UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Aeronautics, Astronautics and Computational Engineering

GPU Based Aeroacoustic Computation with Prefactored Compact Schemes

by

Shuming Miao

Thesis for the degree of Doctor of Philosophy

August_2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT

Aeronautics, Astronautics and Computational Engineering

Thesis for the degree of Doctor of Philosophy

GPU BASED AEROACOUSTIC COMPUTATION WITH PREFACTORED COMPACT SCHEMES

by Shuming Miao

In this work a computational aeroacoustic (CAA) solver, based on finite difference method, used for sound propagation in engineering practice, is accelerated on graphics processing units (GPUs) by using CUDA FORTRAN. The high-fidelity CAA solver is governed by linearized Euler equations (LEE), which features high-order, optimized prefactored compact schemes with low dissipation and dispersion. Solving prefactored compact schemes gives rise to bidiagonal matrices and it is the dominant computational cost in the CAA solver. Multiple methods for solving the bidiagonal matrix are investigated on GPUs. The numerical methods achieve different performance in the *x*, *y* and *z* directions due to anisotropic memory access pattern. The anisotropic memory access pattern refers to coalesced memory access in a direction, which increases the computational performance, and redundant memory access in the other directions, which increases the computational cost. A new hybrid method is proposed in terms of the anisotropic memory access and a strategy is formulated for solving the bidiagonal matrix in 3D computations. In addition, multiple-GPU implementation is added to the solver. Different parallel strategies are applied to different subroutines in accordance with different memory access patterns. The data transfer between multiple GPUs is also optimized by a direct data transfer between GPUs. Based on a comparison of the wall-clock time on the same amount of CPU cores and GPUs, speed-ups of 40-80 are achieved in double precision.

The new solver is used to investigate the scattering of propeller noise off a cylinder and the refraction effect of boundary layer. It is found that the propagation of thickness noise concentrates on the ring plane whereas that of loading noise concentrates on the ring plane and inclines upstream. In addition, the refraction effect of the boundary layer is weak and negligible at M = 0.205 whereas it is significant at M = 0.75. An extension of computation shows that the refraction effect of the boundary layer becomes important when $M \geq 0.3$.

Finally, the new solver is employed to predict the scattering of propeller noise off an ATR-72 like wing-body at a full scale. Results show that the current solver can be used to predict the large-scale engineering cases and an acceptable wall-clock time and speed-up is achieved.

Table of Contents

List of Tables

List of Figures

DECLARATION OF AUTHORSHIP

I, SHUMING MIAO, declare that this thesis and the work presented in it are my own and have been produced by me as the result of my own original research:

'GPU BASED AEROACOUSTIC COMPUTATION WITH PREFACTORED COMPACT SCHEMES'.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

    - S. Miao, X. Zhang, O. G. Parchment, and X. Chen. 'A Fast GPU Based Bidiagonal Solver for Computational Aeroacoustics,' *20th AIAA/CEAS Aeroacoustics Conference*, Atlanta, Georgia, 2014.
    - S. Miao, X. Zhang, O. G. Parchment, and X. Chen. 'A fast GPU based bidiagonal solver for computational aeroacoustics,' *Computer Methods in Applied Mechanics and Engineering,* vol. **286**.pp. 22-39, 2015.
    - S. Miao, Y. Hou, and X. Zhang. 'Evaluation AND COMPARISON OF LINEARIZED EULER EQUATIONS AND EQUIVALENT SOURCE METHOD,' *The 22nd International Congress on Sound and Vibration,* Florence, Italy, 2015.

Signed: .....

Date:.....

Acknowledgements

The author would like to thank my supervisors Prof. Xin Zhang, and Dr. Zhiwei Hu for providing support, guidance and encouragement throughout the course of my PhD. Thanks must also go to Dr. Oswald Parchment for invaluable support with high performance computing.

The author would also like to thank Sylvain Grosse who continually helped me to revise the wing body geometry, and to Dr. David Angland who offered much data and many suggestions on the setup of RANS simulation of the ATR 72-like wing body.

Thanks are also given to my colleagues, Xiaoxian, Yu, James, Ryu, Fernando, Fuyang and Meng who gave much support and many suggestions.

Nomenclature

Roman

B	Blade number
a_F, b_F, c_F	Coefficients of RHS forward derivative
a_B, b_B, c_B	Coefficients of RHS backward derivative
$\mathbf{C}$	Coefficient Matrix
c	Speed of sound
$c_{i,j}$	Coefficients of explicit filter
$D_{i,j,k}$	Primitive derivative
$D^F_{i,j,k}$, $D^B_{i,j,k}$	Forward, backward derivatives
e, s	Biased coefficients of optimized prefactored compact scheme
$\mathbf{F}$	Force
$\mathbf{f}$	primitive variable matrix
f	Control surface
G	Green's function
H	Heaviside step function
h	Elimination step
J	Bessel function of the first kind
$\mathbf{K}$	Incremental vector at an intermediate stage
k	Wavenumber
L	Length
l	Local force
$\mathbf{M}$	Mach number
M	Total elimination step

m	Spinning order of duct mode
$\mathbf{n}$	Normal vector
Nx, Ny, Nz	Domain dimension in the x, y and z directions
p	Pressure
pp	Fraction of code which can be parallelized
P_T	Thickness Noise
P_L	Loading Noise
q	Monopole strength
r	Amplification factor, radius
r_n	Numerical amplification factor
r_e	Exact amplification factor
s_1, s_2, s_3, s_4, s_5	Source terms in LEE
sp	Fraction of code in serial
t	Time
T_{ij}	Lighthill's stress tensor
$\mathbf{U}$	Solution vector
$\mathbf{U}_{\text{target}}$	Forced field in buffer region
u, v, w	Velocity components
$\mathbf{V_0}$	Velocity of mean flow
x, y, z	Cartesian coordinates

Greek

α, β	Coefficients of optimized prefactored compact scheme
γ	Ratio of specific heat
δ	Dirac Delta function

ε	Amplitude of perturbation
ϑ	Angle in the azimuthal direction
ξ, η, ζ	Coordinates in the computational domain
ρ	Density
σ	Damping coefficient
τ	Source time
τ_{ij}	Viscous stress tensor
φ	Angle on the ring plane
ω	Angular frequency

Subscripts

$()_0$	Mean flow field
$()_{ret}$	Values at retarded time

Superscripts

-	Filtered field
$()^B$	Forward derivative
$()^F$	Forward derivative
$()'$	Perturbation

Abbreviations

2D	Two Dimensional
2.5D	Two-and-a-half Dimensional
3D	Three Dimensional
AA	Acoustic Analogy
ANTC	Airbus Noise Technology Centre
APU	Accelerated Processing Unit

ATR	Regional Air Transport
BEM	Boundary Element Method
BL	Boundary Layer
BPF	Blade Passing Frequency
CAA	Computational AeroAcoustics
CESM	Complex Equivalent Source Method
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy condition
CGNS	CFD General Notation System
CPU	Central Processing Unit
CR	Cyclic Reduction
cuBLAS	CUDA Basic Linear Algebra Subroutines
CUDA	Compute Unified Device Architecture
DG	Discontinuous Galerkin
DNS	Direct Numerical Simulation
DLR	German Aerospace Centre
DRP	Dispersion-Relation-Preserving
FDM	Finite Difference Method
FEM	Finite Element Method
FLOPS	FLoating-point Operations Per Second
FVM	Finite Volume Method
FW-H	Ffowcs Williams and Hawkings
GPGPU	General Purpose computing on Graphics Processing Unit
GPU	Graphics Processing Unit

HPC	High Performance Computing
ISVR	Institute of Sound and Vibration Research
LaRC	NASA Langley Research Centre
LDDRK	Low-Dissipation and Low-Dispersion Runge-Kutta
LEE	Linearized Euler Equations
LHS	Left Hand Side
MPI	Message Passing Interface
NRBC	Non-Reflecting Boundary Condition
OpenACC	Open Computing Language
OpenCL	Open Accelerators
OpenMP	Open Multi-Processing
PCI	Peripheral Component Interconnect
PCR	Parallel Cyclic Reduction
PGI	The Portland Group, Inc
PPW	Points Per Wavelength
PSD	Power Spectral Density
RAM	Random-Access Memory
RANS	Reynolds-Averaged Navier-Stokes
RD	Recursive Doubling
RDMA	Remote Direct Memory Access
RHS	Right Hand Side
RK	Runge-Kutta
RPM	Revolutions Per Minute
SD	Spectral Difference

SE	Spectral Element
SM	Streaming Multiprocessor
SPL	Sound Pressure Level
SV	Spectral Volume
URANS	Unsteady Reynolds-Averaged Navier-Stokes
UVA	Unified Virtual Addressing

Chapter 1: Introduction

1.1 Overview

With the development of civil aircraft design requirements, noise generated by aircrafts has become an increasingly important issue for a number of reasons. First, the continuously growing air traffic and a rise in public awareness of its impact on the environment have forced the government to make new regulations on airport noise emission [1]. Second, noise emission is treated as an important performance aspect of the airplane. For example, the whole European aviation industry aims to reduce noise by 50% by 2020 and by 65% by 2050, compared to typical new aircrafts manufactured in 2000 [2, 3]. Lastly, many airports regard noise as a major problem [1, 4]. Effective measures of noise reduction have to be adopted if a long-term increase in air traffic is to be sustained and this will require further investigation into the physics of aerodynamically generated noise.

With the increasing demand for understanding the physics of aerodynamically generated sound as well as the development of computational power of computers and numerical techniques, computational aeroacoustics (CAA) is increasingly employed to study airframe/engine noise. Computational aeroacoustics focuses on the generation, propagation and radiation of aerodynamic sound and requires time-accurate simulations [5]. Compared to that of an aerodynamic flow field, the amplitude of aeroacoustic field is small (normally five or six orders smaller) [6]. However, it travels a long distance [7]. Consequently, the numerical methods in CAA must be of high-order accuracy, and have low dissipative and dispersive properties. The resolution of a discretized mesh has to be kept high up to observer points to accurately capture the sound wave. These factors result in high cost and long runtime of CAA simulations, which can deter CAA methods from being applied to real engineering problems.

A generic CAA engineering problem is the prediction of the propeller/turboprop noise scattering off an aircraft. The computational domain is large and the grid resolution has to be high enough to capture the necessary harmonics of the blade passing frequencies (BPF) with high sound energy. In addition, complex geometry and requirements for the stencil size of high order schemes at boundaries also increase the total mesh amount. Some studies have shown that direct computation of turboprop noise generation and propagation with real-sized wings using an unsteady Reynolds-averaged Navier-Stokes (URANS) is prohibitively expensive [8], let alone for a wing body. Therefore, a hybrid method is preferred in some studies [9, 10]. A CAA hybrid method splits a computational domain into different coupled regions and applies different governing equations to each region.

