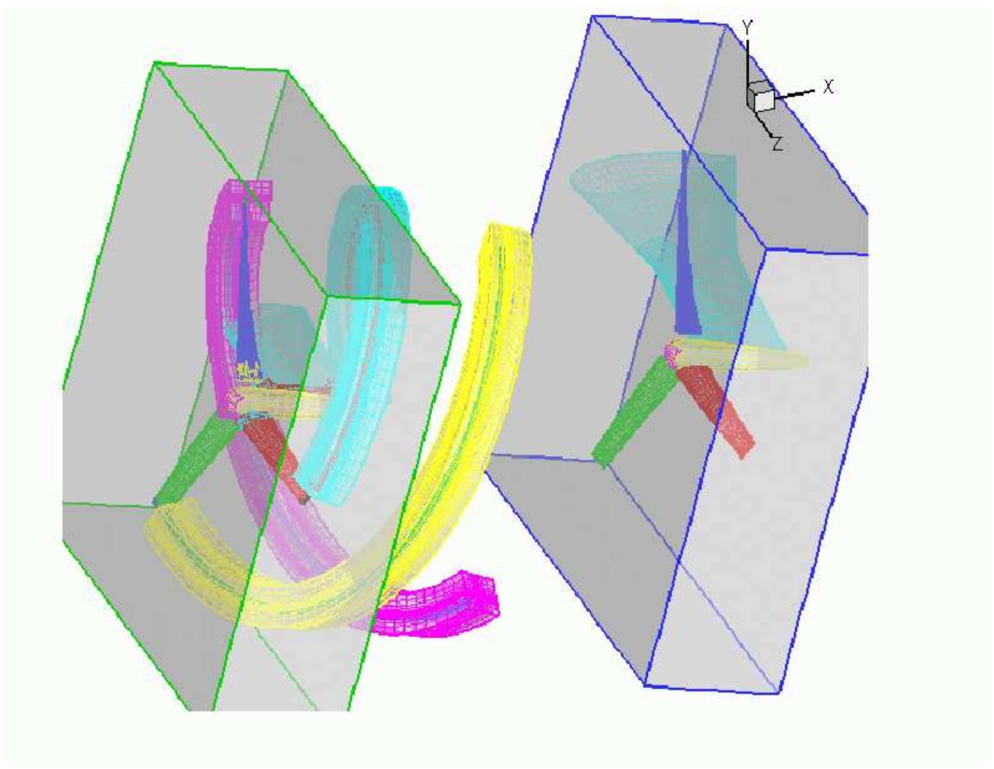


Figure 4.2: QTR rotor-wing-vortex grid system.

preliminary study of interactional aerodynamic phenomena unique to QTR in particular and tilt rotors in general, and to develop automated vortex tracking method that yields economical and high resolution vortices suitable for the study. The work on 3D vortex tracking first started as an investigation of vortex tracking in unsteady 2D BVI although the methods are very different. This 2D problem is discussed in the next section as a precursor.

4.2 2D Blade Vortex Interaction

The 2D BVI problem is the simplest idealized model of the unsteady interaction between a blade and a vortex and has been extensively studied in the last two decades. It can be simulated with either unstructured or overset grids (which require hole cutting) and has steadily matured to the point where the computation from first principles is not too expensive. Its accuracy requirement, however, is still one of the most stringent because of the interaction of the vortex with a possible shock.

The moving overset grids shown in Fig. 4.3 illustrate the effect of hole cutting during BVI. The vortex region is discretized by two nested square Cartesian grids (27×27 Δx =0.095 for outer grid, 41×41 Δx =0.025 inner grid). The inner grid has a resolution of just four cells across the core. A grid system with higher resolution as shown in Fig. 4.4 consists of two coarser outer grids (25×25 Δx =0.125, 31×31 Δx =0.04) that move at the free-stream velocity and a finer inner grid (25×25 Δx =0.01667 with six cells across core) that tracks the movement of the vortex and keeps it at the center. Time step size is kept constant at 0.25% chord for a Mach no. = 0.8.

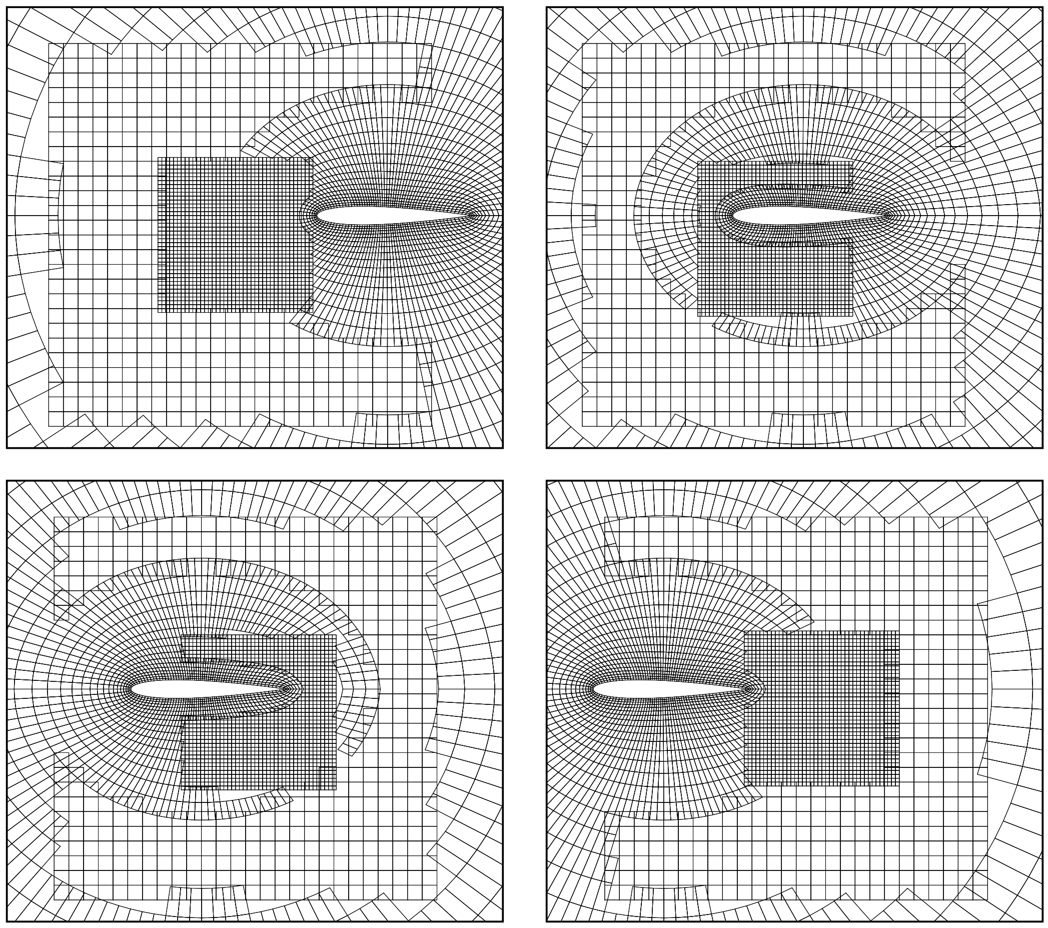


Figure 4.3: Illustration of hole cutting in moving grids during BVI (double vortex grids for lower resolution case).

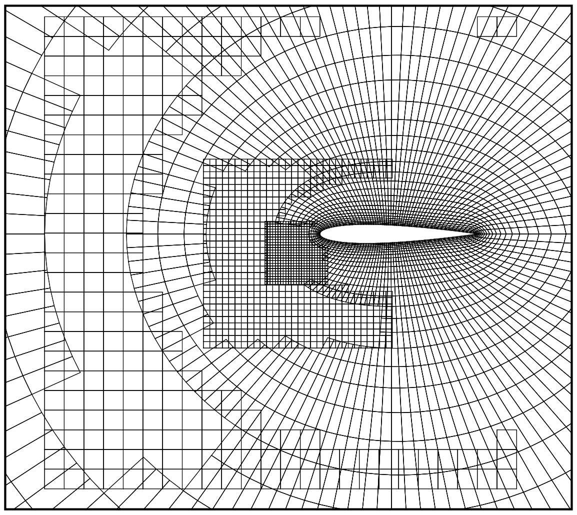


Figure 4.4: Triple vortex grids for higher resolution case. The grids contain approximately equal number of points as that of the double grids.

Tracking is necessary since the vortex trajectory changes in the vicinity of the airfoil before and after the interaction, and the innermost grid might not be large enough to contain the entire core. However, the vortex profile lacks a definite recognizable shape in this vicinity. The grid thus loses track of the vortex in this region. When this happens, the vortex grid defaults to move at the free stream velocity. When the vortex profile eventually reappears after the interaction, the grid is hopefully in the immediate vicinity and able to recapture it and places itself in the center of the vortex, provided that the vortex path has not been too severely affected.

Moving the curvilinear grid of a solid body using grid metrics requires that the connectivity be reestablished every time step. For the 2D Cartesian grid of a vortex (no solid body grid) moving to the right, however, we can employ 'moving static grids' in which the leftmost line is discarded and a rightmost line is added

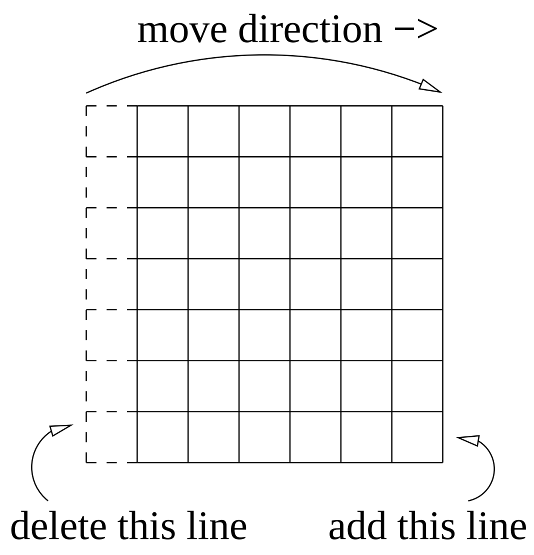


Figure 4.5: A moving static grid.

at the same time after the vortex has moved a distance equal to the grid spacing Δx. This is schematically shown in Fig. 4.5. This way, connectivity needs to be reestablished only once every few time steps depending on the value of Δx. The variables on the new added line are then interpolated from the optimum donor cells.

This BVI problem consists of an O-grid (121×38) around a NACA 0012 airfoil at zero angle of attack in a freestream velocity of Mach 0.8. The non-dimensional vortex strength is 0.2 with a tight core radius of 0.05 chord. The vortex is initialized at -8 chords with a miss distance of -0.26 and -0.125. The computation is terminated when the vortex is at $+8$ chords after 6400 time steps. Elapsed CPU time on a 667 MHz Alpha processor is about 8 minutes for a total point count of about 6800.

Figure 4.6 and 4.7 show the pressure contours before and after the interaction. Results for both linear and cubic interpolation are shown. It is observed that the interaction with the airfoil and shock distorts the vortex profile, resulting in

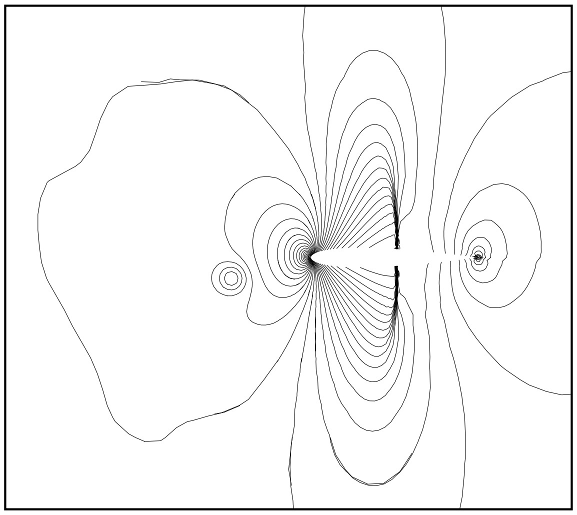


Figure 4.6: Pressure contour just before interaction.

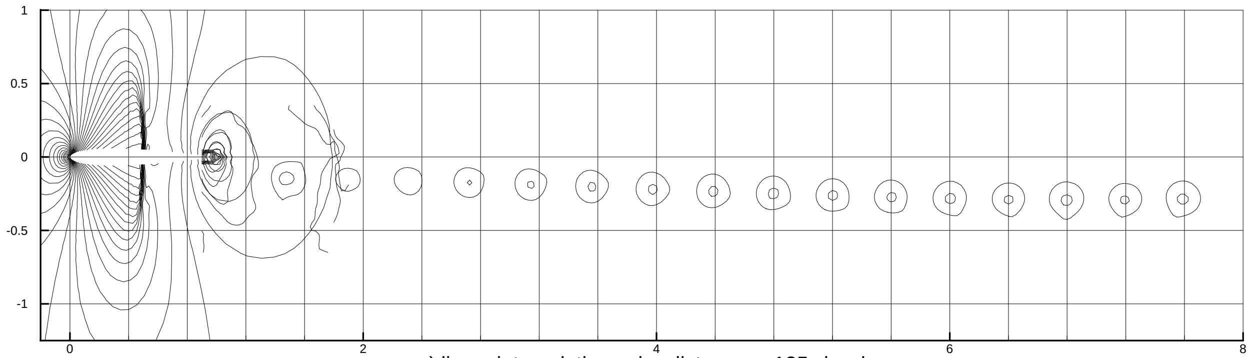
higher vorticity on one side. The subsequent sinusoidal trajectory of the vortex is probably due to this asymmetry and the associated rotation. This phenomenon is also vaguely observable in the results obtained in [28]. The results with linear interpolation in Fig. 4.7a also show the same phenomenon. The vortex is, however, much more diffused, convecting slightly faster and rotating at a higher period of 11 chords/rev. This figure clearly indicates the importance of high-order interpolation of boundary values in a high gradient region.

The instantaneous pressure contours during BVI are displayed in Fig. 4.8 showing four snapshots of a movie. Several additional phenomena can be observed in the movie besides that just described above:

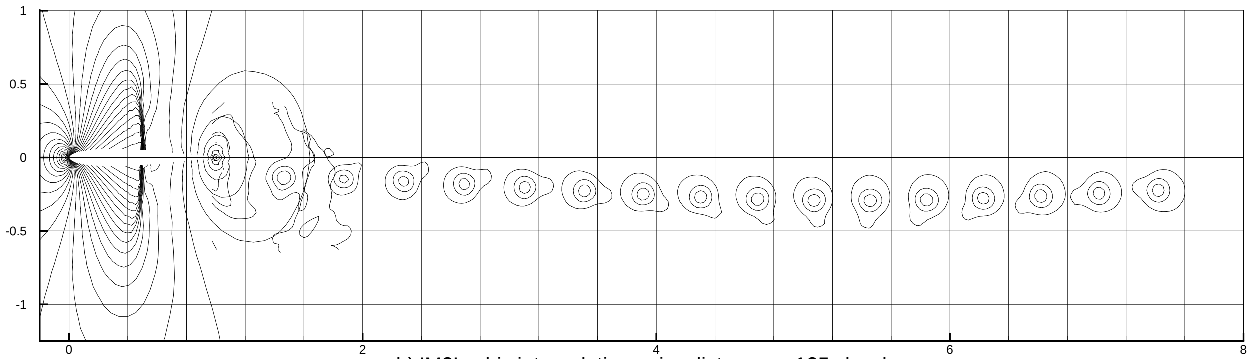
- The λ-type shock formed during interaction is clearly visible.

- Acoustic waves are generated and dispersed (better captured in [28] with adaptive mesh).

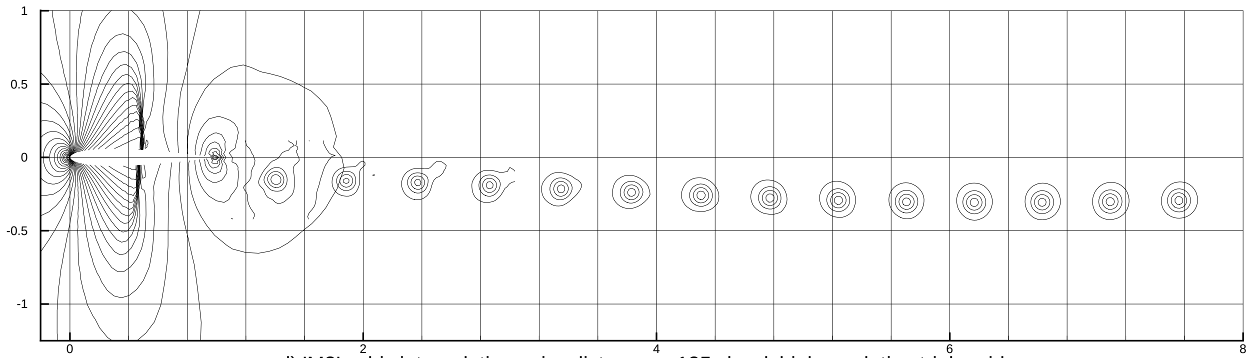
- The lift history curve shown in Fig. 4.10 indicates that the interaction has



a) linear interpolation, miss distance = .125 chord



b) 'M3' cubic interpolation, miss distance = .125 chord



d) 'M3' cubic interpolation, miss distance = .125 chord, high resolution triple grid

Figure 4.7: Pressure contours of distorted vortex after interaction.