Appropriate assumptions and simplifications are made in the propagation and radiation regions so that the computational cost is reduced significantly. By using optimized CAA high-order schemes [11-14], the total size of mesh points can be reduced further. The CAA high-order schemes provide high-order accuracy, low dissipation and dispersion properties and resolve acoustic waves on a much coarser mesh. Consequently, the computational cost is reduced by using CAA high-order schemes. Given a harmonic wave $e^{i\alpha x}$, the resolved harmonic wave can be described by $e^{i\bar{\alpha}x}$, in which α, called true wave number, is a real number whereas $\bar{\alpha}$, called resolved wave number, is a complex [11-14]. The resolved wave can be expressed in real and imaginary parts:

$$e^{i\bar{\alpha}} = e^{i(\alpha_r + i\alpha_i)} = e^{-\alpha_i} e^{i\alpha_r} = re^{i\alpha_r} \tag{1.1}$$

In which, *r* defines the dissipation error while $(\alpha_r - \alpha)$ defines the dispersion error. The CAA high-order schemes endeavour to make *r* as 1, which means low dissipation, and $(\alpha_r - \alpha)$ as 0, which means low dispersion, at an extent of α. Though CAA codes use parallel processing, e g., message-passing interface (MPI) [15] on a high performance computing (HPC) cluster, the unsteady simulations, millions or even tens of millions of mesh points, and high-order schemes make the computational cost expensive and high demand on computing resources, which prevents CAA from being extensively applied to real engineering problems.

To address the complexity of aerodynamic noise phenomena and to meet the stringent accuracy requirements of numerical schemes employed in sound computation, significant improvements in the available computing resources are necessary. HPC plays an indispensable role in modern numerical simulations. A graphics processing unit (GPU), common in desktop computers, has higher floating-point operation performance and memory bandwidth than a multi-core central processing unit (CPU) [16]. These features have motivated numerous studies in the area of general-purpose computing on GPUs (GPGPU). Recent developments in the use of GPGPU applications in computational fluid dynamics (CFD) [17-22] and room acoustics [23-25] have encouraged the author to extend these efforts to the CAA field.

The aims of this research were to first accelerate a high-order, transient acoustic scattering program, which features high-order, low-dissipation and dispersion schemes based on finite difference method, with multiple GPU implementations. Second, employed this new efficient solver to investigate the propeller noise scattering off a cylinder and the refraction effect of the boundary layer computationally. Finally, predicted the propeller noise scattering off a wing-body at a full scale and evaluated the speed-up on the large-scale engineering case with complex geometry.

1.2 Literature Review

CAA simulations are transient and computationally expensive. Multiple aspects of efforts which cover the governing equations on physics, the numerical schemes and HPC on implementation are performed to reduce the computational cost and wall-clock time. On physical models, a hybrid method is used to reduce the computational cost in the sound propagation and radiation regions. On numerical methods, the CAA high-order low-dissipative and low-dispersive schemes are employed to reduce the mesh resolution. Finally, on HPC aspects, HPC computers and GPUs are used to accelerate CAA applications. This section reviews relevant methods used in this research.

1.2.1 Computational Aeroacoustics

CAA investigates sound generation, propagation and radiation in airflows by using numerical methods. The CAA hybrid method splits a computational domain into different coupled regions and applies different governing equations to each region as shown in Figure 1.1. This research focuses on the propagation model.

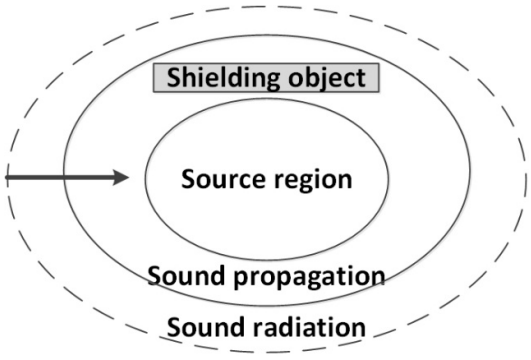


Figure 1.1 Schematic of three sound regions in hybrid method.

1.2.1.1 Source Region

In the sound source region, a strong coupling exists between aerodynamic field and acoustic field. Both fields interacts with each other and cannot be separated [5]. Non-linear effects and viscous effects normally play an important role in sound generation and propagation in this region. The source region is normally governed by the Navier-Stokes equations to include the non-linear effect and viscous effect. In the low Mach number flow, the wavelength of turbulence close to a solid object is much smaller in comparison to the acoustic length [26], thus the mesh resolution is very high, which makes the simulation computationally expensive. The source region must be kept as

small as possible for the sake of efficiency yet large enough to include all the important sound sources.

1.2.1.2 Propagation Region

The propagation of sound forms a link between the generation and the radiation of noise if there are shielding objects outside the source region as shown in Figure 1.1, because the acoustic analogy [27] based on free-space Green's function cannot be employed directly. The control surface used in the computation of sound radiation has to be placed sufficiently far away so that it includes the shielding objects. The shielding objects cause sound reflection and refraction effects. Consequently, acoustic far-field directivities and the near-field sound pressure levels (SPL) can change due to the shielding objects. In the propagation region, a weak coupling exists between the aerodynamic field and acoustic field [5]. Sound is regarded as a small disturbance propagating in a non-uniform flow field with solid geometries. The feedback of the sound field onto the flow field is ignored [26]. In the propagation model, the noise source can be extracted from simulations in the source region, obtained from experiments and modelled by analytical methods. The noise sources are introduced by the inflow boundary condition or source terms at right hand side (RHS) of the governing equations. Therefore, aeroacoustic installation effects of aircraft components can be quantified in comparison to the noise in free space. Some innovative configuration designs, such as Rear Fuselage Nacelle, Scarfed Aft Fan and 'flying wing' [28], which use airframe components as shielding objects to reduce the sound energy received by observers on the ground, can be studied efficiently. Another considerable advantage of the propagation model is that linear disturbance equations are used as the governing equations, ignoring sound non-linear effects, viscosity and feedback on the background mean flow field. This reduces the computational cost in comparison to a sound generation solver. Based on the simplifications above in the propagation region, the general governing equations of sound propagation in the non-uniform flow field can be described by linearized Euler equations (LEE). LEE is used extensively in some CAA communities, such as DLR [10, 29, 30], NASA Langley Research Centre (LaRC) [31], Institute of Sound and Vibration (ISVR) [32, 33] and Airbus Noise Technology Centre (ANTC) [34-37]. LEE can describe three types of perturbation waves: acoustic, vortical and entropy waves. It is hard to distinguish between them by using numerical methods. When the perturbations only contain sound waves or when other waves can be ignored, LEE can be used to simulate the sound propagation. When numerical methods are employed to solve LEE it is a key point for numerical schemes to keep the same wave numbers as partial differential equations [11]. This involves high-order, low-dissipative and low-dispersive CAA schemes used in current research which will be discussed in section 1.2.2.

1.2.1.3 Radiation Region

The radiation region is free of solid objects, thus acoustic analogies (AA) based on free-space Green's function are used to predict radiation efficiently when the volume quadrupoles are ignored [27, 38]. In the radiation region, acoustic disturbance is typically travelling in a uniform mean flow. No coupling exists between aerodynamics and acoustics [5]. A control surface, which encompasses all the important sound information, is used to collect the sound information. Subsequently, the control surface is regarded as a sound source surface which radiates sound outwards.

1.2.2 CAA Numerical Schemes

Because a sound wave travels a long distance [7], spatial derivative and time integral schemes have to be high-order, low-dissipative and low-dispersive [11-14, 39, 40]. Those acoustic waves which are not resolved with fidelity, called spurious short waves, have to be damped by high-order filters [41-43]. These high-order, low-dissipative and low-dispersive schemes resolve higher wave numbers in comparison to low-order schemes in some CFD codes which mainly focus on the flow field close to a solid object and the mean quantity such as lift coefficient. Consequently, the total mesh amount can be reduced because fewer mesh points are required to resolve an acoustic wave with a given frequency. Combined with a high-order low-dissipative and low-dispersive time-marching method, the time-marching step size in time-accurate simulations is also increased in terms of the stability and accuracy requirement [11, 40]. Therefore, the computational cost is reduced significantly since the total steps of iteration are reduced.

1.2.2.1 High-Order Methods

In engineering field, solvers using the second-order schemes based on finite volume methods on unstructured meshes are successful, robust and have become industry-standards in a broad range of engineering problems for a long time [44]. High-order methods are not necessary for those problems that low-order methods can provide required accuracy and efficiency. However, the emerging revolutionary design of aircraft, which tries to predict the phenomena at the limits of flight envelope, aeroacoustics and the unsteady, vortex-dominated flow, requires high-fidelity simulations. These CFD/CAA simulations cannot be handled efficiently by low-order methods [45]; otherwise, dissipation and dispersion properties of the low-order schemes will require huge amount of mesh size, which imposes unacceptable computational cost [46]. At these situations, using high-order method is necessary. High-order method is a research field which focuses on the unsteady, accurate, vortex-dominated phenomenon and aeroacoustics [44-46]. There are some typical high-order, low-dissipative and low-dispersive spatial schemes based on different discretization methods. It can be realized by finite difference method, finite volume method,

Discontinuous Galerkin (DG) method and spectral difference (SD) method. In the following paragraphs, the advantages and drawbacks of different discretization method, tabulated in Table 1-1 below, are reviewed and summarized.

Table 1-1 Features of high order methods implemented by FD, FV, DG and SD methods.

Features of high order methods	FD	FV	DG	SD
Easy to construct high order accuracy	++	-	+	+
Low dissipation and dispersion	++	+	++	++
Compactness of stencil	+	-	++	++
Parallelization	++	-	++	++
Unstructured mesh	-	+	+	+
Mesh quality requirement	-	+	++	++
Computational cost	++	-	-	-
Memory requirement	++	+	-	-
Robust	-	+	+	+
Programming complexity	+	-	-	-

Note: "++","+","-" denote the best, good and bad respectively.

Finite difference method discretizes conservation laws in the differential form on a structured mesh [47]. By using a structured mesh, the computation of derivatives is decoupled between *x*, *y* and *z* directions. High order accuracy is easily achieved by adding the stencil size of schemes in the pointwise direction. As shown in Table 1-1, the main strength of finite difference method is that high order accuracy is easy to construct and program, and algorithms are efficient in terms of computational cost [48]. In addition, the numerical property of schemes has been well analysed and a variety of schemes, such as DRP scheme [11] and optimized prefactored compact scheme [13, 14], optimized in terms of dissipation and dispersion, are available. Consequently, it is normally used in computationally costly problems, such as direct numerical simulations (DNS) and aeroacoustics [6, 46]. Its drawback is restricted to structured mesh. Smooth, high quality structured mesh has to be used for stability reasons, which results in high effort on mesh generation for complex geometries [47]. The high-order solvers based on finite difference method are the most widely used in CAA, such as PIANO in DLR [10, 29, 30], sAbrinA in ONERA [49, 50], which employ DRP schemes and SotonLEE [34-37] that utilizes prefactored compact schemes.