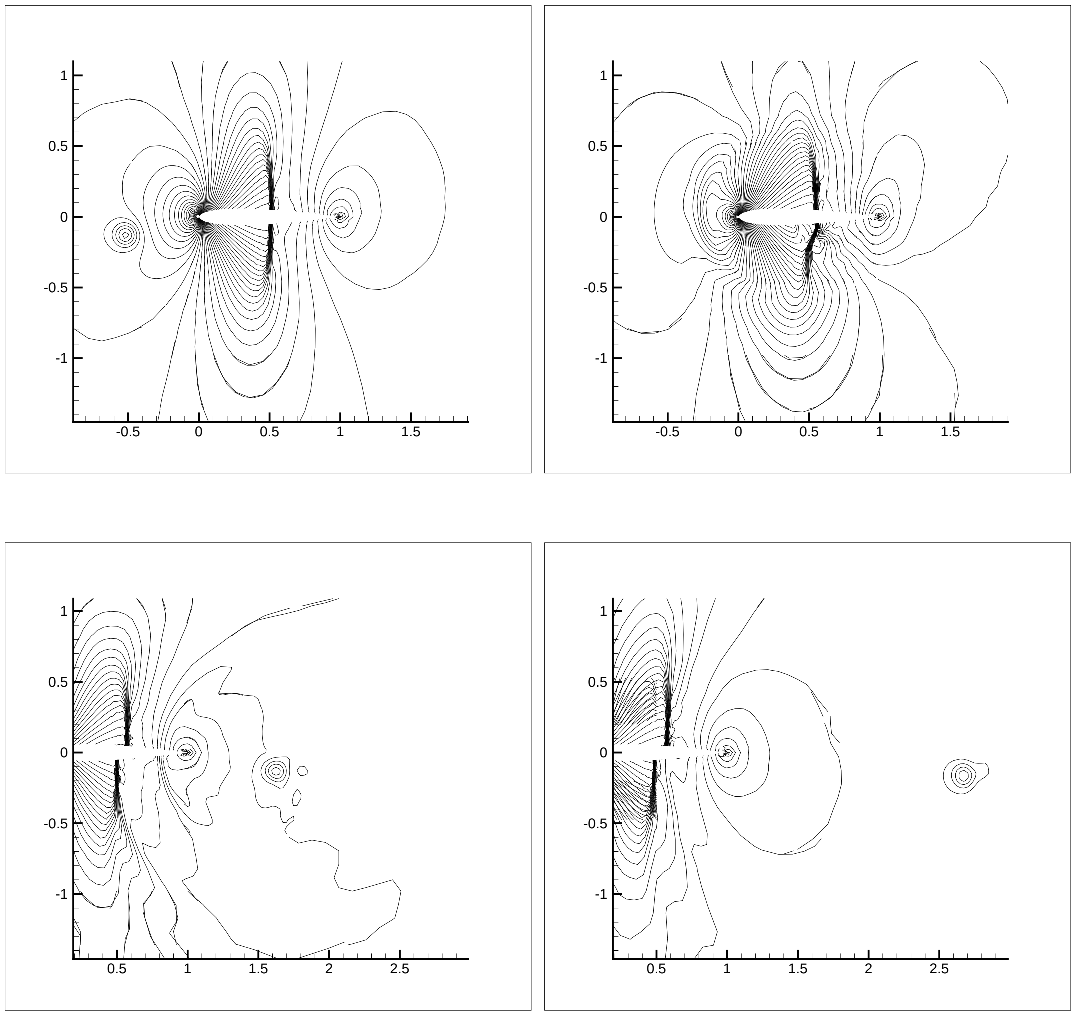


Figure 4.8: Instantaneous pressure contours during BVI.

a long lasting effect on the airfoil.

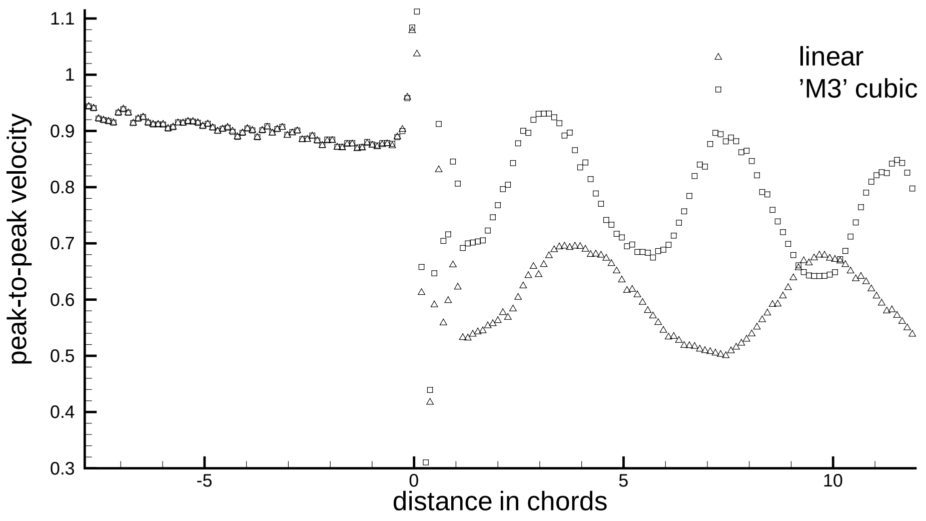
The same distortion effect is also observed in the decay of vortex peak-to-peak velocity in Fig. 4.9a. Before interaction, differences in the linear and cubic results are insignificant. The decay after interaction depends on the direction in which it is measured. The sinusoidal curve in the figure is due to measurements in the fixed horizontal direction as the vortex and its 'lump' rotates. Linear interpolation results of the lower curve shows an increase in vortex diffusion of about 20% and an increase in the rotation period to 11 chords/rev (same as obtained from Fig. 4.7). Note that cubic interpolation is important only during the interaction in which large gradients exist. The erratic oscillation of the curve in the airfoil region is meaningless since the vortex is not well defined there.

Results for the triple grid vortex in Fig. 4.9b shows that high-order interpolation is still important even in a high resolution BVI although the difference is not as drastic as before. The pressure contours in Fig. 4.7c also shows diminished sign of distortion after about four to five chords behind the trailing edge.

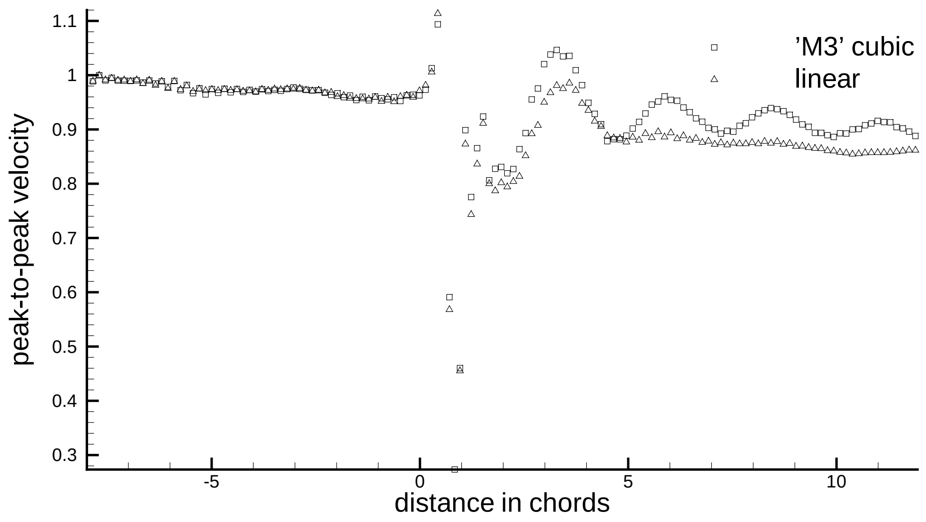
Although the 2D BVI problem has been extensively studied for a long period and its basic phenomenon well understood, its CFD simulation from first principles using different codes still yields inconsistent results (visible in lift magnitude). Even results from the same code using various values of a parameter, e.g., limiter, are visibly different. Figure 4.10 shows the lift and moment time history of the airfoil generated using two different limiters. The general trend is well captured. But they show discrepancies before and after the interaction. The minimum C_l is the same at -0.185. This value has ranged from -0.18 to -0.25 as reported by different researchers (e.g. [48], [27], [28]).

This discrepancy is likely due to the severe interaction of the two flow features with high gradients, i.e., shock and vortex, in the vicinity of the airfoil and its associated numerical sensitivity. Undoubtedly more work is needed to clarify this issue. This fundamental 2D BVI problem is sophisticated but simple enough to provide an excellent test bed for high accuracy check of a compressible CFD code.

One approximate indication of the performance of a connectivity method is

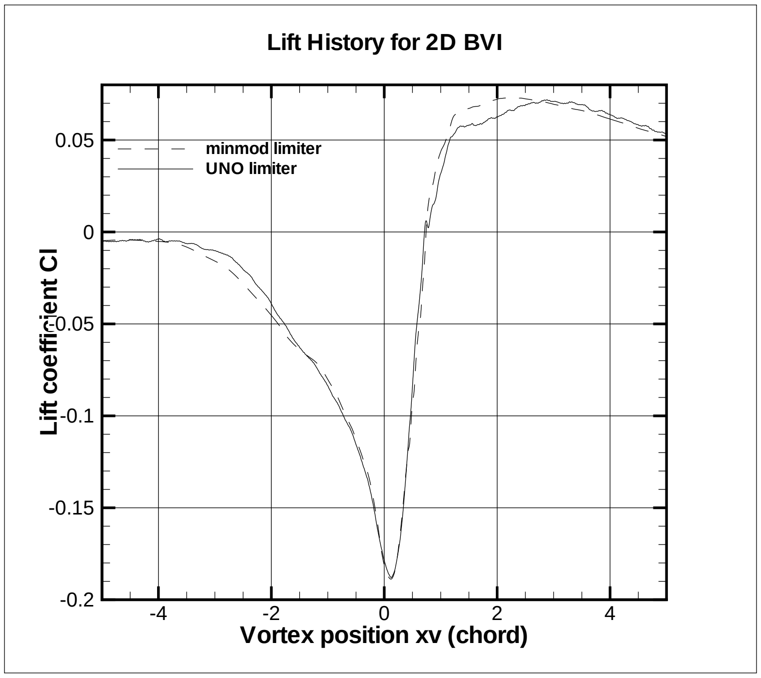


a) 4 cells across core (double vortex grids)



a) 6 cells across core (triple vortex grids)

Figure 4.9: Decay of vortex peak-to-peak velocity.



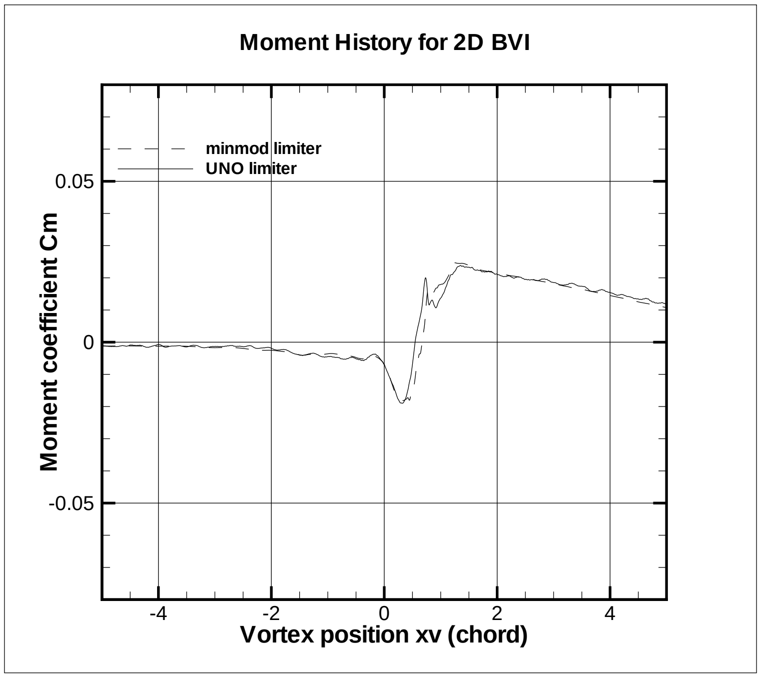


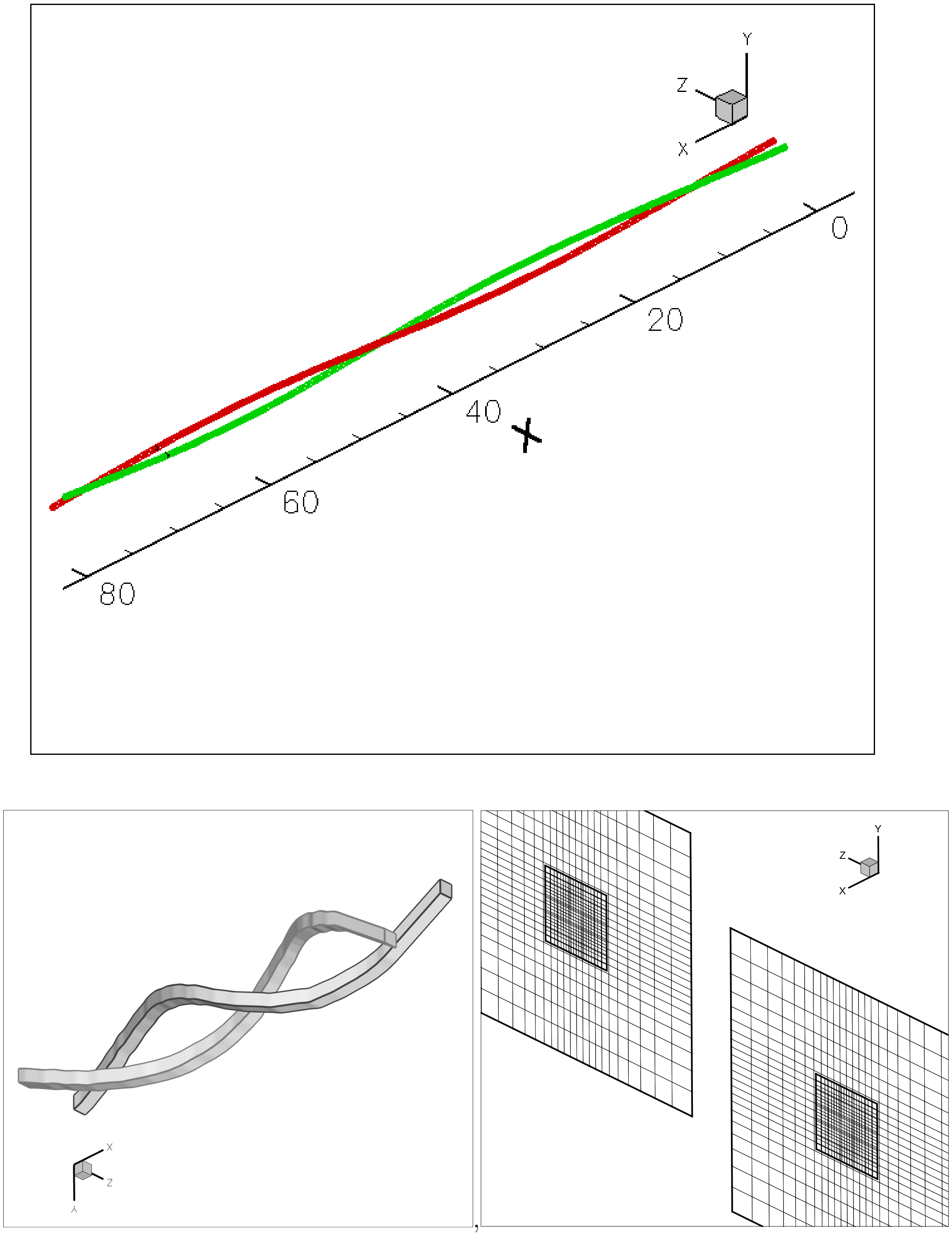
Figure 4.10: Lift and moment time history of an airfoil in a 2D BVI for two different limiters. $M_\infty = 0.8$, $\Gamma/(V_\infty c) = -0.2$, $y_v = -0.26$, $r_c = 0.05$.

the CPU time needed for connectivity establishment relative to the flow solver time. This, of course, depends on the type of solver employed, location and number of grids etc. For the present 2D BVI problem, this figure is about 15%, an acceptable performance.

4.3 CoRotating Vortices

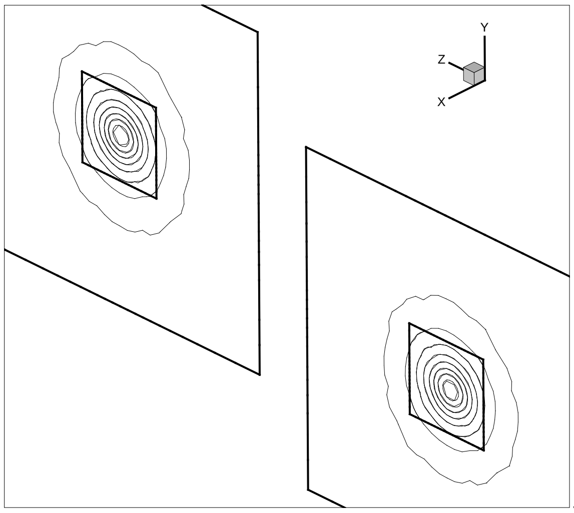
Before investigating the more complicated tracking of helical vortices, it is instructive to briefly study the results and test the method for a simpler dual vortex case without any vortex-generating body. This ideal steady state problem with two co-rotating vortices, which is the result of their mutual swirl effect, would be investigated in this section. The vortices are initialized upstream as boundary condition. Their core radius r_c is 0.05 and their centers are $6r_c$ apart. Mach number and non-dimensional vortex strength are 0.25 and 0.2 respectively. The dual purpose of this study is the employment of a pair of highly concentrated high resolution grids to track the vortices as they rotate around each other to form a double helix and to observe the numerical diffusion. The tracking procedure and the grid system are similar to that for a rotor in the next section except that the vortices here are essentially straight.