Finite volume method discretizes governing equations in integral formulations [47]. The fluxes through mesh boundaries are evaluated. High order accuracy is obtained with the aid of reconstruction procedure [47, 48] which uses the intermediate solution in adjacent cells to achieve high order accuracy. The mesh cells in the reconstruction is called reconstruction stencil. The most

attractive advantage of finite volume method is its well fit to unstructured meshes as shown in Table 1-1. Therefore, the mesh generation of complex geometries is much easier. However, the drawbacks also exist. First, the reconstruction stencil size is large. A lot of information has to be exchanged between different CPUs therefore it is not efficient for parallelization. Second, the high-order scheme is expensive in terms of wall-clock time and is quite complex to program, especially in 3D computations. Third, real high-order schemes are still hard to implement on irregular meshes. Most of high-order schemes are based on smooth, regular meshes [47].

In finite element method, the differential form of governing equations is multiplied by a test function, which is usually a polynomial, and is integrated by parts. It splits the physical domain into a collection of elements (which can be regarded as coarse unstructured meshes). The solution is a linear combination of basis functions, often piecewise polynomials [47]. In finite element method, DG method is an attractive frequently used approach in aeroacoustics [48]. In DG, no global continuity is required between mesh elements and solution is discontinuous between mesh elements. Conservation and high order accuracy is achieved locally in a mesh element. Double-valued solutions exist on element interfaces. A Riemann solver [47] is used to handle the numerical flux and couple the solutions between elements and provide dissipation for stability. In DG, test functions are used as basis functions. First, high order accuracy is easy to construct and depends on the degree of chosen polynomial test functions. Second, the discretization stencil is compact and no reconstruction procedure is required. Therefore, it is well suited for parallelization and hp-adaptations, in which h denotes to increase mesh density whereas p denotes to increase polynomial degree to enhance solution accuracy. Third, it fits well to unstructured meshes. However, the computational cost and programming complexity is high in comparison to finite difference method. In addition, the memory requirement is quite high if an implicit time stepping is used.

Spectral method is another kind of high-order method which features the spectral (exponential) convergence. In traditional spectral methods, unknown variables are expressed as a truncated series expansion in terms of the basis function [46]. The basis functions, which are infinitely differentiable global functions, frequently employ trigonometric functions or Chebyshev and Legendre polynomials [51]. Two types of formulations are available: modal formulations and nodal formulations. The modal formulations are more computationally expensive, therefore the nodal formulations are preferred, in which unknowns are nodal values of unknown variables at so-called collocation points. Fluxes are computed at nodal points using nodal values of unknowns. The major shortcoming of traditional spectral methods is the restriction to simple domains [51, 52].

Recent researches [51, 52] have extended the spectral methods to complex geometries, including spectral element (SE) method, spectral volume (SV) method and SD method, in which the SD

method is the most efficient and requires less memory in comparison to the SE and SV methods. SD is a type of nodal spectral method for unstructured meshes, in which in each mesh structured nodal unknowns and fluxes distribute. In SD, the high-order local representations are employed to achieve conservation and high-order accuracy in a manner similar to DG [46, 51]. The conservative unknowns are defined at quadrature points so that the volume integral is simplified and approximated to the desired order and the computational cost is reduced in comparison to the surface integral in SV. Flux derivatives are obtained by a polynomial reconstruction of fluxes at certain flux points located at surface quadrature points. The advantages and shortcomings of SD are similar to those of DG. In addition, SD is more efficient than DG since the differential form of conservation laws is used which results in a reduction in the computational cost in surface integrals.

It can be concluded that finite difference method, DG and SD fit better to high-order methods in comparison to finite volume method. The most attractive advantage of finite difference method is that it is easy to construct and the properties of the numerical schemes are well analysed [11-14, 39]. The computational cost is the lowest and programming complexity is relatively low. However, the biggest drawback is its restriction to high-quality, smooth, structured meshes and is not robust. In comparison, DG and SD fit well to and are relatively robust on unstructured meshes. However, the computational cost is expensive and memory requirement is high. In addition, the programming complexity is high. Finally, the choice of high-order methods also depends on the CAA code available. The current available CAA code in ANTC is SotonLEE [34-37] which is based on finite difference method. SotonLEE contains features below:

- Written in FORTRAN90
- Based on Linear Euler Equations
- Multi-block curvilinear structured mesh
- Ashcroft and Zhang's fourth-order optimized prefactored compact scheme [14] and Hixon's sixth-order prefactored compact scheme [13]
- Tenth-order and sixth-order explicit filters [41, 42]
- Hu's alternating 4-6 stages low-dissipation and low-dispersion Runge-Kutta time-marching scheme [40]
- Outflow buffer zone condition [53]
- In double precision format
- In non-dimensional format

Consequently, SotonLEE, based on finite difference method, is selected as the CAA solver in this study. The governing equations, time-marching schemes and boundary conditions in SotonLEE in ANTC are almost the same as two other widely used CAA propagation solvers, PIANO [10, 29, 30] in

DLR and sAbrinA [49, 50] in ONERA. In PIANO and sAbrinA, DRP schemes are used. SotonLEE distinguishes from these solvers by using optimized prefactored compact schemes [14], which resolve acoustic waves better if the stencil size is the same.

1.2.2.2 CAA Numerical Schemes in Finite Difference Method

In terms of low-dissipation and low-dispersion schemes discretized by finite difference method, there are two prototypes: explicit schemes, such as DRP schemes, and compact schemes, which are globally dependent on the whole domain. The dissipation and dispersion of compact schemes is lower in comparison to that of DRP schemes if the same stencil size is given [13, 14]. However, the implementation is more complex and the requirement on the boundary conditions is more stringent since compact schemes is globally dependent. Optimized prefactored compact scheme is a kind of CAA high-order schemes proposed by Hixon [13] and Ashcroft and Zhang [14]. It is an optimization of compact schemes which is globally dependent on the elements in the whole domain. However, the coefficients in stencil computation are obtained and fixed by the optimization of wave propagation. In sound generation and propagation areas, prefactored compact schemes offer lower-dissipation, lower-dispersion properties and higher-order accuracy in comparison to the explicit schemes used in the engineering field. Furthermore, it reduces boundary stencil size in comparison to compact schemes since it splits a coefficient system into a couple of smaller ones, which facilitates numerical implementation near boundaries [14]. A prototype, fourth-order prefactored compact scheme used in this research can be expressed in the following form:

$$D_{i,j,k} = \frac{1}{2}\left(D^{F}_{i,j,k} + D^{B}_{i,j,k}\right) \tag{1.2}$$

$$\alpha D^{F}_{i+1,j,k} + \beta D^{F}_{i,j,k} = \frac{1}{\Delta x}\left[a_F f_{i+1,j,k} + b_F f_{i,j,k} + c_F f_{i-1,j,k}\right] \tag{1.3}$$

$$\beta D^{B}_{i,j,k} + \alpha D^{B}_{i-1,j,k} = \frac{1}{\Delta x}\left[a_B f_{i+1,j,k} + b_B f_{i,j,k} + c_B f_{i-1,j,k}\right] \tag{1.4}$$

in which, $D_{i,j,k}$,$D^{F}_{i,j,k}$ and $D^{B}_{i,j,k}$ denote the unknown primitive derivative, the forward derivative and the backward derivative on a mesh point respectively, whereas $f_{i,j,k}$ represents the known primitive field. The coefficients, α, β, a_F, b_F, c_F, a_B, b_B and c_B, are constants. For a given *j* and *k*, *i* varies in descending order from *Nx* - 1 to 2 in Eq. (1.3) and *i* varies in ascending order from 2 to *Nx* - 1 in Eq. (1.4). The system is coupled and globally dependent. The resolved wavenumber of fourth-order optimized prefactored compact scheme is compared with those of typical finite difference schemes as shown in Figure 1.2:

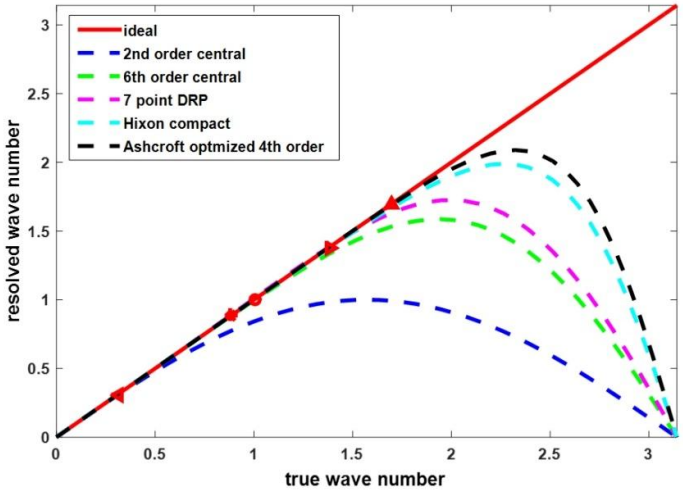


Figure 1.2 Resolved wavenumbers of typical finite difference schemes.

Note: The resolved wave numbers of all finite difference schemes are: second-order central scheme with 0.314; sixth-order central scheme with 1.005; 7 point DRP scheme with 0.880; Hixon's sixth-order compact scheme with 1.382; Ashcroft and Zhang's fourth-order scheme with 1.700.

Figure 1.2 illustrates that CAA finite difference schemes, such as the DRP scheme [11] and Ashcroft and Zhang's optimized prefactored compact schemes [14] can greatly enhance the extent of resolved wave number, thus greatly reducing points per wavelength (PPW) and mesh resolution. Central schemes do not give rise to any dissipation error but only dispersion error since the resolved wave numbers of central schemes are real as shown in Eq. (1.1). When 0.001 [12, 40] is selected as the largest tolerance between the resolved wave number and the true wave number, the red points in Figure 1.2 indicate exact values of the maximum resolved wave numbers for all finite difference schemes. Ashcroft and Zhang's fourth-order optimized prefactored compact scheme offers the best spectral performance in Figure 1.2. The maximum resolved wave number of the fourth-order optimized prefactored compact scheme achieves 1.700, whereas it is 0.314 for the second-order central scheme. This difference makes the PPW 3.7 for the fourth-order optimized prefactored compact scheme and 20.0 for the second-order central scheme. Consequently, the mesh element amount can be decreased roughly by 5 times in one dimension and 125 times in a 3D computation. In addition, the larger mesh element size results in larger time-marching step size in unsteady simulations if the same Courant-Friedrichs-Lewy condition (CFL) is used and the same accuracy is required. In conclusion, the CAA high-order low dissipative and low dispersive finite difference schemes can decrease the total amount of mesh element and computational cost significantly. However, the compact schemes and optimized prefactored compact schemes result in tridiagonal

and bidiagonal matrices, as shown in Eqs. (1.3-1.4), which are spatially global-dependent and make themselves hard to solve in parallel.