The pair of doubly-nested vortex grids with small but high-resolution cores are initially straight and parallel (see bottom right of Fig. 4.11 for a cross section). Gradually they adapt to become a double helix (top of Fig. 4.11) as the co-rotating vortices are being tracked. The system of 5 grids in Fig. 4.11 contains a total of 3165 points per axial section (= $21\times21\times2 + 23\times23\times2 + 35\times35$). Compared with a conventional approach using non-tracking static



,

Figure 4.11: Top – ‘Spaghetti’-like adapted double helix inner vortex tracking grids. Bottom – Axial distance scale contracted (left) and doubly-nested vortex grids (core Δs=0.01) (right). r_c=0.05.



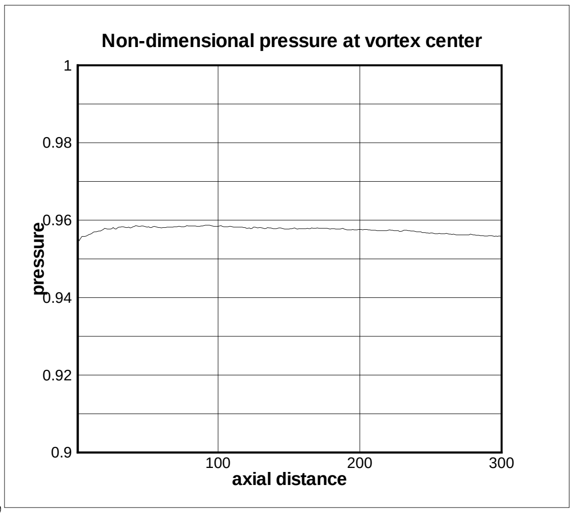
,


Figure 4.12: Pressure contour near far end and numerical diffusion of pressure at vortex core.

grids with similar resolution, the point count of this grid design is significantly lower. The background grid not shown is a static rectangular block (35×35) with a uniformly spaced square core stretched in the transverse directions. The core Δs in the transverse and axial directions are 0.01 and 0.24 respectively. For about 380 axial sections, the total number of points is 1.2 million.

The vortices are tracked and grids adapted until the flow is fully developed. The pressure contour at the far end and numerical diffusion of the vortices (as measured by the pressure at the vortex core) are shown in 4.12. It can be seen that the diffusion is practically nil after a tracking distance of about 1500 core radii or about 330 axial Δs. Since the resolution of 10 grid cells across the core is more than sufficient, this result is within expectation. The significance of this study lies in the economy of the computation due to the use of vortex tracking. The success in this mostly straight line case, however, does not necessarily guarantee its feasibility in helical tracking, which would be investigated next.

4.4 Vortex Tracking Methodology

Despite its promise and potentially high payoff, vortex tracking technology in CFD for rotorcraft aerodynamics is still relatively new and uncommon, especially for the method given here. The central idea underlying this computational methodology is deceptively simple. The difficulty lies in the implementation of this idea in a codable and robust algorithm. The heart of the method is a set of long helical vortex tracking grids with high grid density at the central core where each helical vortex trajectory resides at the completion of the grid adaptation (or tracking) process.

4.4.1 Vortex Grid Generation

The generation of the initial grid system for a helical vortex starts from at least 2 (as used in this work), but not more than 3, nested and mildly stretched 2D Cartesian grids (21×21 points with 1.15 stretching factor for the innermost grid). Three grids are usually preferable for problems with strong interaction such as hover. They are then extruded to become long helical 'spaghetti' grids. The achieved resolution at the innermost grid can be very high (Δs of about 1% tip chord or less) and yet requires only a very small number of points ($300-400$K for one revolution of the grids).

Compared with nested grids, a single stretched grid has the drawbacks of unnecessarily larger number of points and very high cell aspect ratio, of about 30:1 in our case, in the outer edge region. This can be reduced to a much milder 3:1 by using 3 nested grids, or 9:1 for 2 nested grids. Spreading out the clustered points at the outer edge can also reduce the ratio but creates the problem of highly skewed cells at the corners. Another drawback is the lack of a transition zone for grid resolution to decrease smoothly from the very fine square core to the coarse background grid at the far end of the vortex grid. This problem is made much less severe in nested grids by merely extruding each outer grid a little further than the inner one. But perhaps the most important reason for using nested grids is the possibility of faster tracking by using the coarser grid for initial tracking and the finest grid for eventual fine tuning of the trajectory.

An axial spacing of about $20-25$ times Δs is found to be sufficient to capture the vortex well. Anything much larger than that runs the risk of being too dissipative for the vortex. For a 3-grid system, the two inner grids have the same axial spacing, while the outer grid axial spacing is relaxed to 1.5 times larger. Every 2D station of each grid is tilted so that it is orthogonal to the direction of extrusion. The pitch distance of the initial helix can be reasonably arbitrary (but preferably estimated from existing wake models if possible), since the tracking process would gradually adapt the grid to align with the vortex trajectory and keep it at the center.

Figure 4.13 shows one station of a nested 2-grid system and the hole caused by and filled up with the innermost grid. All overset grids connectivity computations are performed using the new approach explained in Chapter 3. This code generates a set of holes in the grid system for the rotors and wings such as that shown in Fig. 3.16.

The above vortex grid system compares very favorably in terms of number of points and resolution with commonly used overset grids, such as those in [20] which contain 64M points of mostly Cartesian background grids with 5% tip chord resolution. More importantly, the axial axis of these helical tracking grids

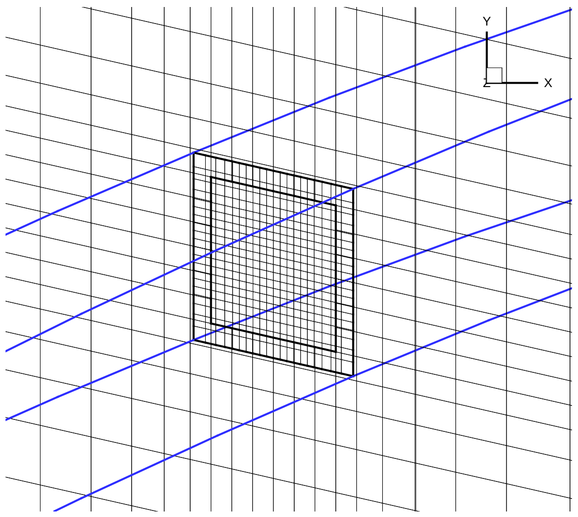


Figure 4.13: One station of the nested vortex grids and a hole at the center.

is approximately aligned with the direction of vortex convection. This eliminates the problem of skewness between convection direction and a grid plane which is a significant source of diffusion for Cartesian grids. In exchange for this advantage, a special grid adaptation method for vortices, as outlined in the next section, is necessary.

4.4.2 Grid Adaptation

The first 2D station of the helical tracking grids must be placed close to the blade tip trailing edge as shown in Fig. 4.14 so that it is able to capture the vortex. Otherwise the entire vortex would be missed completely.

Once a vortex is somewhere within the grid, the location of its center (trajectory) at every n^{th} 2D station is easily determined by testing for the minimum pressure (or maximum vorticity). These offset distances from the grid center (in terms of number of Δs of the square core) for all the in-between stations are interpolated from a cubic interpolant to maintain a grid smoothness of at

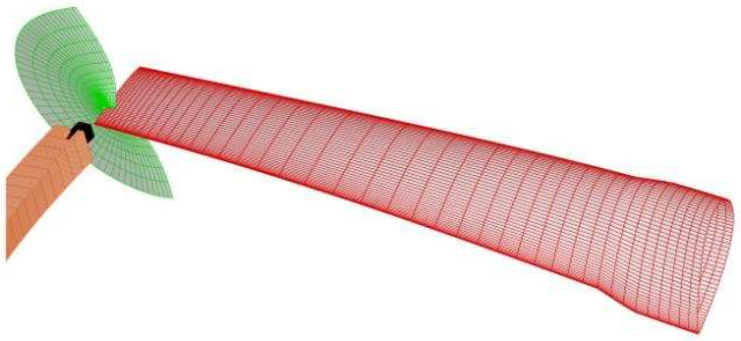


Figure 4.14: One end of the nested vortex grid system is placed at the blade tip trailing edge to capture the vortex. A slice of the semi-circular tip-cap grid is also shown.

least C^1. Naturally they can not be whole numbers of Δs and the vortex at these in-between stations may not be exactly at the center. The grid is subsequently adapted according to these offset distances. The increasing spacing in the transversely stretched tracking grids is taken into account in determining these distances. The two geometric parameters adjusted by the adaptation process are the radial and pitch distance.

Solution values at the new locations after adaptation can either be interpolated from other grids or remain unchanged before they are time-marched again. The latter option may need a little longer time to converge before the next adaptation step.

With an inexact initial estimate of the trajectory, part of the vortex downstream is likely to be outside the grid. No guidance is available to adapt this part of the grid. The amount of offset distance to adapt is then linearly extrapolated for the first such test station. The rest can be taken to be either the same as this first one or extrapolated again. A faster way is to truncate the grid temporarily at a certain preset distance downstream of the location where the vortex leaves the grid. This last option is especially attractive for poor initial estimate of the trajectory.

4.4.3 Frozen Grids

The above grid adaptation process involves multiple steps as more and more of the vortex is captured at the center of the grid. Each step lasts as long as it takes for the vortex to emerge closer to the grid center and stabilize. This takes thousands of iterations. The complete capture of the vortex (from several to a dozen or more such steps depending on how good the initial estimate of the trajectory is) can thus be very time consuming even though the total number of points has been drastically reduced by using vortex grids. Traditional methods of convergence acceleration can only reduce CPU time to a certain extent. Further acceleration requires the use of multiple stages of different 'frozen grids' that can be conveniently implemented with overset grids (whether structured or unstructured).

Frozen grids can only be used for steady state problems and is based on the premise that grids with relatively little solution change and flow activity can be frozen during most of the adaptation process. That implies all except the outer vortex grids used for tracking the trajectory. CPU time for the adaptation can be drastically reduced (about 70−80% for the case in this work). They must be, of course, unfrozen at the last stage. There is plenty of room for the design of different combinations of frozen grids.

Since they are larger and have square cores fine enough to capture the vortex (though not as sharp), the outer vortex grids are used for tracking purposes initially while the inner vortex grids are frozen. They remain frozen until the last adaptation stage is completed. The inner grids can then be used for fine tuning the trajectory. It was observed that without the fine tuning, the accuracy of the captured inner vortex trajectory is $0-2$ grid spacings (or $0-2\%$ c_{tip}), which for many purposes is accurate enough.

4.4.4 Convergence and Vortex Formation

Obtaining a converged solution and capturing the vortices using the slender vortex grids is much harder compared with simply refining the wake grids in the background (which can be prohibitively expensive). It is not sufficient to use thrust or other integrated quantities to determine if convergence is reached. The vortex is a very demanding flow feature that does not form and settle as quickly and easily as the integrated quantities. This is especially true the further away it is from the tip trailing edge where it is generated.

Vortex at any section along the grid can only form after those preceding it. As time progresses from quiescent flow, the tip vortex gradually strengthens itself and stabilizes starting from the blade tip trailing edge. This process then propagates along the vortex grid. Because the initial estimate of the trajectory is generally not very accurate, the vortex would gradually deviate further from the grid center until it leaves the grid and disappears entirely. A residual history curve that is still going sideways instead of dropping off on its way to machine zero most often indicates the vortex is still actively forming somewhere

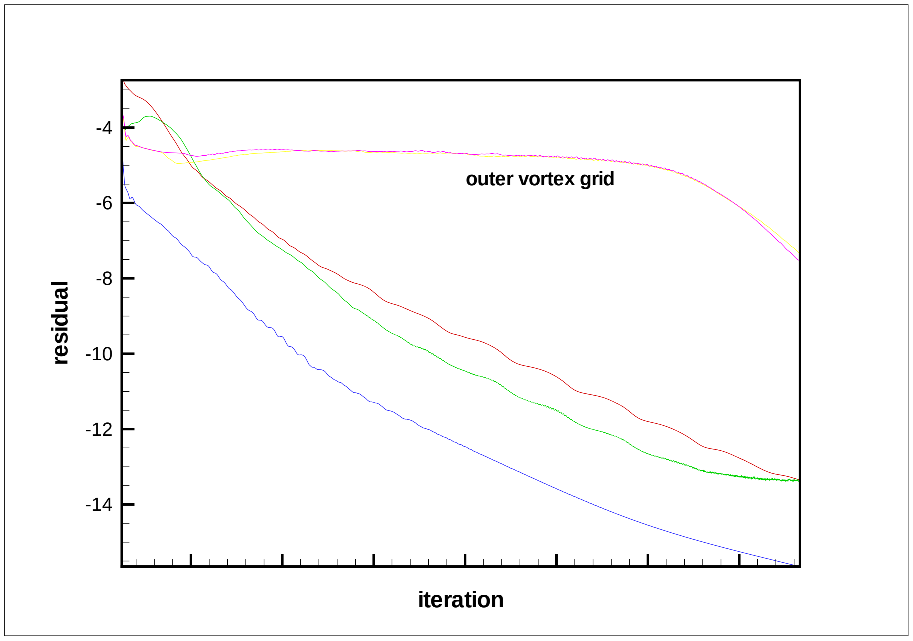


Figure 4.15: Residual history of component grids. All except vortex grids exhibit approximately linear decrease.

downstream even though the integrated quantities have long since converged.

Fig. 4.15 shows a typical example of the convergence history of individual grids before adaptation. As opposed to the curves for other near-body or off-body background grids which drop off more or less linearly, those for the vortex grids level off after an initial drop. They remain leveled for a while before diving precipitously, often passing the curves for the other grids, racing towards machine zero. The extent of the horizontal movement depends on grid resolution. The residual curve for the inner vortex grid can go sideways for tens of thousands of iterations before the plunge, giving an initial impression that it has stagnated.

Fig. 4.16 depicts the convergence history for several adaptation stages. The initial estimate of the trajectory in this case is quite good, as can be inferred from the small number of adaptation steps. The convergence is faster the closer

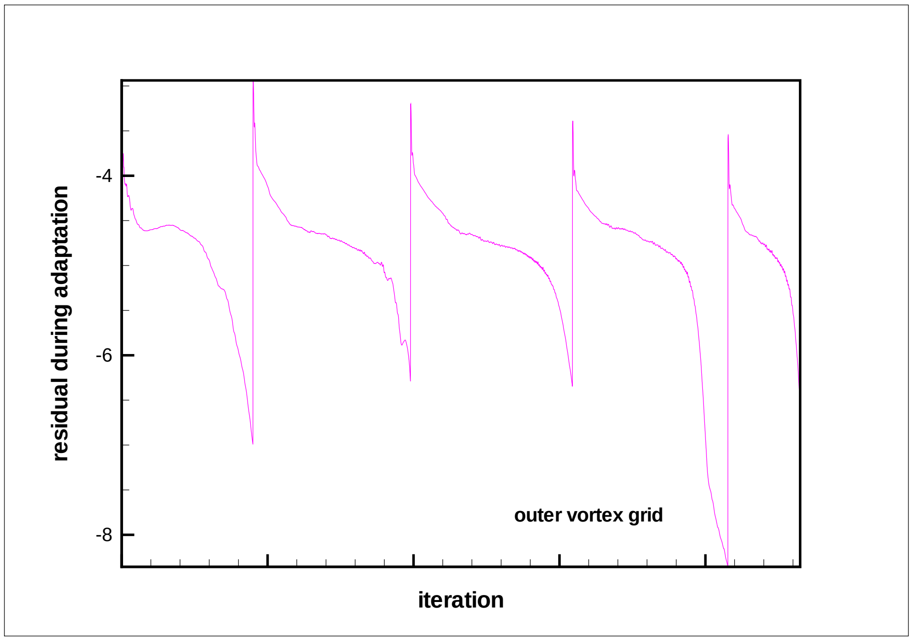


Figure 4.16: Residual history of outer vortex grid during the adaptation steps.

the entire vortex is to the grid center as shown in the last few steps in the figure (the very quick last step is not shown). Another observation is worth noting here. The final capture of the vortex can still be several adaptation steps away even when the entire vortex is already inside the outer grid and within close range of the center. This is possibly due to the fact that the exact trajectory of the vortex is still not definitely known when a sizable part of it is still outside the fine square core of the grid.

4.4.5 Adaptation Automation

The above adaptation process can be automated by first determining if the vortices have settled before adapting the grids. This can be done by checking the locations of the vortex centers at every n^{th} station after a preset number of iterations. If they remain unchanged for three or four such tests, it is assumed that they have all settled and adaptation is activated. Numerous examinations of the convergence curve also confirm that the residual at this point is about half way through the final ‘plunging’ phase as can be seen in Fig. 4.16.

4.5 Demonstration of Vortex Tracking

Examples in this chapter are computed in quasi-steady state. This is a state in which the effect of rotation is simulated but not the physical rotation itself. In a real experiment, rotational effect can be obtained simply by rotating the blades. In a computational simulation, however, this effect is obtained from the combination of two separate steps, namely, the rotation of the blade grid and the addition of a grid velocity term in the flow equation. Without the latter, the blade would move but no rotation would be felt. Quasi-steady state is the state in which the former is omitted. It can not be exactly replicated in a real experiment because these two steps are really inseparable. It can only be thought of as a frozen moment in time, and the computation is marching in time towards that fictitious moment. This is an example of the general ability of computational science to simulate a phenomenon either not realizable or too difficult to perform in an experiment. Quasi-steady state eases the process of vortex capturing and provides an accurate starting state to initiate the unsteady case with rotation.

The underlying flow equation in the simulation is inviscid to ensure that any vortex dissipation detected is numerical in origin. While every grid runs with its own global time step, this value is different for different grids. Towards the end of convergence, the same time step of $\Delta\psi = 0.0625°$ is used throughout.