As can be seen from Figure 1.2, short waves with large wave numbers cannot be resolved with fidelity. The waves which are not resolved with fidelity by high-order finite difference schemes are called spurious waves. Spurious waves have to be removed from solution, otherwise the solution is contaminated and can even crash [11]. High-order spatial filters are usually used to remove the spurious short waves from the solution [12, 41-43]. The filtering properties of two generic high-order explicit filters are shown in Figure 1.3 below:

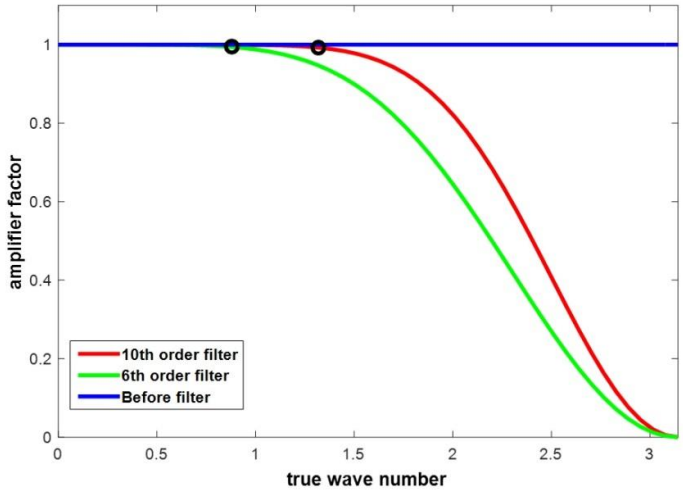


Figure 1.3 Filtering performance of explicit filters.

In Figure 1.3, the tenth-order filter attenuates a signature more sharply than the sixth-order filter. When 0.001 again is chosen as the maximum wave attenuation tolerance, the resolved wave numbers, which are denoted by black circles, are 1.320 for the tenth-order filter and 0.880 for the sixth-order filter. Because the maximum resolved wave numbers are 1.700 for the fourth-order optimized prefactored compact scheme and 1.38 for Hixon's sixth-order prefactored compact scheme, the tenth-order explicit filter with effective wave number 1.320 is used in collaboration in this research.

In addition to spatial-scheme optimization, time schemes in CAA are also optimized to achieve low-dissipation, low-dispersion and high-order accuracy [40]. The multi-stage Runge-Kutta (RK) scheme is a kind of high-order time scheme and is widely used in CFD codes. However, RK schemes in CFD are obtained to keep the maximum formal order rather than to achieve low dissipation and dispersion, which is similar to spatial finite-difference schemes. To accommodate RK schemes with

CAA applications, RK schemes have to be optimized in terms of dissipation error and dispersion error. The dissipation error and dispersion error are defined in terms of an amplification factor of a wave. The amplification factor is defined as the ratio of a wave $e^{-ick(t+\Delta t)}$ at the next time step between e^{-ickt} at current time step [40]. The ratio between numerical amplification factor, $e^{-ick^*(t+\Delta t)}/e^{-ickt}$ in which k^* is the numerical wavenumber, and exact amplification factor, $e^{-ick(t+\Delta t)}/e^{-ickt}$ in which *k* is the true wavenumber, is defined below:

$$r = r_n / r_e = |r| \times e^{-i\sigma} \tag{1.5}$$

in which, *r* is a complex number. r_n and r_e denote the numerical and exact amplification factor respectively; $|r|$ is the dissipation error, whereas σ is the dispersion error. Both of these errors for typical schemes are plotted in Figure 1.4 Figure 1.5 below:

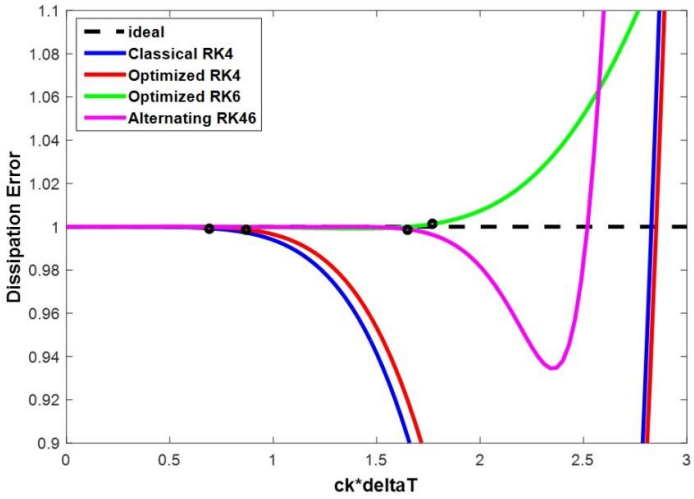


Figure 1.4 Dissipation error of RK schemes.

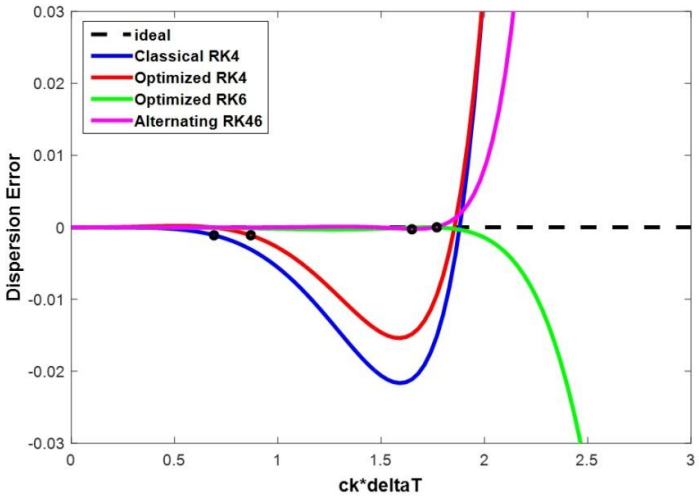


Figure 1.5 Dispersion error of RK schemes.

Again, as in the spatial finite-difference schemes and explicit filters, 0.001 [40] is chosen as the maximum deviation tolerance of dissipation and dispersion errors. As illustrated in Figure 1.4Figure 1.5, optimized six-stage Runge-Kutta schemes and alternating four-six-stage Runge-Kutta (LDDRK) schemes offer the largest accuracy limit $ck^*\Delta t$ which is denoted by black points. c is speed of sound while k^* is the resolved wavenumber by a finite difference scheme. Δt is the time step size. In real practice, the LDDRK scheme is much more stable than the optimized RK6 scheme and is preferable [40]. If the LDDRK scheme is used on a uniform mesh in collaboration with Hixon's sixth-order prefactored compact scheme with the maximum resolved effective wave number 1.382, then the time step size Δt, which is derived from the accuracy limit $ck^*\Delta t < 1.64$, can achieve 1.194 which is much larger in comparison to the classical RK4 scheme. It improves time-marching efficiency significantly.

In conclusion, the above CAA schemes based on finite difference method, which are designed for acoustic computation and distinguish CAA solvers from traditional low-order CFD solvers, greatly reduce the computational cost and enhance the computational efficiency of simulations. However, these high-order schemes suffer from the stability problem which requires the high-quality, smooth meshes and imposes much more complex implementation on boundary conditions and programming. Therefore, though the high order methods can reduce the computational cost, a careful implementation is necessary to apply to problems with very complex geometries in engineering field.

1.2.3 High Performance Computing

A hybrid method, which combines sound generation, propagation and radiation predictions, can improve computational efficiency and reduce computational cost significantly. On numerical schemes, by using CAA high-order spectral-like schemes the computational cost can be reduced even further. However, CAA simulations still demand huge computing resources. To reduce design time in real engineering applications, HPC is an indispensable tool in current numerical simulations. The wall-clock time of CAA simulations can be significantly reduced by using HPC on high performance computers.

To fully explore the performance of HPC computers, a CFD/CAA researcher has to focus on efficient implementation strategies on HPC computers. In addition, efficient parallel algorithms on HPC computers have to be developed if existing algorithms cannot achieve high efficiency in new HPC computers. In CFD/CAA fields, a physical domain is discretized into a collection of mesh points which are stored in data arrays in codes. Intensive computations are performed on mesh elements and the computation on each mesh element is almost the same except for the boundary condition operations, the cost of which can be ignored compared to other operations such as spatial derivatives. Therefore, the computational workload can be scaled by the amount of mesh elements. The total mesh elements are partitioned into some groups as evenly as possible and assigned to each working task to achieve load balancing. This kind of computing fits well with today's HPC multi-core computers. Traditional HPC in CAA/CFD fields can be categorized into three parallel programming models: shared memory model [54], distributed memory model [15, 55] and hybrid model [56-58]. The parallel programming model is an abstraction above hardware and memory architectures. Since there are many kinds of parallel programming models based on different points of view, this review only focuses on relevant programming models based on available computing resources.

1.2.3.1 Distributed Memory Model

The most mature and extensively used programming model is the distributed memory model based on HPC clusters in CAA and CFD, based on the workload partition strategy mentioned above and on the available hardware. Computing clusters are the most extensively used HPC architecture [59]. Computing clusters occupy roughly 86% of the architecture system share according to the latest statistics [59]. A computing cluster is a collection of computers connected by fast network connections [60]. Each computer is called a computing node and multiple CPU cores are equipped in each node. The implementation of the distributed memory model is MPI [15, 55], which is the industry standard for distributed memory implementation. Three key aspects are explained in the distributed memory model (MPI): process, memory model and data exchange. A process is a

fundamental entity implemented on a computer and each CPU core is assigned to an MPI process normally. A process can be regarded as a computational task on a CPU core. In the distributed memory model, many tasks (MPI processes) are set up to run a CAA/CFD application across a number of computing nodes in parallel. These nodes together are regarded as a single super computer with many CPU cores and large memory volume. Each task is assigned a sub-domain (mesh blocks in structured grid) where the computation is carried out. Each task has its own local memory and is invisible to other processes. A sketch of the distributed memory model is shown below in Figure 1.6 Sketch of distributed memory model.:

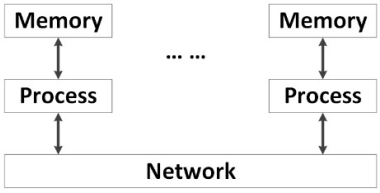


Figure 1.6 Sketch of distributed memory model.

As shown in Fig. 1.6, the distributed memory model regards that the memory between processes is connected via the network; therefore, data exchange is realized by sending and receiving messages between tasks. For CFD/CAA computations, data exchange is necessary close to mesh block interfaces for sub-domains in different processes. The data exchange is realized by calling MPI communication routine libraries. In a CAA/CFD simulation, when the flow field at a neighbouring sub-domain is required, the process first sends its own flow field close to the interface and then waits for the flow field from the neighbouring blocks. Once the send and receive operations complete, the computation starts. An MPI process keeps computing until the next time the flow field in a neighbouring sub-domain is required. In addition to sending/receiving messages, data exchange can also be achieved by visiting globally common memory, which is essentially the shared memory model.

1.2.3.2 Shared Memory Model

In shared memory programming, thread model is the most popular and the most supported [54]. The most widely used implementation of the thread model is Open Multi-Processing (OpenMP) [54]. It is compiler directive-based therefore easy to use. It is the compiler's task to automatically generate parallel codes according to the directives and compiling flags [54]. A sketch of the shared memory model is shown below in Figure 1.7:

Figure 1.7 Sketch of shared memory model.