NACA00xx airfoil sections are assumed for both the blades and wings. Blade twist distribution is a quadratic curve fit of the bilinear distribution of the original blade, with a total twist of 40°. Thickness and chord length at the tip are 9% and 22 in. respectively. The blades are given a precone angle of 2.75°. The wings have a dihedral of 3.5°, a forward sweep of 6° and a chord of 100 inches. These parameters for the rotors and wings (outer section of rear wing) are identical to those for the V-22 tilt rotor. In order to minimize the total number of points, O-H grids have been used throughout. These are blunted at the trailing edge to eliminate the O-grid singularity and smoothen the periodicity condition. To further reduce time to insight, several components deemed more secondary for vortex capturing have been omitted. These include engine nacelles and spinner, fuselage and, in some cases, front wing. Two background grids, one Cartesian for near-field (with a coarse $\Delta s = 0.45\ c_{tip}$) and one stretched Cartesian for far-field, connect all the component grids together.

The two rotors are 53° out of phase and positioned such that the front vortices do not hit other components directly in order to avoid severe interaction. A vortex in a direct hit, such as that through the wing, can only be accurately captured in an unsteady rotating case (this is discussed at the end of the chapter). Freestream Mach number and temperature are 0.445 and −2.8°C respectively, which are taken from Meakin [76] for a V-22 forward flight simulation. With a rotational speed of 333 rpm and a radius of 19 ft, this yields an effective tip Mach number of 0.76. The rotational speed is increased to 397 rpm for hover or any flight condition in the transition zone.

Parasite drag is estimated from the equivalent flat plate area of 54 ft^2, as suggested in [41], and an assumed air density of 0.0015 slug/ft^3 at an altitude of 14,930 ft to be about 9,378 lb, or 0.0476 when non-dimensionalized by $\frac{1}{2}\rho_\infty V_\infty^2 A_{rotor}$ (which is really flat plate area / rotor area). For this drag, the required average forward thrust on each of the 12 blades is 780 lb (or 0.0040). This can be obtained by setting the pitch angle at the tip to about 43.3° (or 46.4° at 75% R). For the same cruise condition, the collective pitch angle in [76] was found to be 39° at the tip (45.1° at 75% R). The higher pitch angle obtained in this thesis is expected considering the use of symmetric NACA00xx instead of the cambered XN-series airfoils. The baseline case studied here has a tip angle of 45°. This results in a thrust about 2.3 times higher and a shock at the blade tip that is extended inboard to a radial distance of $r \simeq 0.8$.

Several different basic body and grid configurations are set up to study the unique aerodynamic problem of QTR in forward flight. These include front rotor only, front-aft rotor pair and front-aft rotor pair with rear wing, all with or without vortex tracking grids. Collective angles at the tip for these cases are set to 43°, 44° and 45°. A minor study of grid refinement of the background grids has also been conducted, but the results are inconclusive as of this writing and will be left to future work.

4.5.1 Normal Force Distribution

Figure 4.17 shows the spanwise non-dimensionalized normal force (thrust) distribution on the three aft rotor blades with and without the rear wing. Azimuth ψ in the figure is measured counter-clockwise front view from down position, while the blades are rotating clock-wise. In this convention, the wing's azimuth is 270°. The three curves without wing are not identical because the front and rear rotors are not aligned and thus the flow is not axisymmetric.

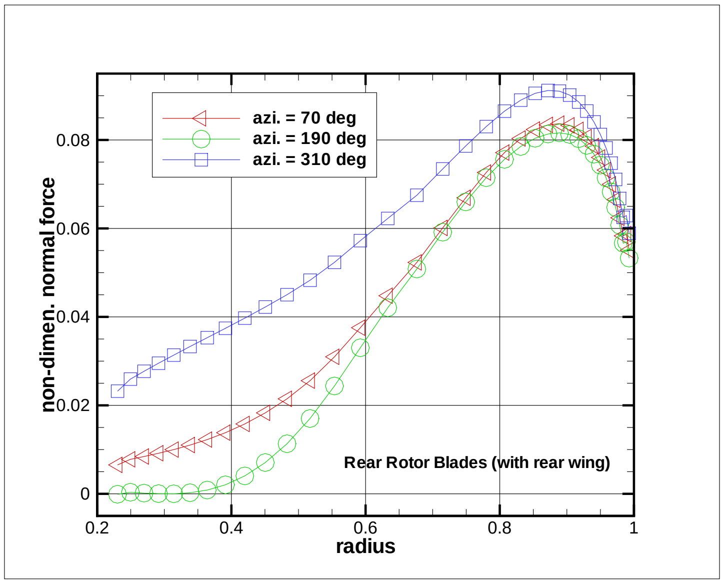
The presence of a wing closely behind the prop-rotor has a significant effect on the blade thrust. Referring to the top figure, it is observed that blades that are approaching the wing (from below) experience a beneficial influence in the thrust, while those that are receding from it (above the wing) experience an opposite effect. Similarly, this is also true for the front blades as shown in Fig. 4.18 for a lower tip collective pitch of 44°. The thrust on each blade (area under curve) is 0.00722, 0.00613 and 0.00371. The large difference of 50% between the first and the last value is due to the negative blade load in the region inboard of about $r \simeq 0.5$. It is noted here that the largest streamwise separation distance between the preconed blade and the forward-swept wing is only 2.37 c_{tip} at the blade tip.

A clearer picture of this effect is the plot in Fig. 4.19 of the thrust on a blade against azimuth angle. Results are given for 3 different tip collective pitch, 43°, 44° and 45°. The sharp jump at ∼270° reflects the presence of the wing. The thrust in each case in the absence of the wing is indicated by the dash lines.

Comparison of spanwise normal force distribution on one of the rear rotor blades ($\psi = 310°$) with and without using vortex grids in the absence of a wing is shown in Fig. 4.20. The difference is non-negligible despite only mild vortical interaction with the blade. With the vortex grids, the normal force is lower inboard and higher outboard of $r \simeq 0.8$, resulting in a net decrease in total thrust of ∼3.5% from 0.00985 to 0.00951 as given in the table later.

4.5.2 Pressure Distribution

Figure 4.21 shows the corresponding chordwise pressure distribution of the aft blade, with and without the rear wing and vortex grids, at $\psi = 310°$ and $r =$



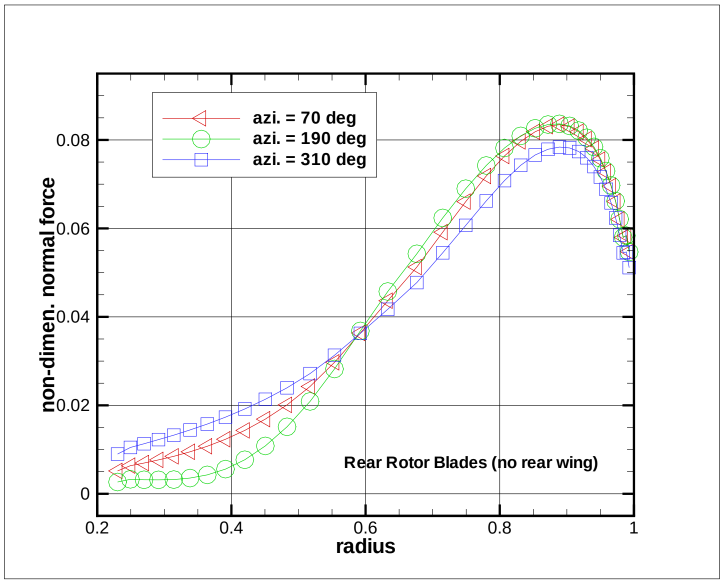


Figure 4.17: Spanwise normal force distribution for the three aft rotor blades. Top – with rear wing. Bottom – without rear wing. Tip collective pitch = 45°.

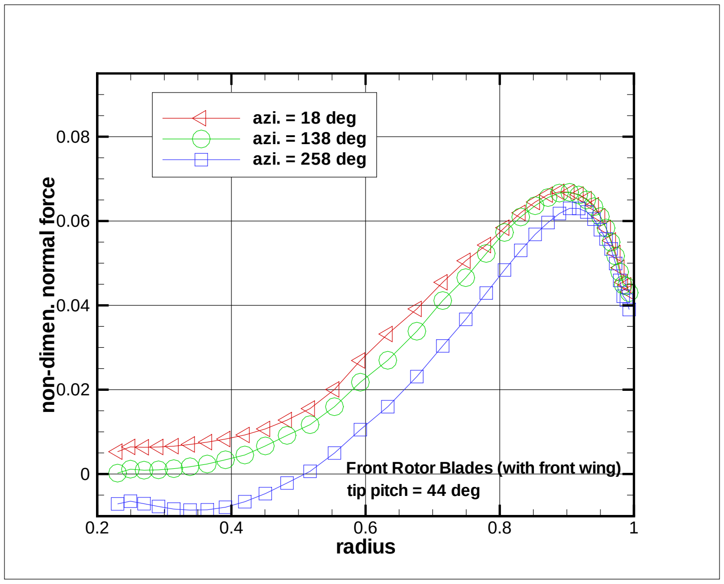


Figure 4.18: Spanwise normal force distribution for the three front rotor blades with front wing. Tip collective pitch = 44°.

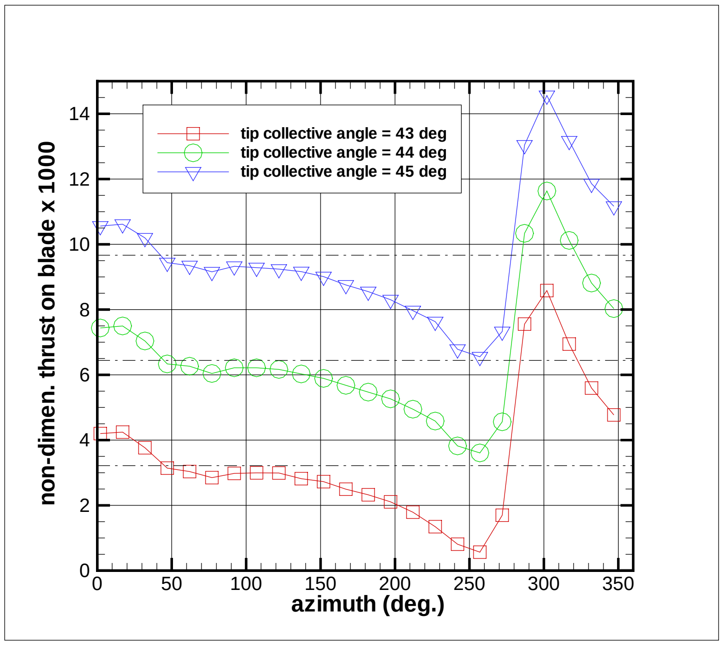


Figure 4.19: Blade thrust (non-dimensional) vs. azimuth in the presence of a wing at 270°. Dash lines indicate the constant thrust level without wing.

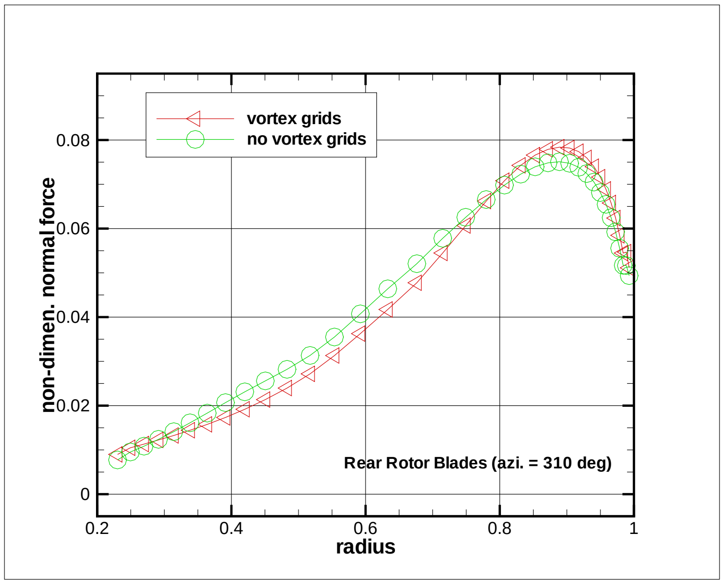


Figure 4.20: Spanwise normal force distribution with and without vortex grids.

0.88R, where the maximum normal force of Fig. 4.17 is. The sudden jump in C_p at the far right is due to the blunt trailing edge of the O-H grid, resulting in a stagnation point there. The larger thrust in the presence of the rear wing, which is evident in the previous figures on normal force, is manifested in the higher pressure at the bottom surface and lower pressure at the top. A small rearward shift in C_p (0.02-0.03 c_{tip} is also seen when vortex grids are used for tracking.

4.5.3 Blade Thrust

A comparison of some thrust values on the rear blades are given in Table 4.1. In general, a difference of about $0-3\%$ can be expected between cases with and without vortex grids. The values in the presence of the rear wing differ considerably from those without the wing (as in Fig. 4.17). Results in line 4 and 2 correspond to the top and bottom figures in Fig. 4.17 respectively.

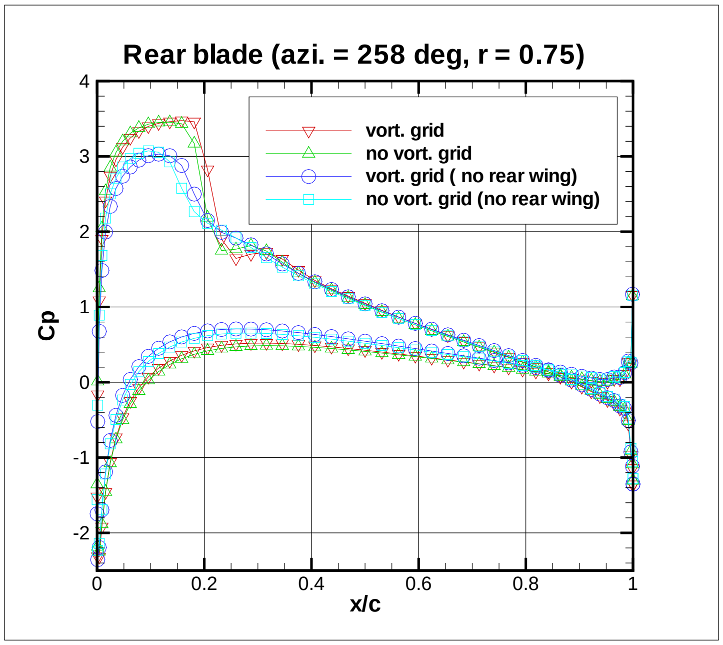


Figure 4.21: Pressure distribution at $r = 0.88$R with and without rear wing and vortex grids.

Blade ($\psi =$)	70°	190°	310°
No wing (no VG)	0.00920	0.00922	0.00985
No wing (VG)	0.00940	0.00899	0.00951
Wing (no VG)	0.00964	0.00832	0.01402
Wing (VG)	0.00970	0.00814	0.01422

Table 4.1: Comparison of forward thrust on the three rear rotor blades. VG – vortex grid.

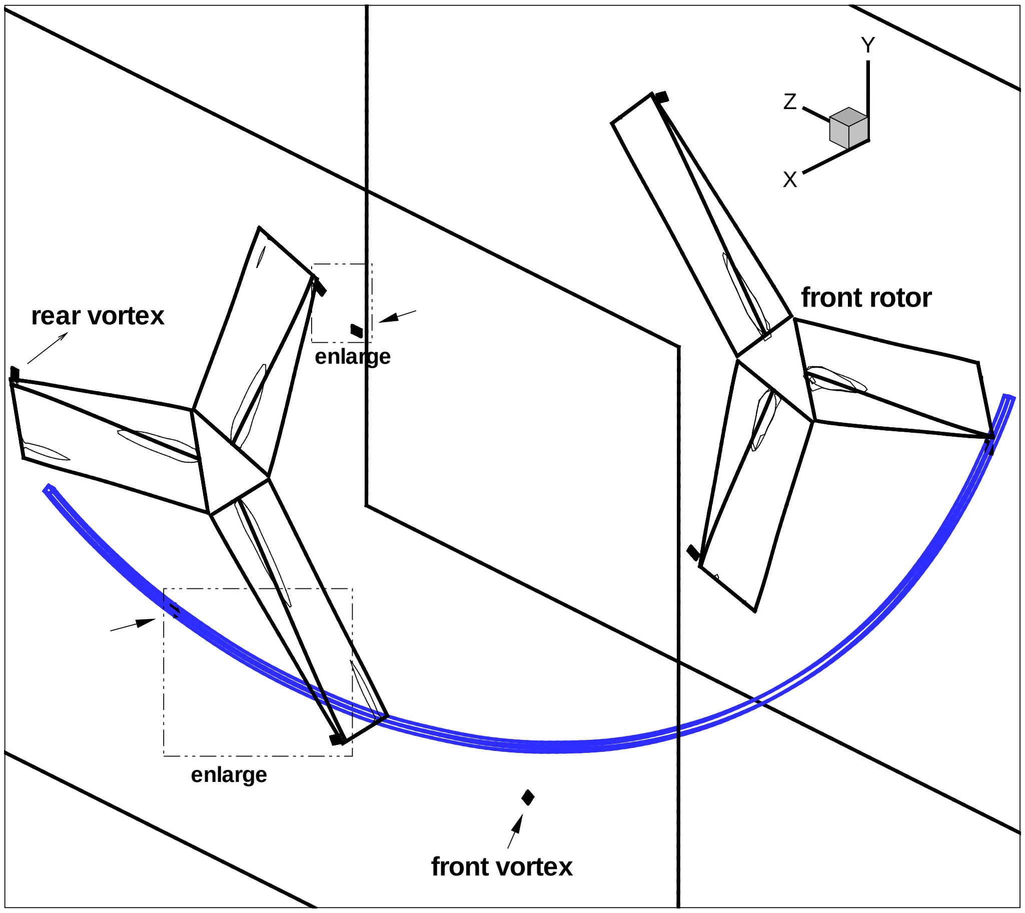
4.5.4 Flow Field and Vortex Contour

Pressure contours of the six blade tip vortices at two streamwise sections immediately behind each rotor are shown in Fig. 4.22 together with two enlarged views. The straight lines at the blades and in the background represent grid section boundaries. The almost constant contour line density of the front vortices near the rear rotor shows that they have suffered very little diffusion. The miss distance (closest encounter with the blade, refer to the bottom right of Fig. 4.22) in this case is 1.39 c_{tip}, at span- and chord-wise location of $r = 0.780$R and $x =$ 0.158c on the upper surface. The strength of the interaction is thus considered weak to mild. The closest distance between the front and rear vortices is 1.18 c_{tip}.