In the thread model (shared memory), there are also three key aspects: thread, memory model and data exchange. A thread can be regarded as an instruction stream from a CPU's perspective though the concept is more complicated in an operating system. A single 'high-cost' process has multiple concurrent 'low-cost' threads. A process acquires all necessary system resources and it is 'high-cost'. A process performs serial work and creates multiple concurrent threads to do the parallel work. Each thread shares the resources and memory address of a process and has local data. The data exchange is performed through the global memory in a process. Consequently, a data race may occur if the code is not well designed. In the thread model, two parallel strategies can be employed on CAA/CFD applications. The first one is similar to that used in the distributed memory model. Each thread is assigned a physical sub-domain and computations are performed on the sub-domain. Based on the available cluster specifications [60], there are 16 CPU cores on each node. Therefore, threads over 16 have no additional benefit in this parallel strategy. The second parallel strategy involves using the concurrent multiple threads to solve a CAA/CFD application in each sub-domain in parallel. As mentioned in the distributed memory model, many sub-domains are solved by MPI processes simultaneously. However, the computation in each sub-domain is done serially. Using multiple concurrent threads to solve each sub-domain in parallel can extend the maximum degree of parallelism. This kind of model is called a hybrid model.

1.2.3.3 Hybrid Model

A hybrid model is a combination of the distributed memory and shared memory models [56-58] sketched in Figure 1.8 below:

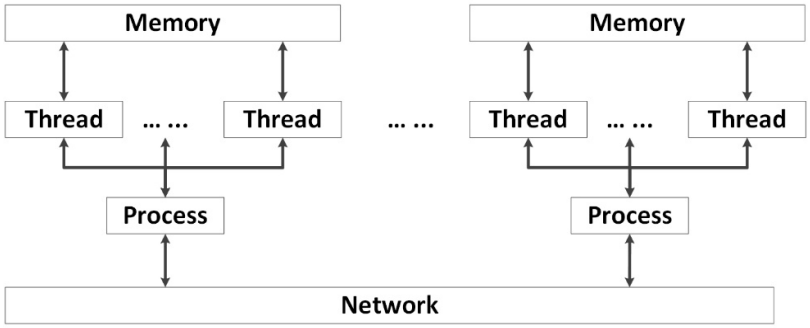


Figure 1.8 Sketch of hybrid programming model.

As shown in Figure 1.8 in the hybrid model, there are two levels of parallelism. One consists of parallel computations among different sub-domains. The physical domain is partitioned into a set of sub-domains which are assigned to and solved by many MPI processes. The other consists of concurrent intensive computations in each sub-domain. In each MPI process, multiple OpenMP threads are set up to perform the intensive computation on each sub-domain concurrently. The hybrid model is characterised by each MPI process being used to acquire a collection of mesh blocks and memory volume. An MPI process is responsible for data exchange between sub-domains through the network. The OpenMP threads are mainly used to perform the intensive computations concurrently in each sub-domain. The hybrid model is well suited to current, common HPC clusters with multi-core nodes.

In comparison to the distributed memory model, the advantage of the hybrid model is mainly the reduction in message passing cost. The performance of an application running in parallel is always proportional to the amount of used CPU cores (MPI processes in the distributed memory model). However, due to load balancing, there must be at least one mesh block in a process, which means the number of available MPI processes is limited to the number of mesh blocks. If more CPU cores are to be used, the physical domain must be divided further into additional, smaller mesh blocks. For a problem with mesh distributed among many small blocks, the granularity is fine-grain in comparison to the same problem partitioned by fewer large mesh blocks. This indicates that higher computational costs are accrued by data communications and related operations like buffer packing and index transformation between two blocks.

In comparison to the shared memory model, the advantage of the hybrid model is that more CPU cores can be used in parallel. Based on the thread model and current available IRIDIS 4 cluster [60], the maximum thread number is 16 (usually equal to CPU cores on a single node). Therefore, if only the shared memory model is used, the application cannot run any more than 16 times faster. The

number of CPU cores available in the hybrid model is a product of the MPI process amount and CPU core amount on each node, which is much larger.

However, there are also drawbacks in hybrid models. First, the hybrid model programming will increase coding complexity and effort, though this is not a major issue if the MPI code is already available. Second, the number of available threads is limited by the number of CPU cores in a computing node, normally 16 in a cluster, whereas the mesh size in a mesh block can be tens of thousands or even larger. The number of available CPU cores in each sub-domain hinders the maximum performance benefit in a single mesh block in the hybrid model. An alternative solution is a heterogeneous model in which 'accelerators' are used to improve the performance in a mesh block. This is increasingly popular in high-end supercomputing in the world's largest computers [59].

1.2.3.4 General Purpose Computing on GPUs

A heterogeneous model that uses 'accelerators' to accelerate the computation can be used to exploit the maximum degree of parallelism within mesh blocks distributed on different processes. A sketch of the heterogeneous model is shown below in Figure 1.9:

Figure 1.9 Sketch of heterogeneous programming model.

The accelerators in Figure 1.9 refer to some hardware architectures that are different from CPU and offer higher computational performance. The architectures of accelerators vary and include many types, such as GPUs [16], Intel Phi co-processors [61], and accelerated processing units (APU) [62]. GPUs and Intel Phi co-processors are the most extensively used [59]. Using accelerators to increase the performance of an HPC cluster is an important feature in recent HPC. The performance of the latest GPU is higher than that of the latest co-processor and GPUs are more extensively used in HPC clusters [16]. In addition, the choice of hardware also depends on the computing resources available. Since this research started in March 2012, there were only GPU nodes on the IRIDIS 3 cluster [60] and no Intel Phi co-processor nodes were available. In this research, GPUs are selected as the accelerator.

A GPU is an indispensable component of a personal computer and is used to process graphic information for displaying. A GPU has higher floating operation power and memory bandwidth than a multi-core CPU, as shown in Figure 1.10 andFigure 1.11 below:

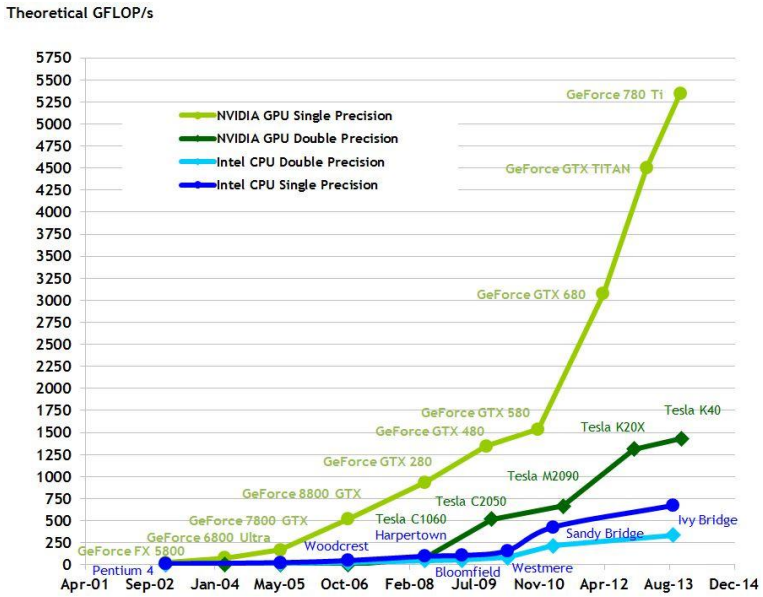


Figure 1.10 Socket floating-point operations per second (FLOPs) for CPUs and GPUs [16].

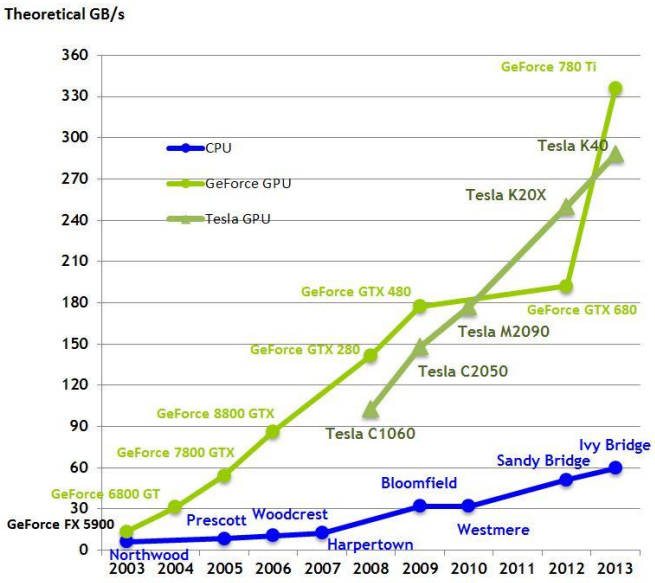


Figure 1.11 Socket memory bandwidth for CPUs and GPUs [16].

As shown in Figure 1.10, the latest GPU's double precision performance is about twice that of a CPU. The memory bandwidth gap between the GPU and CPU is roughly 4.6 times in Figure 1.11. In this

research, the Intel Xeon E5620 CPU and Tesla M2050 GPU were used on the IRIDIS 3 cluster, while Intel Xeon E5-2670 CPU and Tesla K20m GPU are used on the IRIDIS 4 cluster. The double precision operation performance and memory bandwidths are listed in Table 1-2.

Table 1-2 Socket performance comparisons between used CPUs and GPUs

	Intel Xeon E5620	**Tesla M2050**	**Gap**
Flops (Double Precision)	38.4 GFLOPs	515 GFLOPs	13.4
Peak Bandwidth	25.6 GB/s	148 GB/s	5.8
Cache Size	12MB(Level 3)	64 KB	192
	Intel Xeon E5 - 2670	**Tesla K20m**	**Gap**
Flops (Double Precision)	166.4 GFLOPs	1.17 TFLOPs	7.03
Peak Bandwidth	51.2 GB/s	208 GB/s	4.1
Cache Size	20MB(Level 3)	1.25 MB	16

Table 1-2 shows that the performance gap is roughly 7 times for the peak double precision operation and 4 times for the peak memory bandwidth. Many scientists and engineers take advantage of GPU's high performance to conduct general science research and engineering computation which is called GPGPU.

As the GPU is specialized for graphics rendering, which requires parallel computing-intensive computations, more transistors are used for data computing rather than for flow control and caching. Because of the differences in hardware design and implementation between a CPU and a GPU, a GPU code has its own programming model and languages that are distinct from CPU code.