4.5.5 Vortex Trajectory

We lastly examine the trajectory of a vortex as it passes and interacts with the rear rotor and wing. Figure 4.23 plots the trajectory of the vortex that experiences the closest blade encounter in terms of non-dimensional radial position and deviation in the streamwise direction from initial estimated trajectory. The azimuth angle is measured from the twist axis of the front blade. The results for front rotor only (unimpeded convection) are also plotted for comparison and to check for any path deviation due to the presence of the rear rotor-wing assembly. The relative positions of the interacting components are more clearly depicted in the two views of Fig. 4.24.

Neglecting the absolute values of the trajectories, it is observed that the two paths in both plots of Fig. 4.23 are very close for most of the trajectory before about 180°. They start to deviate after that, indicating the vortex is being de-



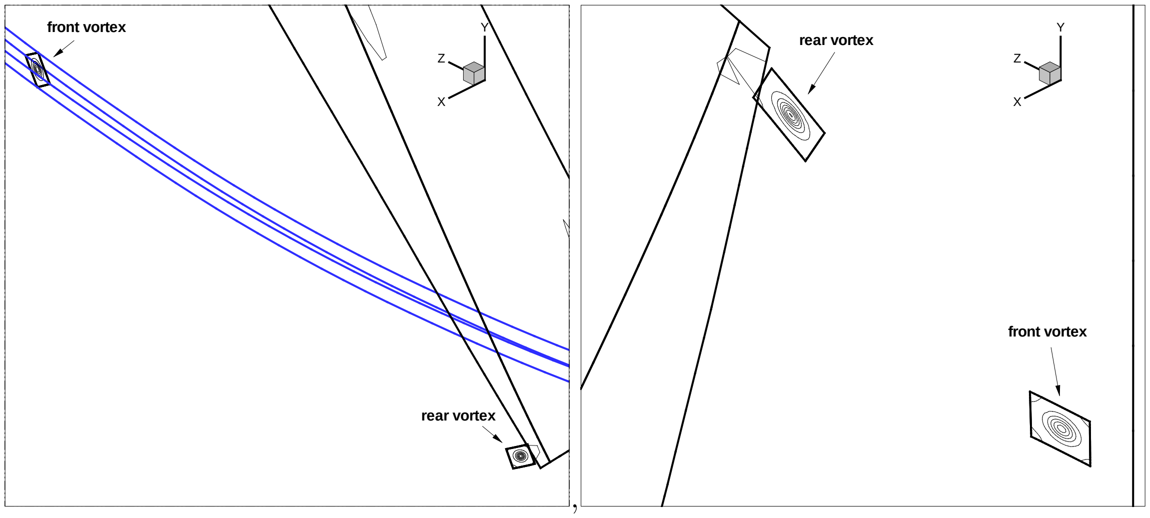
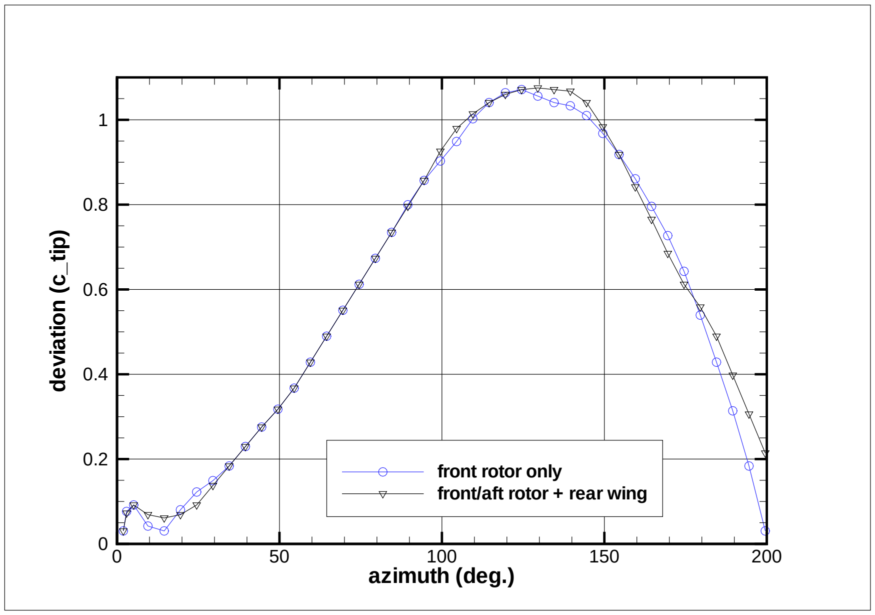


Figure 4.22: Top (left half-span view from behind) – Pressure contour of six tip vortices at two streamwise sections immediately behind the rotors. Bottom – Two enlarged views.



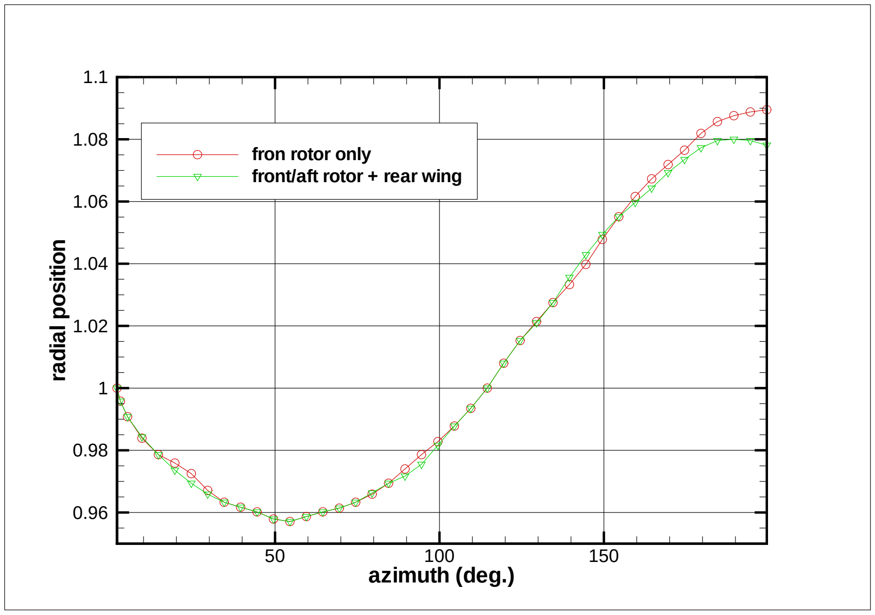


Figure 4.23: Vortex trajectory vs. azimuth from blade of vortex origin. a) stream-wise deviation from initial estimated path. b) radial position.

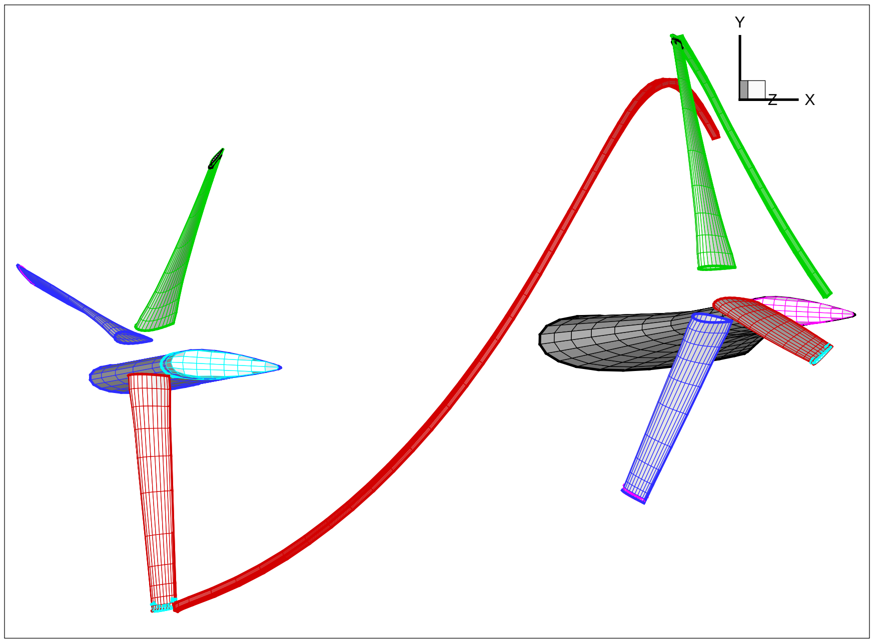
flected by the rear rotor-wing. The deflection from the unimpeded trajectory at 199.5° is 18.4% and 11.4% c_{tip} in the streamwise and radial direction respectively. The closest distance to the rear blade and its vortex, given previously as 1.39 c_{tip} and 1.18 c_{tip}, occur at 149.5° and 168.3° azimuth respectively.

In contrast, the trajectories of the other two front vortices (not shown) experience almost no deviation caused by the presence of the rear rotor-wing. This is easy to understand considering the nearest interacting body or vortex is more than four c_{tip} away.

As a first-order accuracy check of the vortex trajectory, it is noted that the trajectory is determined only by the free stream and blade rotational velocity since the effect of induced velocity and wake contraction in high speed propeller mode is negligible. For M_∞=0.445 and a_∞=1081 ft/s (T_∞=−2.8°C), V_∞ is 481 ft/s. In the time the vortex takes to travel a front-aft rotor separation distance of 38.74 ft, it would have rotated 38.74'/481 × 333rpm/60 × 360° = 161°. This is confirmed both visually in 333rpm 60 visually in Fig. 4.24 and from the adaptation result which gives a value of 160°.

4.5.6 Accuracy of Interpolation

Accuracy of the CFD solution is determined partly by interpolation of the boundary values. The interpolation accuracy requirement for the QTR vortex tracking problem is much less critical than that for the 2D BVI with shock examined in the last section in which solutions are interpolated from inside the vortex and across the shock. Nevertheless, it is still important to conduct at least a first-order check. This can be glimpsed through continuity of the solution and its derivative across external boundaries and hole fringes. This is shown in the



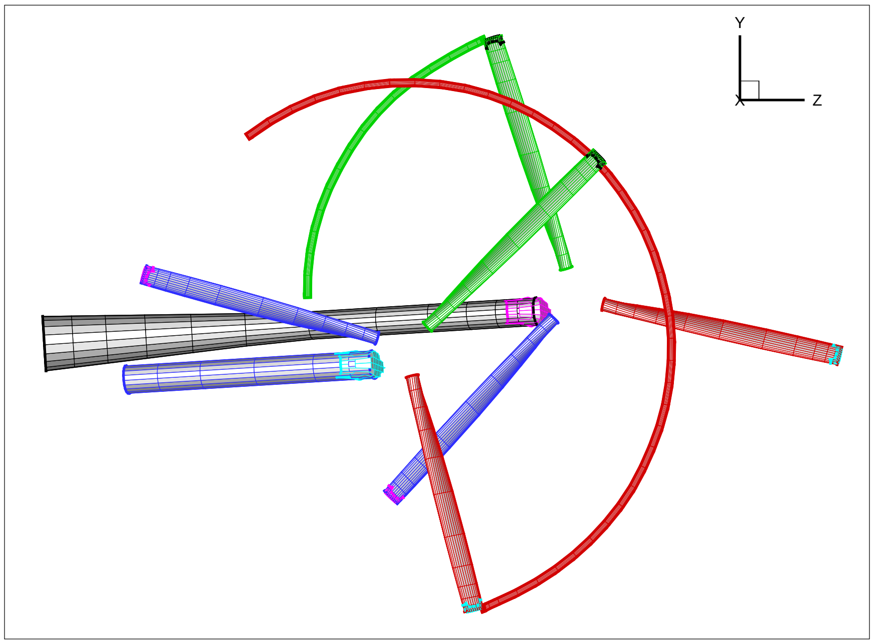


Figure 4.24: Oblique view and front view showing front/aft rotor vortex interaction.

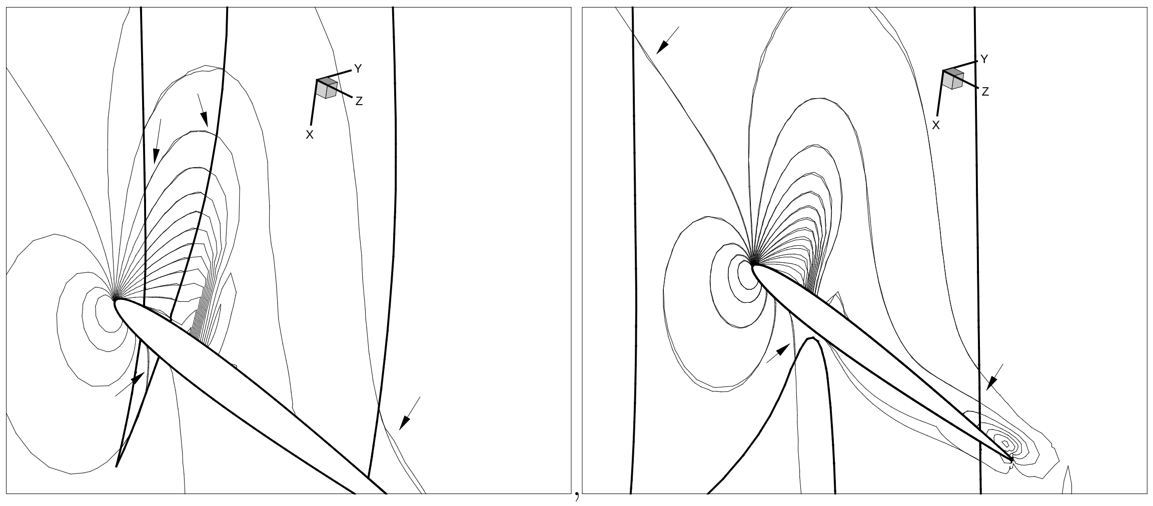
pressure contours of Fig. 4.25. The superimposed lines near the boundary of a grid cut-section (thick lines) indicated by the arrows show that data have been accurately transferred between the grids. The accuracy is seen to be slightly lower at hole fringes where the grid is coarser. Some oscillations can also be observed across the shock. This is due to the non-TVD nature of the BAP limiter. The figure shows good overall interpolation accuracy.

4.6 Some Difficulties with Quasi-Steady Vortex Tracking

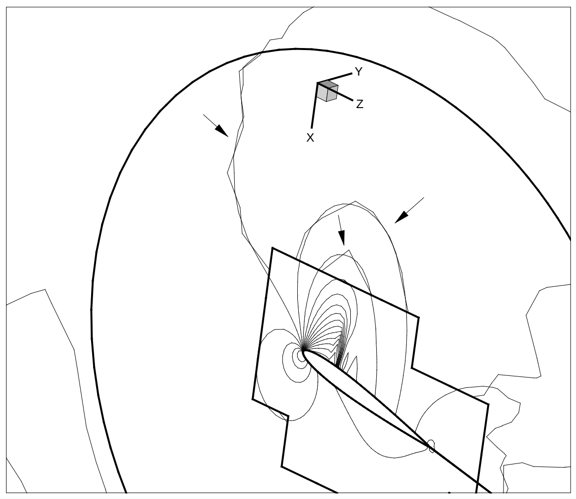
As stated before, vortex capturing and grid adaptation is most conveniently performed in quasi-steady state, which can be thought of as a frozen moment in time. There are, however, some flight conditions for which capturing and tracking in quasi-steady state may pose some difficulties. This section discusses three such cases in QTR, namely, a) a direct hit through the front wing, b) a weak vortex in forward flight and c) a very long vortex grid. While results from the first and last case indicate some positive aspects, those from the second is less so and does not immediately suggest any possible workaround.

4.6.1 Direct Hit

What happens to a vortex behind a solid body after passing through it in a quasi-steady state? This is an interesting question because, as mentioned before, the situation does not exactly exist in this state in the real world. It is, nonetheless, important in the context of vortex tracking. Figure 4.26 shows such a case with the same forward flight condition as in the previous section but a different



a) External boundary interpolation between blade and blade tip grids.



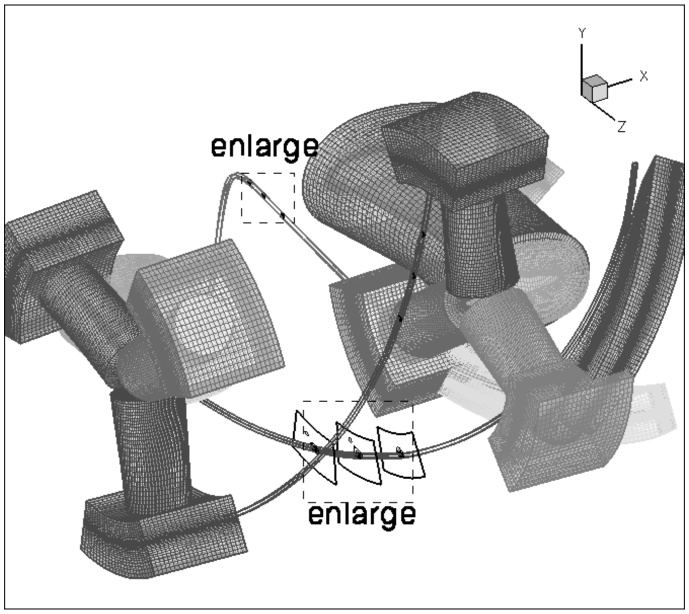
b) Interpolation at the hole fringe of a tip refinement grid from blade tip grid.

Figure 4.25: Accuracy of inter-grid interpolation. Solution/derivative continuity in pressure contour at a blade tip.

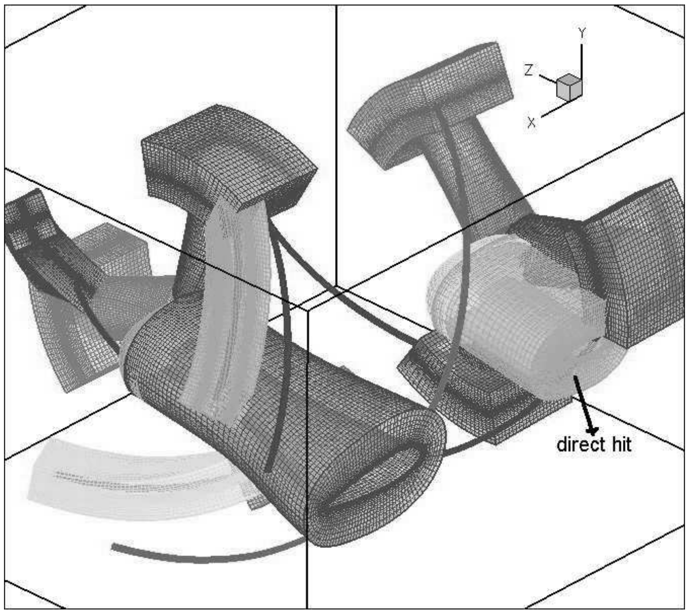
front rotor azimuth such that one of the vortices passes right through the front wing. Enlarged views of this vortex and another mostly freely convecting vortex are shown in Fig. 4.27. For some yet unknown reasons, capturing the vortex emerging from the other side of the front wing proves to be elusive. Repeated adaptations of the vortex grid have failed to capture it definitively. The vortex does not seem to be contented with settling down at the center of the grid.

A much better solution to this problem, if possible, is to avoid having to track a vortex that has experienced such a direct hit. A quasi-steady state is, after all, merely a starting condition for the simulation of a rotating rotor. Starting from a different condition without such strong interaction is more desirable and equally valid. Furthermore, a direct hit in the rotating case is a transient phenomenon lasting only a short period of time and has no appreciable effect on the vorticity downstream of the disrupted vortex. In some problems, however, finding an initial geometric configuration free of such strong interaction may not be easy.

Another plausible solution that has not been attempted is tracking 'on the fly', i.e., while the rotor is rotating. The initial condition can be either impulsive or a quasi-steady state solution. The feasibility and solution behavior of this method for an impulsive start is still unknown. Based on experience gained from quasi-steady vortex tracking, this has its own merits if it works, two of which are described and exemplified in this and the following subsection on weak vortex. This is an area of research ripe for further investigation considering automated vortex tracking for mild vortical interaction is now gradually maturing and in routine use without significant user effort. It is, however, outside the scope of this thesis and will be left for future study.

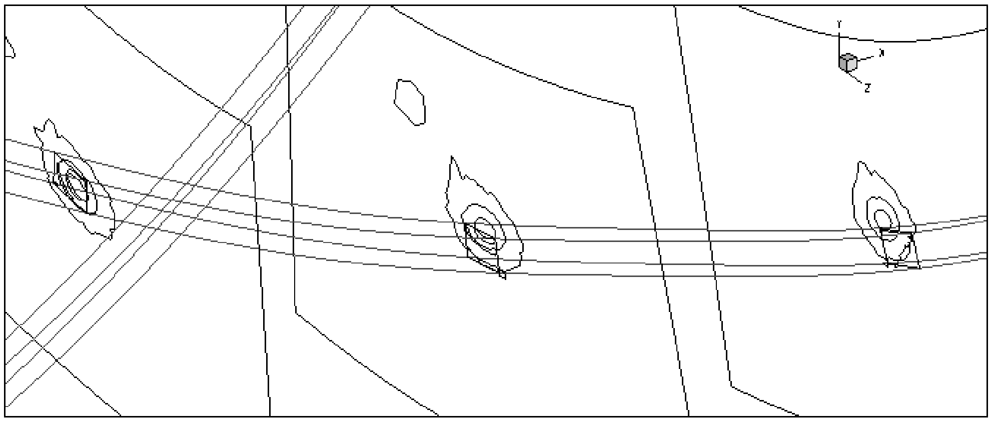


(a) Front oblique view. Outer vortex grids of front rotor not shown.

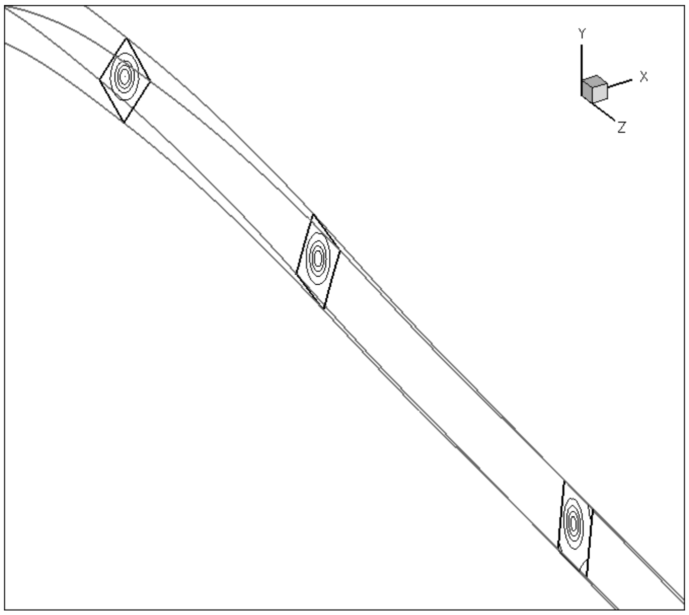


(b) Rear oblique view.

Figure 4.26: A direct hit with the front wing at Mach 0.445.



(a) Vortex after direct hit.



(b) An unimpeded vortex.

Figure 4.27: Vortex pressure contours of the direct hit case.

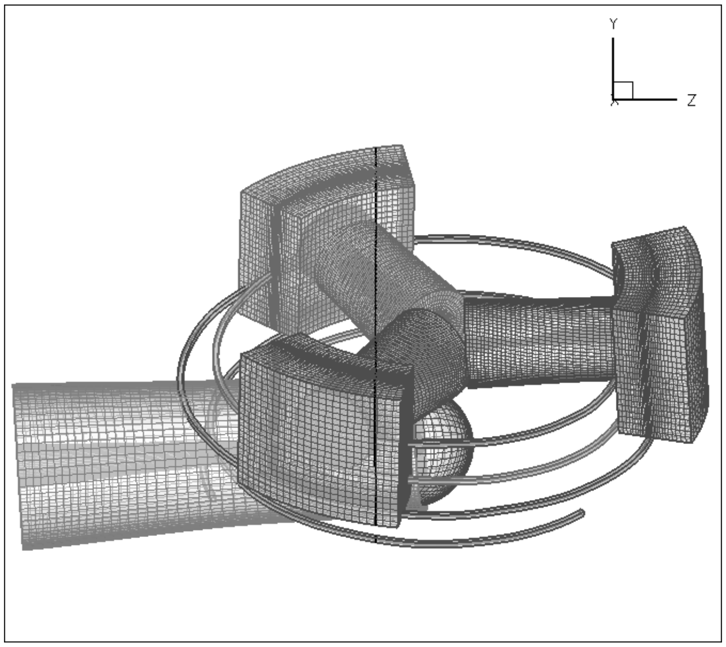
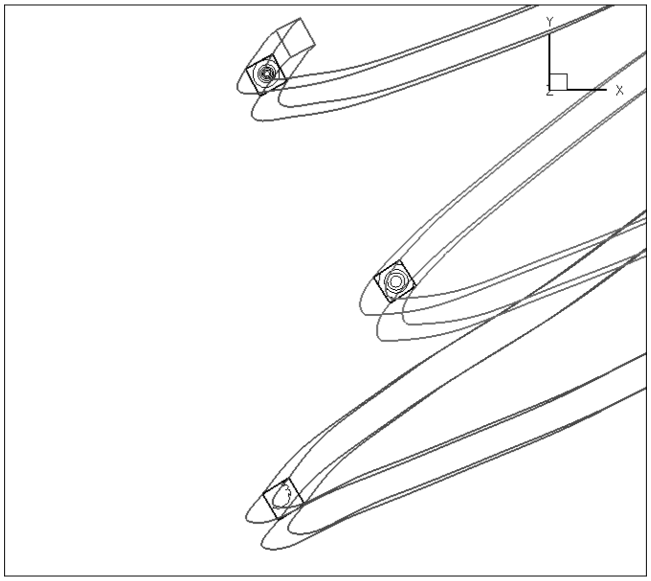


Figure 4.28: Tracking a weak vortex at Mach 0.1 ($\mu = 0.137$), nacelle tilt = 30°. Front view of grids for rear rotor/wing. Outer vortex grids not shown.

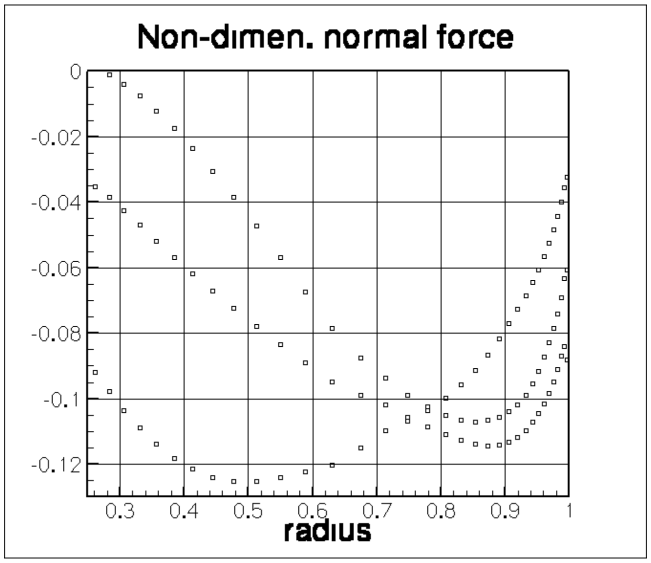
4.6.2 Weak Vortex

Spanwise loading on a rotating blade at certain azimuths in an edgewise forward flight may have a profile that peaks much further inboard than that in an axial flight condition. The loading at the tip in such a profile is usually small. In a quasi-steady state, this results in a weak or no vortex generated at the tip. Tracking such a weak vortex is likely to be more difficult and unreliable due to a relatively noisier background. The problem is particularly acute further downstream where the vortex is more diffused. Vortex tracking, if possible at all, may then increasingly require repeated user intervention for a successful capture.

Figure 4.28 shows such a case for the rear rotor-wing assembly in forward



(a) Pressure contour of the 3 vortices at a cut section indicated by the vertical line in the previous figure shows one of them is weak.



(b) Spanwise loading on the blades also indicates a small loading at the tip for the weak vortex.

Figure 4.29: A weak vortex from a blade on the retreating side.

flight of 64 knots (Mach number = 0.1, $\mu = 0.137$). Rotor tilt is 30°. This nacelle incidence lies somewhere in the middle of the upper and lower limits for conversion protection for the V-22 tilt-rotor. The spanwise loading on one of the blades, as shown in the figure, is seen to have a small loading at the tip even though the total thrust on this blade is higher than the other two. It is also seen from the vortex pressure contour that the weak vortex is barely strong enough to be distinctly recognizable.

Again, this problem is only restricted to vortex tracking in quasi-steady state. Similar to the above direct hit case, the problem is a transient phenomenon and should be avoidable for tracking in an unsteady rotating case.

4.6.3 Long Vortex Grids

In low speed flight, vortices stay close to the rotor and other components for an extended period of time because they are not being swept away as quickly. For the study of interactional aerodynamics, this implies that a much longer vortex needs to be captured than that required in high speed flight. A natural question that arises is then how long can a vortex be tracked and how rapidly does it decay? Another closely related question is how the computational cost is related to vortex grid length?

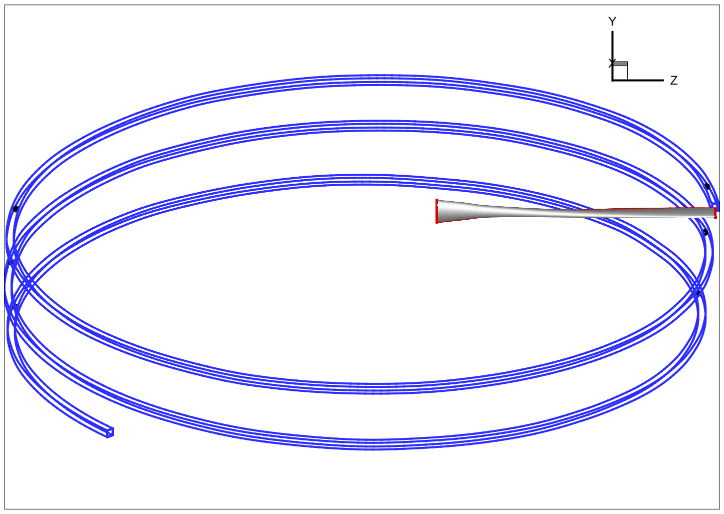
To obtain a basic understanding, a single-bladed rotor in a forward speed of $\mu = 0.137$ (same as the above case) with a vortex grid of length extending for more than 900° is examined. This grid length is equivalent to about 160 c_{tip}. The converged results are shown in Fig. 4.30. It is seen from the figure of the pressure contours that the vortex appears to remain sharp and suffer little diffusion after such long convection. Figures for the decay in pressure at the center and peak-to-peak swirl velocity show that diffusion after about 2 1/2 revolutions is around 30%. This is a sufficiently small amount that should not cause accuracy degradation in high fidelity vortex interactional aerodynamics.

The gradual rate of vortex decay appears to be steady and does not seem to indicate any obstruction to the use of an even longer vortex grid. The only impediment in a real application is computational cost, which increases non-linearly with grid length. This is understandable since a traveling wave in the solution after an adaptation would need more time iterations to traverse the entire length, reflect off the ends of the grid and eventually settle down. This problem manifests itself in the longer time required for convergence.

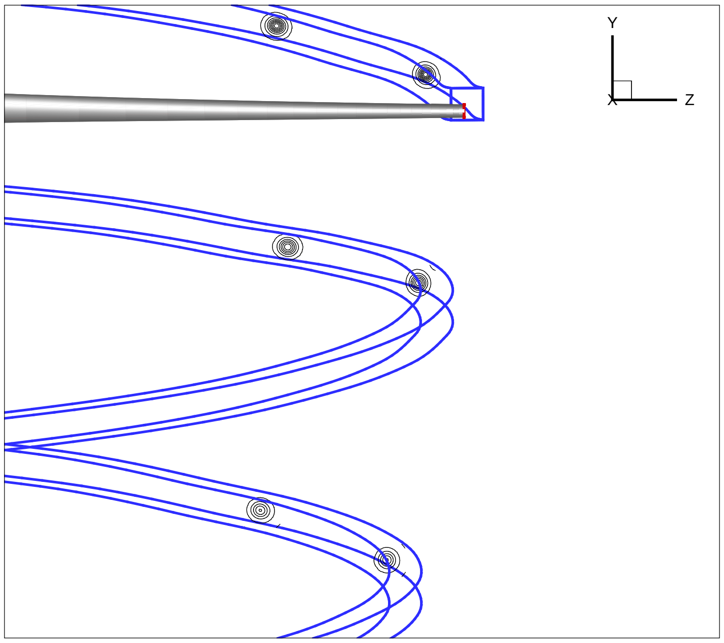
It should be noted here the potentially misleading comparison of the total length of vortex convection as given here with that often cited for an axisymmetric multi-bladed hovering rotor in which only a single blade is modeled with periodic boundary conditions on the two opposite sides of a wedge-shaped grid. For the latter case, a two-revolution convection for a 4-bladed rotor, for example, is really only 2/number of blades = 1/2 revolution. There is no vortex convection and thus diffusion for the other 1 1/2 revolutions since solutions on one periodic boundary are directly injected into the other.

4.6.4 Numerical Sensitivity of Vortices

A significant problem with high-resolution vortex-tracking grids is that downstream trajectory is very sensitive to upstream results and certain numerical parameters (i.e., limiter, order of accuracy, interpolant etc.). This implies that a small disturbance or error upstream could be increasingly magnified downstream. Because this has a large effect on the predicted BVI which may occur at

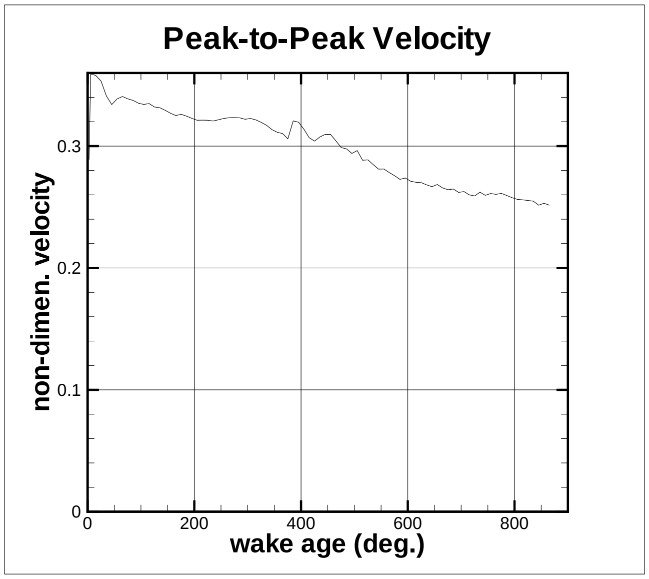


(a) Long vortex grid extends for about 900°.