From a GPU's perspective, a computational domain can be treated as a collection of images, whereas an element in a domain can be regarded as a pixel in an image. Multiple developing tools based on different compiling and implementation strategies can be employed to design a code on a GPU, such as CUDA, Open Computing Language (OpenCL) [63, 64] and Open Accelerators (OpenACC) [65, 66]. OpenACC is an extension of existing programming languages including C, FORTRAN and other languages. It is a high-level directive-based parallel library on GPU platforms [67, 68]. It uses compiling derivatives to specify pieces of code implementing on a GPU, which is similar to OpenMP. The detailed implementation of the codes on different GPUs depends on the intelligent compilers. Therefore, the portability of the codes is better. OpenCL is a general developing environment to implement a code on a heterogeneous architecture containing multi-core CPUs and many-core GPUs [63, 64]. In comparison to OpenACC, OpenCL and CUDA are low-level programming interfaces, which expose more flexibility on a GPU to programmers so that the performance is higher [65, 66]. However, the code is more complex and the portability is poor. In addition, CUDA is designed specifically for NVIDIA's GPUs and can obtain higher performance on these GPUs. CUDA is released in C by NVIDIA. In this study, PGI's CUDA FORTRAN [69], which

facilitates the programming on a GPU by an extension of FORTRAN, is used as the development platform as the current code running on multiple CPUs has been coded in FORTRAN and is going to run on NVIDIA's GPUs. A GPU has an array of streaming multiprocessors (SMs) each containing many CUDA cores. An SM is treated as an equivalent CPU core. To offer a brief introduction to CUDA programming model, three key aspects are explained here: thread hierarchy, memory hierarchy and synchronization, with a schematic shown in Figure 1.12:

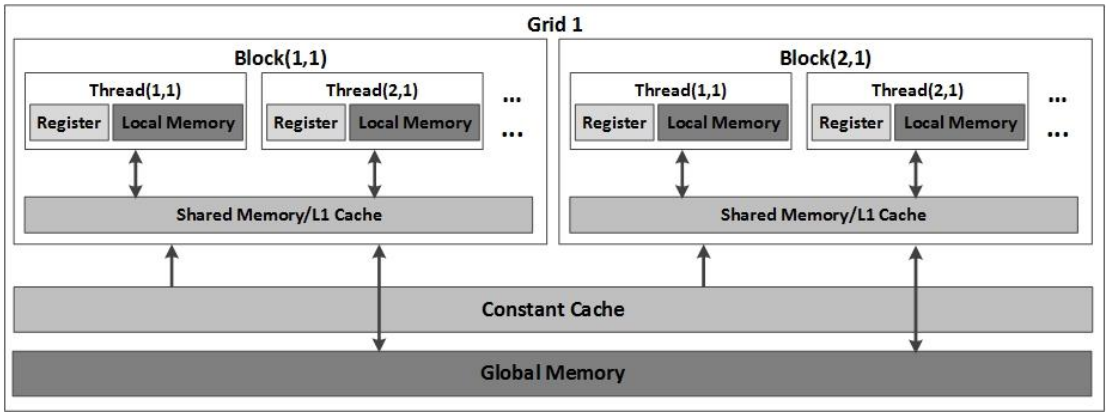


Figure 1.12 The schematics of thread and memory hierarchy in the CUDA programming model.

Note: variant memories are denoted in greys: light grey denotes low access latency, while dark grey represents high access latency. The two-way arrow denotes read and write operation, while the one-way arrow denotes that only read operation is allowed.

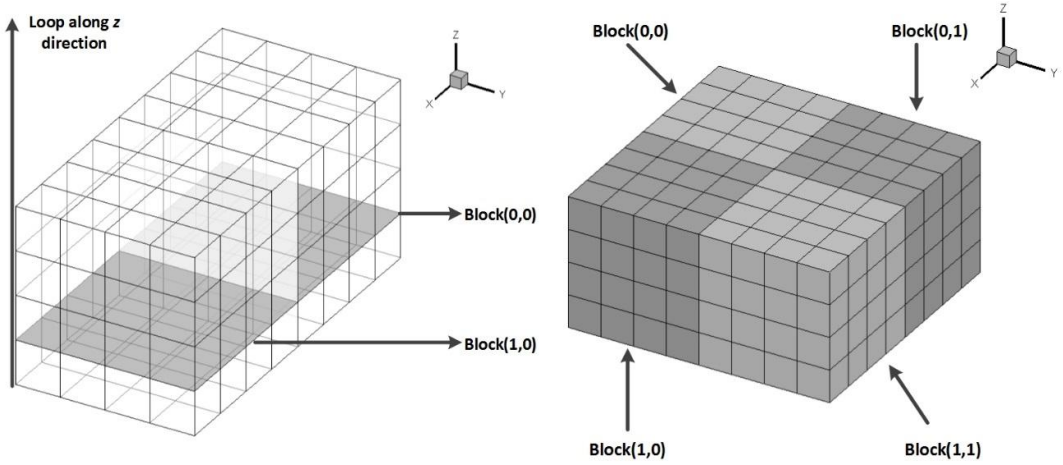
In the CUDA programming model, a subroutine implemented on a GPU is called a kernel. A kernel launches many threads which perform computations on the elements of a computational domain concurrently. Such a large amount of threads is organized into blocks as shown in Figure 1.12. A thread block is a collection of threads residing on an SM. All the blocks in a kernel constitute a grid which is used to distinguish between kernels. A thread can be identified by a derived thread index variable in its local block. Each block has a unique index variable. Thus, a thread in a kernel can be identified globally and a one-to-one correspondence can be made between an element in a computational domain and a thread in a kernel. The thread hierarchy refers to this kind of thread organization in multiple levels.

The memory hierarchy denotes the variant memories and their features. A thread has its own exclusive fast-accessed registers and slow-accessed local memory. Threads in a block can share data via shared memory. The use of shared memory plays a key role in enabling increased performance because the access latency to global memory is many times that of shared memory. The global memory can be accessed by all the threads in a grid. There is an interface between the global

memory on a GPU and the host memory (RAM). The interface refers to the peripheral component interconnect (PCI) express whose bandwidth (8GB/s on PCI-express × 16 Gen2) is far less than a GPU peak bandwidth (144GB/s on Tesla C2050). Thus, memory copy between a CPU and a GPU has to be minimized to extract the maximum performance.

Synchronization is an indispensable component in parallel computing. Local synchronization can be performed in a thread block to ensure that no thread can execute any instructions until all the threads in a block reach the barrier. When there is data dependency between thread blocks, a subroutine must be separated into two kernels. An implicit global synchronization occurs automatically when a kernel terminates similarly to OpenMP.

Furthermore, the key role of coalesced memory access in the performance of GPU codes must be emphasised. It mainly evolves 2D/3D data layout and memory padding on a GPU. The 3D data layout determines how the arrays are accessed by a CUDA thread. In a CUDA FORTRAN code, the 3D data layout corresponds to the kernel execution configuration. In explicit scheme computations, the 3D data layout is mainly realized by two methods: the 'slicing method' and the 'tiling method'. Schematics of the two kinds of data layouts are shown in Figure 1.3:



(a) Slicing Method.

(b) Tiling method.

Figure 1.13 Schematics of 3D data layout in explicit scheme computation on the GPU.

The maximum coalesced memory access can be achieved by using the 'slicing method', as demonstrated by Micikevicius [70] and Lopez et al. [25]. Given a computational domain with dimensions of (*Nx, Ny, Nz*), the slicing method treats the computational domain as *Nz* slices. Each slice is a 2D plane whose dimensions are (*Nx, Ny*). A CUDA thread in the slice sweeps in the *z* direction. The 'tiling method' [25] also allows good coalesced memory access in *x*, *y* and *z* directions.

It treats the computational domain as a mass of bricks. Each brick is a small 3D domain, e.g., with dimensions of (16, 8, 4). In a prefactored compact scheme computation, data layouts are different from the 'slicing method' or 'tiling method', as will be explained in the following section.

Array padding also increases the performance of the application on the GPU since it decreases the global memory access. In the current code, if a random mesh is used, the dimension in the *x* direction is padded to be a multiple of 16, whereas the dimensions in the *y* and the *z* directions remain unchanged. There are two reasons to this: first, a memory transaction loads/writes a memory segment with 128 bytes on the Tesla M2050 GPU. Coalesced memory access is improved by aligning every start element in the *x* direction to a 128-byte segment. Second, the current code is run in double precision format in which 8 bytes are allocated for each element. Therefore, 16 = 128 / 8 is determined. Since the padding only occurs in the *x* direction, no obvious performance loss is observed from its implementation.

1.2.3.5 GPGPU in CAA

The application of GPU computing in the CAA field is quite new and, to wit, only a few study cases have taken place thus far. CAA/CFD solvers that solve partial differential equations by finite difference/volume/element methods share a common feature in that the spatial stencil computation dominates the wall-clock time. For high-order finite difference solvers, stencil computation can achieve 90% of the iteration time or even more, such as the scenario in Table 1-3. The *compactSchemeDz*, *compactSchemeDy* and *compactSchemeDx* subroutines perform Hixon's sixth-order prefactored compact scheme computations and consume 92% of the wall-clock time. Consequently, the performance of a code is mainly attributed to the performance of finite difference computations for both codes on CPUs and GPUs.

Table 1-3 The contribution of the subroutines in a typical 3D computation.

Functions	Implementation time (Seconds)	Fraction
compactSchemeDz	111.2	61%
compactSchemeDy	49.1	27%
compactSchemeDx	8.2	4%
rhsLeeEquation	2.5	1%
explicitFilter10z	2.4	1%
incrementSummation	2.4	1%
compactBoundary	2.2	1%
explicitFilter10y	1.5	1%
bufferZone	1.4	1%
explicitFilter10x	0.2	0%

Due to the dominant wall-clock time of the finite difference scheme computation, the reviews on CAA/CFD codes on GPUs are categorized into two types: explicit stencil and implicit stencil. Abdel-

Hay et al. [71] ported a CAA solver, solving LEE to a GPU for the first time. In their work, the high-order finite difference scheme used was the DRP scheme [11] which is an explicit scheme and is easier to implement on a GPU than a compact scheme which contains a globally dependent system to solve. SotonLEE employs optimized prefactored compact schemes, which is the main difference between the work of Abdel-Hay et al. and SotonLEE. The governing equations, time-marching schemes and boundary conditions are almost the same. The strategies of how to design an explicit scheme solver have been well studied in the CFD [17-21] and room acoustics community [23-25]. As mentioned above, the slicing method and tiling method are employed to achieve the maximum performance for explicit scheme computations on a GPU. Hu [72] accelerated a time domain boundary element method (TDBEM) on multiple GPU cards and applied it to the acoustic scattering case of an airplane. The application ran about 33 times faster on 3 GPUs than on 3 CPU cores.

The GPGPU on CFD has switched the research focus from solvers based on structured meshes to those based on unstructured meshes [44, 73-76] since 2010. The key strategy to improve the performance of an unstructured CFD solver is “index renumbering”, which tries to renumber the indices of unstructured meshes to improve the local features of the flux computations and the memory bandwidth [44, 73-76]. The “index renumbering” is not necessary on a CFD solver using structured meshes since the local feature of the stencil is good. In addition, CFD programmers made more efforts on porting these explicit solvers to multiple GPUs by using OpenACC [67, 68, 74, 77-79]. Two causes were found: First, the strategies of porting algorithms in an explicit CFD solver to GPUs were well developed based on the ‘slicing method’ and the ‘tiling method’. Second, the portability, maintainability and readability of the codes on GPUs started to be emphasized. Another noticeable trend is that some CFD groups tried to create higher level of frameworks to mitigate the programming complexity on GPUs, including CU++ET at University of Wyoming [80] and PyFR at Imperial College London [81]. CU++ET is based on C++ and is available on both structured and unstructured meshes whereas PyFR is a high-order solver using flux reconstruction on unstructured meshes written in Python. These frameworks tried to make minimal changes on existing CFD codes on CPUs by using their own languages and generated kernels implemented on GPUs automatically during compile time. These frameworks were based on developed algorithms on GPUs. Considerable speed-ups on some problems with simple geometries were achieved by using these frameworks.

In this study, the research focus is not placed on explicit schemes or solvers on unstructured meshes but rather on the prefactored compact schemes used in the current solver based on structured meshes. Hixon’s sixth-order and Ashcroft and Zhang’s optimized fourth-order prefactored compact schemes require solving bidiagonal matrices. To wit, the bidiagonal matrix has not been investigated on a GPU before, even in cuBLAS [82] and PETSC [83] libraries. The back substitution is

directly used by subroutine DGTTRS [82] when a tridiagonal matrix is solved in double precision in these libraries. In this work, effort is extended to the implementation of a bidiagonal matrix on GPUs. The solution to a bidiagonal matrix on a CPU is straightforward and in serial by nature. The back substitution or sweep is employed to solve a bidiagonal matrix in LAPACK library [84]. The bidiagonal matrix is similar to the tridiagonal matrix. The tridiagonal matrix has been well investigated on a GPU. Zhang et al. [85] investigated three generic algorithms on a GPU, namely cyclic reduction (CR), parallel cyclic reduction (PCR) and recursive doubling (RD). It showed that the PCR is the fastest generic algorithm for a tridiagonal matrix on a GPU. Then, Stone et al. [86] applied the PCR algorithm to the tridiagonal matrix benchmarks, a speed-up of 3.6 was expected in comparison with a Thomas algorithm on four CPU cores when data transfer was removed. Egloff [87] also implemented the PCR algorithm in his derivative pricing solver and a speed-up of 36 was achieved on two GPUs compared to two CPU cores.

The PCR algorithm treats the tridiagonal matrix as a recursive system. Also, the tridiagonal matrix can be treated as a sparse matrix. Kruger and Westermann [88] realized techniques of linear operators on matrices on a GPU. In a 2D, laminar, incompressible Navier-Stokes equation simulation, the matrix-vector multiplication was used to build the conjugate gradient iterative method for a large sparse matrix. Bolz et al. [89] applied the same method to the same 2D, laminar, incompressible Navier-Stokes equation simulation, and developed the multi-grid technique on a GPU. Tutkun and Edis [90] investigated Lele's sixth-order compact scheme [12] which gave rise to tridiagonal matrices on a CFD code. The generated tridiagonal matrices were directly solved by inverting the coefficient matrix. This could be done conveniently by using the CUDA Basic Linear Algebra Subroutines (cuBLAS) library [82].

While tridiagonal matrix solvers have been investigated on the GPU, little attention has been devoted to a bidiagonal matrix solver. An efficient bidiagonal matrix solver on the GPU is a critical ingredient in some applications that use prefactored compact schemes in aeroacoustic and turbulence studies. In addition, all the studies reviewed above developed the numerical techniques in the *x* direction and then applied the same techniques to all the directions in 2D/3D computations. The anisotropic memory access pattern was ignored when the computation in the *y* and the *z* direction was performed, which caused a loss of overall performance. In this work, an existing CAA solver is extended to run on a multi-GPU platform. The main effort is focused on developing an efficient way to solve the bidiagonal matrix system in solving prefactored compact schemes on GPUs. For this purpose, a fourth-order optimized prefactored compact scheme [14] is selected in this work. The NVIDIA Tesla M2050 GPUs serve as the computing devices and PGI's NVIDIA CUDA FORTRAN is used as the development tool.

1.2.3.6 Performance Metrics

In the CAA/CFD field, the performance of a GPU is roughly equivalent to that of many CPU cores [72, 91-93]. Therefore, using GPUs can significantly reduce wall-clock time. If only CPUs are used, the performance of an application run in parallel is always proportional to the amount of used CPU cores (tasks). Some performance indicators are used to evaluate parallel performance quantitatively, such as speed-up and efficiency. Speed-up is defined as the ratio of wall-clock time of serial execution to wall-clock time of parallel execution. It is the simplest and most widely used indicator and denotes how many times the code runs faster in parallel execution. The maximum speed-up of an application converges to a limited number and is explained in Amdahl's Law [94]:

$$speedup_{\max} = 1/(1-pp) \tag{1.6}$$

in which, *pp* denotes the fraction of code which can be parallelized. If *N* processes are employed to run an application in parallel, then the ideal speed-up is modelled by:

$$speedup = 1/(pp/N + sp) \tag{1.7}$$

where *sp* denotes the fraction of code in serial. Speed-up can be regarded as a function of the number of used CPU cores. However, the speed-up also depends on problem size. When the problem size of a CAA/CFD application is small, the serial fraction is comparable to the parallel fraction. With the increase of the problem size, time spent on the parallel fraction will dominate the wall-clock time of the application.

Efficiency is equal to the ratio of speed-up to number of used processes. It indicates how well the hardware is used to solve the problem compared to the time used in communication. Given a fixed-size problem, the amount of communication increases with the increased number of used MPI processes and thus the efficiency decreases.

In traditional HPC, only CPU cores are used to parallelize applications by using the distributed memory model, shared memory model or hybrid model. The speed-up can be defined exactly in terms of Eq. (1.7). Given an application using the structured multi-block mesh, the mesh is distributed as evenly as possible on CPU cores in terms of load balancing and the maximum CPU cores available. However, in heterogeneous models, the accelerators are employed to parallelize applications. The architecture of an accelerator, such as a GPU, is different from that of a traditional CPU core. Therefore, direct use of Eq. (1.7) in GPGPU is not feasible, which makes a hard evaluation of the speed-up. For example, the Tesla M2050 GPU contains 448 CUDA cores, which are the arithmetic elements and are used in groups, in terms of SMs. Each SM, which can be regarded as an equivalent independent CPU core, contains 32 CUDA cores.

In addition, the definition of speed-up also depends on the solving strategy and implementation. In the current research, the original CAA solver using the multi-block structured mesh, SotonLEE, has been a parallel solver by using MPI. The mesh blocks are distributed across CPU cores in terms of load balancing and the maximum CPU cores available. However, in each mesh block, the implementation on mesh elements is in serial and is performed one mesh element by one mesh element. The current research is to keep the original MPI implementation, and to extend the parallelization on each mesh element with GPU accelerations. The implementation is actually a dual parallelization, in which the first parallelization is at the mesh block level between different CPU cores while the second one is at the mesh element level on GPU cards. The first parallelization is kept the same and the gains of using GPUs are benefited from the second parallelization.

Consequently, in the current research, the speed-up is defined to evaluate the benefit from the reduced wall-clock time between an application by using the MPI+CUDA in comparison and that by only using MPI. The speed-up is defined as the ratio of wall-clock time of MPI implementation to wall-clock time of MPI + CUDA implementation. The definition of the speed-up used in the current research is defined below:

$$speedup = \frac{wall\_clock\_time_{CPU\_cores+GPUs}}{wall\_clock\_time_{CPU\_cores}} \tag{1.8}$$

in which, $wall\_clock\_time_{CPU\_cores+GPUs}$ denotes the implementation time of an application by using MPI + CUDA whereas $wall\_clock\_time_{CPU\_cores}$ denotes the implementation time of an application by using only MPI.

1.2.4 The Scattering of Propeller Tonal Noise

With the multi-GPU implementation capability, the current solver is applied to a large scale engineering application, the scattering of propeller tonal noise off an aircraft, to verify the performance of the new CAA solver. A turboprop/propfan system offers a higher economic efficiency than a turbofan power system and this has been a key driving factor for research on the turboprop/propfan power system [95]. However, the radiating noise from these systems has been a major concern. The prediction of noise generated by an isolated propeller has been well studied and improvement is in progress [95-101]. However, installation effects also play a key role in the prediction of propeller noise in the real world [10, 95, 102]. The acoustic installation effect refers to the difference between the noise generated and propagated in an ideal laboratory environment and that in the real world. The acoustic installation effect can be divided into two categories: The acoustic installation effect on noise generation and that on noise propagation [95]. The acoustic installation effect on noise generation mainly refers to the distortion of inflow and the aerodynamic

interaction between the flow field and the solid geometry [95]. The acoustic installation effect on propagation contains the reflection effect of the aircraft, and refraction effect of the non-uniform flow field close to the aircraft surface. In this study, the CAA hybrid method is used to predict the scattering of the propeller tonal noise off an aircraft. The propeller noise sources are assumed to be known and the installation effect on the noise generation is ignored. The research mainly focuses on the reflection effect off the aircraft and the refraction effect induced by the non-uniform flow field close to the aircraft surfaces.

The propeller noise spectra are characterized by the dominant harmonics, narrow-band random noise and broadband noise [95]. The harmonics are concerned with periodic movements of the blades. Given a propeller with B blades rotating at a constant frequency of f_b, the resulting frequency of fundamental sound harmonic is Bf_b. Other harmonics occur at multiples of BPF. In this study, the research focuses on the propagation of the dominant tonal noise rather than narrow-band random noise or broadband noise.

1.2.4.1 Propeller Noise Source

In terms of the noise generation mechanism, propeller noise sources are categorized into three types: steady sources, unsteady sources and random sources [95, 103]. Steady sources are constant in time for any point on the surface of a rotating blade. For a stationary observer, the acoustic field of steady sources is periodic due to the cyclic rotational movement of the blades. In this study, only steady sources are considered, which means the amplitudes of the noise sources are assumed constant and all phases of the noise sources are the same. Steady noise sources are usually categorized into three types: linear thickness noise, linear loading noise and non-linear quadrupole noise.

Thickness noise comes from the periodic displacement of air due to the volume of rotating blades. Thickness noise can be represented by a distribution of monopoles on blade chords along the span-wise direction. The strength of monopoles is proportional to the blade volume, and dependent on the rotational speed and blade section shape. The contribution of thickness noise is important at high blade tip speeds [95].

Loading noise results from the thrust and torque (lift/drag) on the blade surface. Consequently, it can be described by a distribution of dipoles on blade surfaces. The strength and direction of dipoles are dependent on the thrust and torque [10, 95]. Loading noise is important at low to moderate speeds.