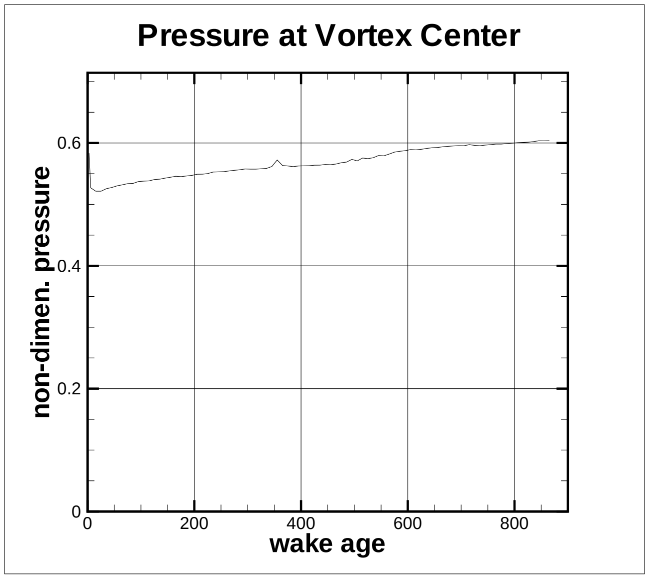


(b) Pressure contour of vortex shows little diffusion.

Figure 4.30: Tracking a single long vortex at Mach 0.1 ($\mu = 0.137$), nacelle tilt $= 8°$.



(a) Vortex diffusion (swirl velocity).



(b) Vortex diffusion (pressure).

Figure 4.31: Vortex diffusion with wake age.

Case	Time Integration	Spatial accuracy	Limiter	Vortex interp.
Baseline	Implicit O(1)	2nd-order	BAP	normal
C1	Implicit O(1)	3rd-order	BAP	normal
C2	Implicit O(1)	3rd-order	none	normal
C3	Explicit O(3)	3rd-order	BAP	normal
C4	Implicit O(1)	2nd-order	BAP	frm BackGr
C5	Implicit O(1)	2nd-order	BAP	frm BackGr

Table 4.2: Solver parameters (for background grids only) employed for trajectory sensitivity study.

some distance from the blade tip, the solver must be capable of predicting the upstream trajectory very accurately. In contrast, force related quantities such as thrust, pressure etc., are not very sensitive to these parameters.

It is instructive to conduct a limited study on this sensitivity issue for a low speed forward flight. The problem in Fig. 4.28 for a weak vortex is revisited and used as a test problem with the difference that each vortex is now assigned three levels of nested grids. It is re-computed here for different input parameters and numerical schemes. The various options selected for this study are given in Table 4.2. Case C4 and C5 are the same except that all hole points of the background grids (those inside bodies) in C5 are not updated every time step (i.e., `dQ = 0.`). Note that this updating is irrelevant for explicit time integration.

Figure 4.32 shows the deviation in the trajectory of one of the three vortices from the baseline result. Several parameters, e.g. time integration, spatial accuracy, limiter, interpolant etc., are changed to gauge the sensitivity of the predicted trajectory. It is observed that the choice of donor grids, i.e., the in-

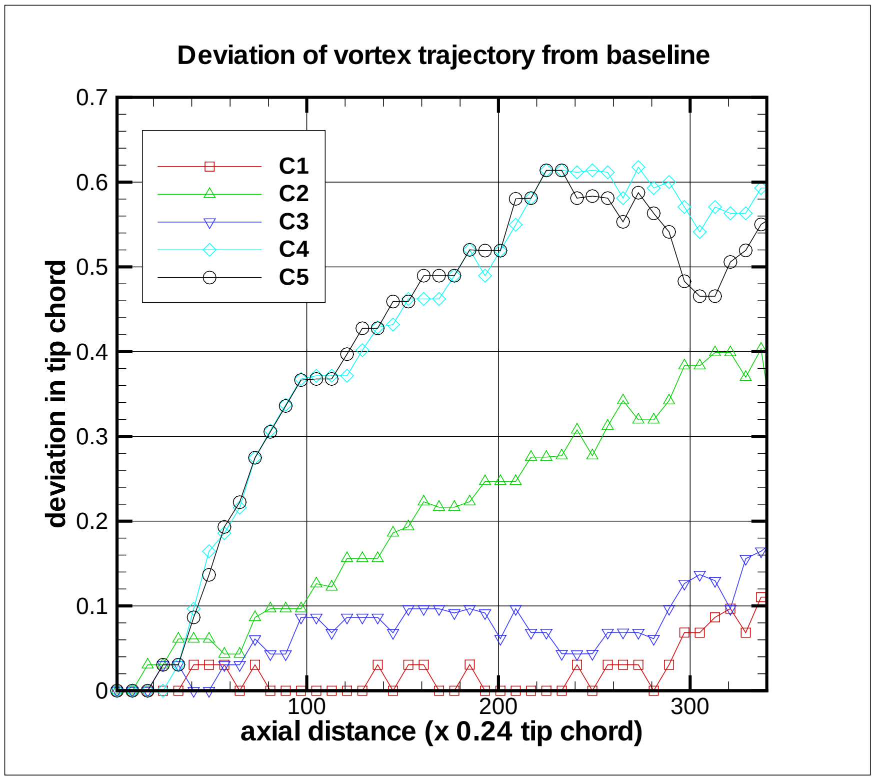


Figure 4.32: Deviation of predicted vortex trajectory from baseline.

terpolant, for the boundary points has the largest effect on the downstream trajectory (C4 and C5). The deviation from the baseline result is 0.6 tip chord after about one revolution. Temporal and spatial order have a much smaller effect of < 0.1 chord as indicated by the two curves at the bottom (C1 and C3). Comparison between C4 and C5 (top two curves) also shows that updating the hole points of the background grids has only a moderate effect far downstream after the first revolution.

Chapter 5

Conclusion

This thesis has developed two new numerical techniques that are very useful in the simulation of high fidelity vortex capturing and vortical interactional aerodynamics for rotorcraft. The first is a simple, general and fundamentally different approach to the problem of overset grids connectivity for the numerical solutions of PDE. The second is a technique to automatically capture and track rotating vortices and subsequent vortical interactions using very-high resolution yet economical nested helical grids. While the underlying problem in the former is fundamental and of general interest to computational science, the latter is specific to problems in rotorcraft CFD and is still in its infancy.

Some of the major conclusions in this thesis are summarized in the following.

5.1 Overset Grids Connectivity

1) Of the several advantages of the IHC overset grids connectivity concept given in this thesis, the most significant is its inherent simplicity. This results in a compact algorithm with about an order of magnitude lower count of source line of code (SLOC) compared with existing codes. More importantly, the simplicity

increases robustness through more automation and by avoiding extra code logic to handle new exceptions.

2) Holes in the grids that overlap with bodies of arbitrary shape are not explicitly cut in the algorithm. This explicit hole cutting process is the most difficult step in the existing approach. Holes arise naturally as a by-product of the cell size comparison process. They are also inherently optimum for numerical accuracy without requiring another step in the routine.

3) Mostly because of the absence of any required overhead in the hole cutting and donor cell searching routines, the execution speed of the IHC code, after limited comparison with several existing codes, is 3-10 times faster in the first time step. However, similar executions for subsequent time steps in a moving grid problem are much faster for all codes because the starting shape and extent of the holes are already known and the change at every time step is incremental. The IHC speed advantage in all subsequent time steps is thus understandably diminished.

4) Difficulty of the IHC algorithm at body surface is inherently non-existent and no user input or hole-cutting expertise is required. This increases robustness and is especially important for unsteady moving body cases where it would be impossible to check and modify input at each time step.

5) All tested problems thus far indicate that this new concept does not seem to have any apparent comparative disadvantage or deficiency.

6) High-order (cubic) monotone interpolation for boundary points, which is still uncommon among existing overset CFD codes, carries a disproportionately higher cost (4-6 times on the Cray X1E depending on the relative number of boundary points) than the more common linear interpolant. However, the cost

could be justified in cases where the interpolation point is near a vortex core and vortex preservation is important.

5.2 2D Blade Vortex Interaction

A simulation of the well-investigated unsteady 2D BVI problem using vortex tracking grids was performed on a NACA 0012 airfoil and a vortex with a non-dimensional strength of 0.2 and a core radius of 0.05 chord. This acts as an initial test bed of vortex tracking for the general 3D implementation.

1) This problem can now be simulated at a sufficiently low cost of only 8 minutes on the Alpha 667 MHz processor using overset vortex grids such that various parametric studies (various limiters, time integration schemes, oscillating airfoils etc.) can be conveniently performed for an in depth investigation of this problem.

2) The interpolation order of accuracy and monotonicity was found to be critically important for the preservation of the vortex structure after the interaction with the airfoil. Cubic interpolation is able to clearly capture the distorted vortex profile which the linear interpolation almost completely misses. The latter also results in a predicted vortex swirl velocity that is about 25-30% lower.

3) The salient features of the problem were well captured. These include the λ-type shock, acoustic wave propagation, vortex distortion due to interaction with airfoil and long duration of the after-effect.

4) Reasonable agreement was achieved with experimental measurement of instantaneous lift and moment history as well as other qualitative flow features.

5) The calculated minimum lift coefficient (and C_l during and after inter-

action), remains a sensitive quantity of questionable accuracy. This value is calculated to be 0.185 here, and ranges from 0.18 to 0.25 as reported by different researchers.

6) The most glaring issue is the accuracy of the shock-vortex interaction and the inconsistency and sensitivity of the results as reported by various researchers, although the trends are in reasonable agreement.

7) C_l is found to be quite sensitive when comparing the results using two different parameters, namely minmod and the higher-order UNO limiters. The C_l with minmod is about 15% higher when the vortex is two chords downstream and 10% lower when it is two chords upstream of the leading edge.

8) The multiple holes in the 3- or 4-grid system for this 2D problem are found to be optimally cut at all instances using the new IHC connectivity approach. In addition, the contour lines are also smooth across the hole boundaries.

9) This 2D BVI problem provides an excellent testbed for checking the high accuracy of a compressible flow solver because of its geometric simplicity and the presence and intense interaction of a shock and a vortex.

5.3 Vortex Tracking in Rotorcraft CFD

A half-span Quad-Tilt-Rotor-like configuration serves as the platform for the development, testing and investigation of automated vortex tracking. All geometric dimensions (except airfoil sections) are taken from the V-22 tilt-rotor. The tracking of a pair of co-rotating vortices is also being tested for robustness.

1) Initial study of vortex tracking using specially designed long nested vortex grids suggests its feasibility as a practical investigative tool.

2) The most immediate problem in the development of a practical vortex tracking methodology that was successfully solved in this work is the automation of the tracking process. Tracking the vortex and adapting their grids manually is very labor intensive and requires frequent user attention. The automated process devised in this thesis works fine for weak to mild vortical interaction without any user intervention. This may need minor modification for strong interaction.

3) The difficulty of vortex tracking lies in the implementation and its related issues such as, among others, grid adaptation, unambiguous vortex detection, vortex convergence, tracking automation and adaptation acceleration. Some of these were not anticipated at the outset. They arised in the course of investigation of the initial failures of this methodology. Implementation of this methodology for mild vortical interactions in quasi-steady state is successful.

4) The tracking of a pair of co-rotating vortices and their grid adaptation, which was basically a 2D process for a cross section, were performed without major difficulty. The vortices were very well preserved with practically no decay after a convection distance of 1500 core radii, except that at that distance they were showing some signs of instability.

5) Vortex tracking was demonstrated on the front/aft-rotor interactional aerodynamics of a Quad-Tilt-Rotor-like aircraft in a steady high speed forward flight of Mach 0.445. All six blade tip vortices from the front-aft rotor pair were simultaneously and automatically tracked and captured in quasi-steady state. A mild interaction of a front vortex with a rear blade was simulated and the vortex trajectory was visibly deflected 0.2 c_{tip} by this interaction.

6) Expectedly, the pitch angle of 46.4° at 75% R for a steady flight of Mach 0.445 was slightly higher than the published results of 45.1° produced by OVER-

FLOW using cambered XN-series instead of the symmetric NACA00xx airfoils used in this work.

7) As a first check of tracking accuracy, vortex trajectory extracted from the adaptation result is consistent with that calculated from the combined free stream and blade tip rotational velocity of Mach 0.445 and 663 ft/s respectively.

8) Interpolation accuracy as revealed in the continuity and smoothness of the solution and its derivative across grid boundaries was found to be satisfactory, particularly at the blade tip and vortex core region.

9) The vortex tracking method was also tested to its limit in quasi-steady state under three flight conditions that are difficult to simulate: i) a direct hit of the vortex through the wing, ii) a weak vortex in edge-wise forward flight, and iii) a very long vortex grid. In the case of i) a direct hit, the vortex downstream of the hit, for some yet undetermined numerical reason, could not be perfectly contained in the core even after repeated tracking.

10) Tracking a weak vortex in forward flight (ii) may become more problematic the further downstream the vortex is. Tracking it on the 'fly', a future research extension, may possibly be the only solution.

11) The results from the test with very long vortex grids (iii) was encouraging. With a very fine grid spacing of 1% c_{tip} at the core, each nested vortex grid contains only about 300K−500K points per revolution depending on the blade aspect ratio. This core resolution is sufficient to preserve the vortex and reduce diffusion to about 30% decay for a convection distance of 160 c_{tip} or about 2.5 revolutions for the QTR rotor. This is achieved using a spatial scheme with only 3^{rd}-order accuracy, compared with 7^{th}-order or higher generally agreed to be necessary.

12) For the numerical sensitivity study, the downstream trajectory of the predicted high-resolution vortex is found to be very sensitive to some parameters, the most significant of which is the interpolants for the boundary points of the vortex grids. Its deviation is on the order of one tip chord length after a convection distance of one revolution. Other parameters such as temporal and spatial order of accuracy and non-updating of hole points have a smaller effect.

13) The most important and fundamental conclusion that can be drawn from this work is that automated vortex tracking and interaction in rotorcraft CFD is feasible and can be made practical, although some problems still need to be successfully overcome. The methodology is still in its infancy. Considering the potentially large payoff, it is now ripe for further research and development.

5.4 Key Contributions

The major contributions of this work are the development of two new numerical techniques. The first one is a whole new approach to overset grids connectivity, i.e., cutting holes in grids. It significantly alleviates two of the major deficiencies in existing codes, namely complex logic/algorithm and high level of expertise required of the users. This results in a simple, efficient and robust approach.

The second technique, automated vortex tracking, is specific to rotorcraft CFD. Here the nested vortex grids design allow a very compact and high-resolution core with only a small number of grid points. The grid axis is also aligned with the vortex axis, thus minimizing the skewness problem. These advantages, however, come with the price of vortex tracking and its associated difficulties some of which this work has successfully overcome.

Another contribution in this work concerns piecewise cubic Hermitian interpolation in overset grids. This interpolant has proven to be significantly more vortex-preserving than its more common linear counterpart.

5.5 Future Work

Overset Grids Connectivity

The new concept for overset grids connectivity given in this thesis is equally applicable to unstructured grids. Two modifications to the algorithm, however, are required. Stencil walking in the donor search step is no longer an increment or decrement of one for the index. The neighboring cell number is available in the cell connectivity array. Another modification is required when the search for donor cell steps outside the computational domain or crosses into a solid body. This is also checked from the cell connectivity array in a similar way. Since unstructured grids is not absolutely necessary for vortex tracking, this extension is placed lower in the priority list.

The calculation of interpolation coefficients using Newton-Raphson iterations are relatively expensive. Efficiency of the connectivity code can be further improved if initially only the cross and dot product test is used, without the interpolation coefficients, to determine the approximate indices of all possible donor cells. The coefficients are calculated only after the best donor cell has been identified. This requires some non-trivial rearrangement of the code logic.

Other planned work include:

-*Automatic detection of untrimmed grids.* When some region of a wall is replaced or altered by a protrusion, the original body-conforming grid does not necessarily have to be redesigned. This grid, with part of the wall missing, is now called an untrimmed grid. The new protruded wall (and its associated grid) causes a hole in the untrimmed grid that extends all the way to the wall surface surrounding the protrusion. The automatic detection of untrimmed grids and the protruded wall grids is a more challenging problem that requires some careful study and testing. It is also one of the last remaining hurdles for complete transparency for problems with untrimmed grids. It is, however, not critical for users who can visually identify these grids quickly.

-*Increase robustness in the detection of highly concave boundaries.* Stepping out of a highly concave grid boundary in the search path does not necessarily imply no donor cell is in the grid. Preferably, the code is able to cross the gap (possibly through some ghost cells) to reach the other side and continue the search. Similar to the above, this is another automatic geometric feature identification. It is, however, more difficult to identify visually and specify as special input.

-*More test cases for documentation of robustness and efficiency.* This is a necessary step in order to increase the confidence level in the maturation of an algorithm.

Vortex Tracking in Rotorcraft CFD

The most important and challenging future work for the vortex tracking methodology is the extension to unsteady rotating blades. Unlike other planned work outlined in this section, it also has the most uncertain outcome. Experience with quasi-steady state vortex tracking to date does not unambiguously reveal the likelihood of success. The most immediate question is the possibility of capturing the vortex far downstream since it has been found that a downstream vortex requires a more stable flow environment and a converging solution to form and strengthen in a quasi-steady state. A related question is the effect of time-varying grid metrics on the magnitude of vortex dissipation. But as mentioned in the last section of Chapter 4, this is probably the only way to overcome some of the difficulties encountered in quasi-steady state vortex tracking. It is also a major undertaking laden with uncertainties.

Another problem that needs more attention is the unambiguous detection of the correct vortex when two vortices are in the same neighborhood. This problem has been encountered in hover computation in which the vortex grid immediately after the first blade passage mistakenly tracks the vortex from the wrong blade. Visual detection and manual adjustment is only a temporary solution. This problem is a subset of the more general issue of automated tracking for strong vortical interaction in which either the vortex profile is occasionally but severely distorted or its trajectory passes through a solid body.

To gain more confidence of the computation, grid independence of the solution is another issue to be confronted with. A minor study of grid refinement of the two background grids has yielded inconclusive evidence. This, as well as the refinement of the near-body grids, needs to be revisited and more carefully studied in the near future.

Perhaps the most interesting future problem for research and study is vortical interactional aerodynamics. This could be interaction with tail rotor, itself or airframe under any flight condition. This problem can only be generally tackled after the above problems are solved.

Lastly, the remaining issue of numerical accuracy in the 2D BVI problem of

Chapter 3 must not be forgotten. Despite the simplicity of the problem setup compared with other 3D problems, this issue is probably the most stringent accuracy test for a compressible CFD code.

These possible future research topics are proposed on the basis of limited experience to date with vortex tracking methodology. Other candidate problems would almost certainly arise as more is learned about this methodology. It can be seen from the above proposed work that vortex tracking is a rich and young research topic in rotorcraft CFD providing many more years of challenging research and study.

Bibliography

[1] K. Duraisamy, "Studies in Tip Vortex Formation, Evolution and Control," *Ph.D. Dissertation, 2005, Department of Aerospace Engineering, University of Maryland.*

[2] J. L. Steger, F. C. Dougherty and J. A. Benek, "A Chimera Grid Scheme," *Advances in Grid Generation*, K. N. Ghia and U. Ghia (eds.), ASME FED, Vol. 5, 1983, pp. 59−69.

[3] A. Kageyama, M. Kameyama, S. Fujihara, M. Yoshida, M. Hyodo and Y. Tsuda, "A 15.2 TFlops Simulation of Geodynamo on the Earth Simulator," SC2004 High Performance Computing, Pittsburgh PA, Nov. 6−12, 2004.

[4] H. A. Dwyer, H. Nirschl and V. Denk, "Heat, Mass and Momentum Transfer about Arbitrary Groups of Particles," 25th International Symposium on Combustion, Irvine CA, July 1994.

[5] D. Barnette and C. Ober, "Progress Report on High-Performance High Resolution Simulation of Coastal and Basin-Scale Ocean Circulation," Proceedings of the 2nd Overset Composite Grid Solution and Technology Symposium, Fort Walton Beach FL, 1994.

[6] N. E. Suhs, S. E. Rogers and W. E. Dietz, “PEGASUS 5: An Automated Pre-Processor for Overset Grid CFD,” AIAA Paper 2002−3186, 32nd AIAA Fluid Dynamics Conference, St. Louis MO, June 24−26, 2002.

[7] R. Meakin, “Object X-Rays for Cutting Holes in Composite Overset Structured Grids,” AIAA Paper 2001−2537, 15th AIAA Computational Fluid Dynamics Conference, Anaheim CA, June 11−14, 2001.

[8] N. C. Prewitt, D. M. Belk and Wei Shyy, “Parallel Computing of Overset Grids for Aerodynamic Problems with Moving Objects,” *Progress in Aerospace Sciences*, Vol. 36, 2000, pp. 117−172.

[9] D. L. Brown, W. D. Henshaw and D. J. Quinlan, “Overture: Object-Oriented Tools for Overset Grid Applications,” AIAA Paper 99−3130, 15th AIAA Applied Aerodynamics Conference, June 1999.

[10] G. Chesshire and W. D. Henshaw, “Composite Overlapping Meshes for the Solution of Partial Differential Equations,” *J. Comp. Phys.*, Vol. 90, 1990, pp. 1−64.

[11] Z. J. Wang and V. Parthasarathy, “A Fully automated Chimera Methodology for Multiple Moving Body Problems,” *Int. Jour. for Num. Meth. in Fluids*, Vol. 33, 2000, pp. 919−938.

[12] N. A. Petersson, “An Algorithm for Assembling Overlapping Grid Systems,” *SIAM J. Sci. Comput.*, Vol. 20, No. 6, 1999, pp. 1995−2022.

[13] N. A. Petersson, “Hole-Cutting for Three-Dimensional Overlapping Grids,” *SIAM J. Sci. Comput.*, Vol. 21, No. 2, 1999, pp. 646−665.

[14] R. Noack, “DiRTlib: A Library to Add an Overset Capability to your Flow Solver,” AIAA Paper 2005−5116, 17th AIAA Computational Fluid Dynamics Conference, Toronto Canada, June 6−9, 2005.

[15] R. Noack, “SUGGAR: A General Capability for Moving Body Overset Grid Assembly,” AIAA Paper 2005−5117, 17th AIAA Computational Fluid Dynamics Conference, Toronto, Canada, June 6−9, 2005.

[16] T. W. Roberts and E. M. Murman, “Solution Method for a Hovering Helicopter Rotor using the Euler Equations,” AIAA paper 85−0436, 1985.

[17] B. E. Wake and N. L. Sankar, “Solutions of the Navier−Stokes Equations for the Flow about a Rotor Blade,” *J. American Helicopter Soc.*, Vol. 34, 1989, pp. 13−23.

[18] B. E. Wake and J. D. Baeder, “Evaluation of a Navier−Stokes Analysis Method for Hover Performance Prediction,” *J. American Helicopter Soc.*, Vol. 41, No. 1, 1996, pp. 7−17.

[19] G. R. Srinivasan and J. D. Baeder, “TURNS: A Free-Wake Euler / Navier−Stokes Numerical Method for Helicopter Rotors,” *AIAA Journal*, Vol. 31, No. 3, 1993, pp. 959−962.

[20] R. C. Strawn and M. J. Djomehri, “Computational Modeling of Hovering Rotor and Wake Aerodynamics,” 57th Anual Forum of the American Helicopter Society, Washington D.C., May 9−11, 2001.

[21] M. A. Potsdam and R. C. Strawn, “CFD Simulation of Tiltrotor Configurations in Hover,” 58th Anual Forum of the American Helicopter Society, Montreal Canada, June 11−13, 2002.

[22] H. J. Kang and O. J. Kwon, "Unstructured Mesh Navier−Stokes Calculations of the Flow Field of a Helicopter Rotor in Hover," *J. American Helicopter Soc.*, Vol. 47, No. 2, 2002, pp. 90−99.

[23] P. Beaumier, E. Chelli and K. Pahlke, "Navier−Stokes Prediction of Helicopter Rotor Performance in Hover Including Aeroelastic Effcts," *J. American Helicopter Soc.*, Vol. 46, No. 4, 2001, pp. 301−309.

[24] C. Benoit and G. Jeanfaivre, "Three-Dimensional Inviscid Isolated Rotor Calculations using Chimera and Automatic Cartesian Partitioning Methods," *J. American Helicopter Soc.*, Vol. 48, No. 2, 2003, pp. 128−138.

[25] P. Renzoni, et al. "EROS − A common European Euler code for the Analysis of the Helicopter Rotor Flowfield," *Progress in Aeospace Sciences*, Vol. 36, 2000, pp. 437−485.

[26] M. M. Rai, "Navier−Stokes Simulation of Blade Vortex Interaction using High Order Accurate Upwind Schemes,"AIAA Paper 87-0543, 25th Aerospace Sciences Meeting, Reno NV, Jan. 12−15, 1987.

[27] W. S. Oh, J. S. Kim and O. J. Kwon, "Numerical Simulation of Two-Dimensional Blade Vortex Interactions using Unstructured Adaptive Meshes," *AIAA Journal*, Vol. 40, No. 3, 2002, pp. 474−480.

[28] W. S. Oh, J. S. Kim and O. J. Kwon, "Time-accurate Navier−Stokes Simulation of Vortex Convection using an Unstructured Dynamic Mesh Procedure," *Computers & Fluids*, Vol. 32, 2003, pp. 727−749.

[29] S. Lee and D. Bershader, "Head-on Parallel Blade Vortex Interaction," AIAA Journal, Vol. 32, No. 1, 1994, pp. 16−22.

[30] N. S. Hariharan and L. N. Sankar, "First-Principles Based High Order Methodologies for Rotorcraft Flowfield Studies," 55th Anual Forum of the American Helicopter Society, Montreal, Canada, May 25−27, 1999.

[31] R. E. Spall, "Numerical Study of a Wing-Tip Vortex using the Euler Equations," *Journal of Aircraft*, Vol. 38, No. 1, 2001, pp. 22−27.

[32] T. A. Egolf, B. E. Wake and C. Berezin, "Recent Rotor Wake Simulation and Modeling Studies at United Technologies Corporation (Invited)," AIAA Paper 2000−0115, 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV, Jan. 2000.

[33] R. L. Meakin, "On Adaptive Refinemeneand Overset Structured Grids," AIAA paper 97−1858, 13th AIAA Computational Fluid Dynamics Conference, Snowmass CO, June 1997.

[34] Bagai, A. and Leishman, J. G., "Rotor Free-Wake Modeling using a Relaxation Technique - Including Comparisons with Experimental Data," *Journal of the American Helicopter Society*, Vol. 40, No. 2, April 1995.

[35] Bagai, A. and Leishman, J. G., "Rotor Free-Wake Modeling using a Psuedoimplicit Relaxation Algorithm," *Journal of Aircraft*, Vol. 32, No. 6, 1995, pp. 1276−1285.

[36] Bagai, A. and Leishman, J. G., "Free-Wake Analysis of Tandem, Tilt- Rotor and Coaxial Rotor Configurations," *Journal of the American Helicopter Society*, Vol. 41, No. 3, 1996, pp. 196.

[37] Bhagwat, M. J. and Leishman, J. G., "Stability ,Consistency and Convergence of Time Marching Free-Vortex Rotor Wake Algorithms," *Journal of the American Helicopter Society*, Vol. 46, No. 1, 2001, pp. 59−71.

[38] Bhagwat, M. J. and Leishman, J. G., "Accuracy of Straight-Line Segmentation Applied to Curvilinear Vortex Filaments," *Journal of the American Helicopter Society*, Vol. 46, No. 2, 2001, pp. 166−169.

[39] Bhagwat, M. J. and Leishman, J. G. "Rotor Aerodynamics in Maneuvering Flight Using a Time-Accurate Free-Vortex Wake," *Journal of the American Helicopter Society*, Vol. 44, No. 2, April 1999, pp. 109−120.

[40] Leishman, J. G., Bhagwat, M. J. and Bagai, A., "Free-Vortex Methods for Helicopter Rotor Wake Analyses, (Invited Paper)" *Journal of Aircraft, Special Edition on Rotorcraft Wakes*, Vol. 39, No. 5, 2002, pp. 759−775.

[41] E. D. Snyder, "The Quad Tilt Rotor: Its Beginning and Evolution," 56th Anual Forum of the American Helicopter Society, Virginia Beach VA, May 2−4, 2000.

[42] J. J. Corrigan, R. L. Bennet and P. Y. Hseih, "Copter 2000: The QTR and Beyond," 57th Anual Forum of the American Helicopter Society, Washington D.C., May 9−11, 2001.

[43] "Vertiflite," *The American Helicopter Society Publication*, Summer 2001, pp. 62−100.

[44] V. Gupta, "Quad Tilt Rotor Aerodynamics in Helicopter Mode," *Ph.D. Dissertation, 2005, Department of Aerospace Engineering, University of Maryland.*

[45] J. Sitaraman and J. D. Baeder, "Analysis of Quad Tilt Rotor Blade Aerodynamic Loads using Coupled CFD/Free Wake Analysis," AIAA Paper 2002−0559, 40th AIAA Applied Aerodynamic Conference, St. Louis MO, June 2002.

[46] W. D. Henshaw, "Ogen : An Overlapping Grid Generator for Overture," Lawrence Livermore National Lab Research Report, UCRL−MA 132237, 1998.

[47] C. Hirsch, *Numerical Computation of Internal and External Flow Vol. II*, Wiley Interscience, 1990.

[48] L. Tang, "Improved Euler Simulation of Helicopter Vortical Flow," *Ph.D. Dissertation, 1998, Department of Aerospace Engineering, University of Maryland*.

[49] H. J. Choi and J. G. Liu, "The Reconstruction of Upwind Fluxes for Conservation Laws: Its Behavior Dynamic and Steady State Calculations," *J. Comput. Phys.*, Vol. 144, 1998, pp. 237−256.

[50] C. W. Shu and S. Osher, "Efficient Implementation of Essentially Non-oscillatory Schemes," *J. Comp. Phys.*, Vol. 77, 1989, pp. 439.

[51] P. K. Sweby, "High Resolution Schemes using Flux Limiters for Hyperbolic Conservation Laws," *SIAM J. Numer. Anal.*, Vol. 21, 1984, pp. 995−1011.

[52] A. Harten, "High Resolution Schemes for Hyperbolic Conservation Laws," *J. Comp. Phys.*, Vol. 49, 1983, pp. 357−393.

[53] A. Harten, B. Engquist, S. Osher and S. R. Chakravarthy, "Uniformly High-Order Accurate Essentially Nonoscillatory Schemes III," *J. Comp. Phys.*, Vol. 71, 1987, pp. 231−303.

[54] A. Harten and S. Osher, "Uniformly High-Order Accurate Nonoscillatory Schemes I," *SIAM J. Numer. Anal.*, Vol. 24, 1987, pp. 279−309.

[55] C. de Boor and B. Swartz, "Piecewise Monotone Interpolation," *J. Approx. Theory*, Vol. 21, 1977, pp. 411−416.

[56] F. N. Fritsch and R. E. Carlson, "Monotone Piecewise Cubic Interpolation," *SIAM J. Numer. Anal.*, Vol. 17, 1980, pp. 238−246.

[57] J. C. Ferguson and K. Miller, "Characterization of Shape in a Class of Third Degree Algebraic Curves," *TRW Report 5322−3−5*, 1969.

[58] A. Suresh and H. T. Huynh, "Accurate Monotonicity-Preserving Schemes With Runge-Kutta Time Stepping," *J. Comput. Phys.*, Vol. 136, 1997, pp. 83−99.

[59] H. T. Huynh, "Accurate Monotone Cubic Interpolation," *SIAM J. Numer. Anal.*, Vol. 30, No. 1, Feb 1993, pp. 57−100.

[60] H. T. Huynh, "Accurate Upwind Methods for the Euler Equations," *SIAM J. Numer. Anal.*, Vol. 32, No. 5, Oct. 1995, pp. 1565−1619.

[61] J. Butland, "A Method of Interpolating Reasonable-Shaped Curves through any Data," Proc. of Computer Graphics 80, Online Publications, Northwood Hills, Middlesex, U.K., 1980, pp. 409−422.

[62] F. N. Fritsch and J. Butland, “A Method for Constructing Local Monotone Piecewise Cubic Interpolants,” *SIAM J. Sci. Statist. Comput.*, Vol. 5, 1984, pp. 300−304.

[63] R. E. Carlson and F. N. Fritsch, “Monotone Piecewise Bicubic Interpolation,” *SIAM J. Numer. Anal.*, Vol. 22, No. 2, Apr 1985, pp. 386−400.

[64] R. E. Carlson and F. N. Fritsch, “An Algorithm for Monotone Piecewise Bicubic Interpolation,” *SIAM J. Numer. Anal.*, Vol. 26, No. 1, Feb 1989, pp. 230−238.

[65] http://www.cray.com/products/x1e/index.html

[66] P. Muzio and R. Walsh, “Total Life Cycle Cost Comparison: Cray X1 and Pentium 4 Cluster,” Cray User Group Conference, Columbus OH, May 12−16, 2003.

[67] Andrew A. Johnson, “Computational Fluid Dynamics Applications on the Cray X1 Architecture: Experiences, Algorithms and Performance Analsis,” Cray User Group Conference, Columbus OH, May 12−16, 2003.

[68] L. Oliker, A. Canning, et al., “Evaluation of Cache-based Superscalar and Cacheless Vector Architectures for Scientific Computations,” SC’03, Phoenix AZ, Nov. 15−21, 2003.

[69] H. Shan and E. Strohmaier, ”Performance Characteristics of the Cray X1 and Their Implications for Application Performance Tuning,” ICS’04, Malo France, June 26−July 1, 2004.

[70] http://www.co-array.org

[71] R. W. Numrich and J. Reid, "Co-array Fortran for Parallel Programming," *ACM SIGPLAN Fortran Forum*, Vol. 17, No. 2, Aug 1998, pp. 1−31.

[72] Y. Dotsenki, C. Coarfa and J. Mellor-Crummey, "A Multi-Platform Co-Array Fortran Compiler," 13th International Conference on Parallel Architecture and Compilation Technique (PACT'04), Antibes Juan-les-Pins, France, Sept. 2004.

[73] R. Meakin, "Composite Overset Structured Grids," *Handbook of Grid Generation.* Joe F. Thompson, Bharat K. Soni, Nigel O. Weatherill (Eds.), *CRC Press*, 1999, Chapter 11.

[74] A. M. Wissink and R. L. Meakin, "Computational Fluid Dynamics with Adaptive Overset Grids on Parallel and Distributed Computer Platforms," International Conference on Parallel ane Distributed Computing, Las Vegas NV, July 1998.

[75] A. M. Wissink and R. L. Meakin, "On Parallel Implementations of Dynamic Overset Grid Methods," Proceedings of SC97: High Performance Computing and Networking, San Jose CA, Nov. 15−21, 1997.

[76] R. Meakin, "Unsteady Aerodynamic Simulation of Static and Moving Bodies using Scalable Computers," AIAA 99−3302, 14th AIAA Computational Fluid Dynamics Conference, Norfolk VA, June 1999.

[77] Y. Lee and J. D. Baeder, "High-Order Overset Method for Blade Vortex Interaction," AIAA Paper 2002−0559, 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV, Jan. 14−17, 2002.

[78] V. Gupta and J. D. Baeder, “Quad Tilt Rotor Aerodynamic Performance in Forward Flight,” AIAA Paper 2002−0559, 40th AIAA Applied Aerodynamic Conference, St. Louis MO, June 2002.

[79] Y. Lee and J. D. Baeder, “Vortex Tracking in Overset Method for Quad Tilt Rotor Blade Vortex Interaction,” AIAA Paper 2003−3531, 21st AIAA Applied Aeroddynamics Conference, Orlando FL, June 23−27, 2003.