Linear thickness noise and loading noise dominate when the blade section speed is moderate. When the blade section speed is transonic, non-linear effects can become significant and are

modelled by quadrupole sources [95, 96]. They contain non-linear propagation effects and viscous effects. The thickness noise, loading noise and quadrupole noise are obtained by the solution of the general acoustic analogy Ffowcs Williams and Hawkings (FW-H) equation [96]:

$$\left(\frac{\partial^2}{\partial t^2}-c^2\nabla^2\right)\left\{(\rho-\rho_0)c^2H(f)\right\}=\frac{\partial^2\left\{T_{ij}H(f)\right\}}{\partial x_i\partial x_j}-\frac{\partial\left\{L_i\delta(f)\right\}}{\partial x_i}+\frac{\partial\left\{Q\delta(f)\right\}}{\partial t} \tag{1.9}$$

where *H(f)* is the Heaviside step function whereas the control surface is denoted by *f = 0* as shown in Figure 1.14 below:

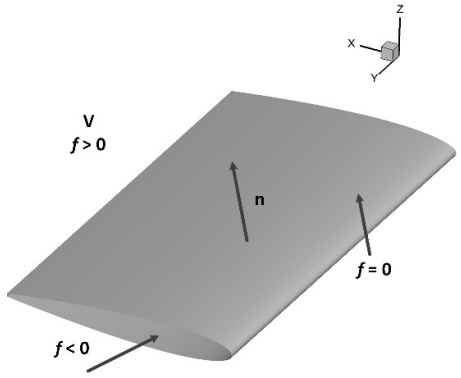


Figure 1.14 The definition of control surface.

As shown in Figure 1.14, the volume space within the control surface is denoted by *f < 0,* whereas *f > 0* represents the volume space outside the control surface. The vector normal to the control surface is $\mathbf{n}=\nabla f$. ∇^2 denotes the Laplace operator. When the density perturbations are small and the observation distance is far away, the term $c^2(\rho-\rho_0)$ can be replaced by *p'*, which is acoustic pressure. *Q* and L_i denote distribution of mass and linear momentum on the control surface. T_{ij} is the Lighthill's stress tensor and contains non-linear propagation effects and viscous effects with the following form:

$$T_{ij}=\rho u_i u_j+(p'-c^2\rho')\delta_{ij}-\tau_{ij} \tag{1.10}$$

where δ_{ij} is the Kronecker Delta function. τ_{ij} is the viscous stress tensor. The non-linear effects are described by $\rho u_i u_j$ whereas τ_{ij} denotes the viscous effects.

In this study, a simple scenario is considered. The airplane is a full scale, ATR-72 500-like wing body [104-106] which performs an initial climbing at a Mach number of 0.205. Only point sources

concentrated on a ring are considered. Due to the moderate blade section speed, the point sources only contain linear thickness noise and loading noise.

Farassat offered a linear theory for the prediction of propeller noise based on the Ffowcs Williams and Hawkings (FW-H) equation [38]. The general form of acoustic analogy can be described by FW-H equations [38]. In terms of linear theory offered by Farassat [96], the linear form of the FW-H equation is shown below:

$$\nabla^2 p' - 1/c^2 \times \partial^2 p'/\partial t^2 = -\partial\left[\rho_0 v_n |\nabla f| \delta(f)\right]/\partial t + \partial\left[l_i |\nabla f| \delta(f)\right]/\partial x_i \quad (1.11)$$

where *p'* is the acoustic pressure, *c* and ρ_0 are the speed of sound and density in the undisturbed medium; v_n is the local normal velocity of the surface *f = 0*:

$$v_n = -\left(\partial f/\partial t\right)/|\nabla f| \quad (1.12)$$

The surface is defined by *f = 0*, with *f > 0* outside the surface and *f < 0* inside it. l_i is the local force on per unit area. $\delta(f)$ is the Dirac delta function. The first term at the RHS denotes thickness noise source, while the second refers to loading noise source. Unless the sound wavelength considered is comparable to the blade section thickness, the detailed shape of the blade section can be ignored. The source terms can be simplified into a volume distribution shown below:

$$\nabla^2 p' - 1/c^2 \times \partial^2 p'/\partial t^2 = -\partial q/\partial t + \nabla \bullet \mathbf{F} \quad (1.13)$$

Now the thickness noise source is a distribution of monopoles with the strength of *q*, while the loading noise source is a distribution of dipoles with the strength of **F**. The values of *q* and **F** can be obtained from an experiment [10, 102] or a CFD simulation [49]. In this study, the strengths of monopoles and dipoles, *q* and **F**, are assumed to be known and a small value is given to ensure a linear acoustic propagation.

1.2.4.2 Acoustic Scattering

Acoustic scattering phenomena can be described by the acoustic installation effect on propagation and the convection effect of the flow field. Acoustic scattering phenomena contain reflection effects off solid surfaces and refraction effects induced by the non-uniform flow field close to the solid wall. Based on different treatments of these two effects, scattering prediction models can be categorized into two types: a) those that focus on the reflection effect and convection effect, and b) those that can account for all the convection, reflection and refraction effects [29, 107]. The reflection and convection effects can be described by an inhomogeneous, convective acoustic equation with a uniform flow [107-109]. Solving the inhomogeneous convective acoustic equation

can be performed by a surface method, such as boundary element method (BEM) [110-112] and equivalent source method (ESM) [107-109]. In the surface method, only a surface mesh is required to discretize the solid surface. Therefore, the mesh generation is much easier and it is computationally efficient. It offers an acceptable estimation for some external flow applications and is mainly used in engineering fields [110-112]. However, the refraction effect is significant in some applications and should not be ignored, e. g., the refraction effect of the boundary layer is strong when the mean flow Mach number is greater than 0.7 [29, 30, 49, 113-116]. In this study, the research focuses on the acoustic scattering induced by the non-uniform mean flow-field around the scattering objects. Sound Pressure Level (SPL) on the scattering object and acoustic far-field directivities will be evaluated by CAA methods.

Some studies on the refraction effect of the non-uniform flow, especially the boundary layer close to a fuselage surface, on sound propagation have been performed. These studies include experiments, analytical methods and numerical methods. These studies focused on the refraction effect at the flight Mach number around 0.8 since the sound on the fuselage surface was assumed to be an important contributor to the cabin noise during cruise. Experiments [113, 117, 118] revealed that considerable shielding effect on the fuselage surface by the boundary layer occurred upstream of the propeller plane. The shielding effect was also studied by analytical methods. Analytical methods [113-117, 119] modelled a fuselage as an infinitely long cylinder with a constant boundary layer along its flow direction and circumferential direction. The velocity gradients in the boundary layer was assumed responsible for the refraction effect. The velocity profile of the boundary layer was specified and only varies in the radial direction. The governing equation in the boundary layer was the shear-flow wave equation which is given by Goldstein [113, 117]:

$$\frac{D}{Dt}\left(\nabla^2 p-\frac{1}{c_0^2}\frac{D^2}{Dt}p\right)-2U'\frac{\partial p'}{\partial x}=0 \tag{1.14}$$

Here the prime denotes $\partial/\partial r$. U' only varies in the radial direction and is constant in the flow and circumferential direction. The convective derivative is:

$$\frac{D}{Dt}=\frac{\partial}{\partial t}+U\frac{\partial}{\partial x} \tag{1.15}$$

The elementary solution was assumed to have the form $P(r)e^{ikx}e^{in\varphi}e^{-i\omega t}$ in the cylindrical coordinates. Then Eq. (1.14) was solved by a Runge-Kutta integration method. The results showed that at a Mach number around 0.8, the shielding effect was strong whereas the shielding effect diminished at lower Mach numbers for large sound wavelengths [114, 117]. The analytical methods showed the following deficiencies in terms of the real geometry and mean flow-field:

- A fuselage geometry was ideally simple, normally an infinitely long cylinder.
- Boundary layer was simplified and was assumed constant along the flow direction and the circumferential direction.

In comparison, the CAA methods show the following advantages:

- A realistic fuselage geometry is considered.
- The refraction effect of a realistic boundary layer is considered.

When a realistic boundary layer and a realistic fuselage geometry were considered, the numerical methods must be used. CAA methods were employed to study the refraction effect recently at cruise Mach numbers and the strong shielding effect upstream of the propeller plane was also found. In CAA methods, the sound sources were coupled to the physical domain by an inflow buffer zone [30, 115] or the RHS sources [10, 49] in the governing equations. LEE or the Euler equations were utilized as the governing equations in propagation. Siefert and Delfs [29] investigated the mean boundary layer effects on sound field on fuselage surface using a point source with different frequencies at Mach numbers 0.7, 0.75 and 0.8. The fuselage was assumed to be a flat plate while the boundary layer was obtained by a RANS simulation. The incoming waves radiated by a point source was coupled into the physical domain by an inflow buffer zone. Then Dierke et al. [30] extended the previous work to the investigation of the turbofan modes on the fuselage surface. Garrec and Reboul [49] applied the CAA method to study the installation effect of the rotor-alone noise of a rear CROR mounted on a single aisle aircraft. The geometry was at full scale and the mean flow-field was obtained by a RANS simulation, which is similar to the current study. The CROR noise sources were modelled by spinning monopolar sources. The result showed that a considerable change on the directivity induced by the non-uniform mean flow was found in comparison to that in a uniform flow. In contrast, the refraction effect of the non-uniform flow on an acoustic wave at a low Mach number was often neglected in the previous research since take-off time is relatively short and the sound on the fuselage surface is assumed to be more important at a cruise condition than at take-off. In the current study, the CAA methods are employed to predict the refraction effect of the boundary layer. As mentioned in Section 1.2.1, LEE can be used to describe all the convection, reflection and refraction effects in acoustic propagation with fidelity. In this study, LEE is selected as the governing equations to investigate the sound scattering off an aircraft, especially the refraction effect of the non-uniform flow at both low and high Mach numbers. The acoustic difference on the solid wall induced by the refraction effect is evaluated.

1.3 Aims of the Current Work

In the literature review, the central part of this research has been outlined. To improve the computational efficiency and reduce the wall-clock time of current CAA simulations, CAA hybrid method and high-order accurate, low-dissipative and low-dispersive schemes have been used in the current CAA solver. Furthermore, the solver can run in parallel with MPI implementation on many CPU cores. Due to the long wall-clock time of current simulations and high computational performance of GPUs, the current solver is extended to implementation on multiple GPUs. The new developed CAA solver is applied to investigate the scattering of the propeller noise off an aircraft and the refraction effects of the mean flow field. The specific aims of the study were to:

- develop an efficient algorithm and strategy to solve a bidiagonal matrix on a GPU to implement optimized prefactored compact schemes;
- develop an efficient CAA solver based on finite difference method on a GPU to reduce the wall-clock time of current CAA simulations;
- optimize the data exchange among multiple GPUs and multiple CPUs;
- apply the new developed CAA solver to investigate the refraction effect of non-uniform mean flow field on the propagation of propeller noise off a cylinder;
- apply the new developed CAA solver to a large scale engineering application: the prediction of propeller noise scattering off an aircraft, and evaluate the speed-up.

1.4 Original Contributions

The original contributions of this thesis are listed below:

- How to solve a bidiagonal matrix is focused on and investigated on a GPU for the first time. The performance of a series of algorithms are tested and evaluated on different mesh sizes in 2D and 3D computations.
- The best generic algorithms to solve a bidiagonal matrix are found on different mesh sizes in 2D and 3D computations on a GPU. The best choice of the algorithms depends on the grid size.
- The reasons why some algorithms behave better is analysed when a bidiagonal matrix is solved on a GPU. Different algorithms are compared and even the same algorithms in the x, y and z directions are compared and analysed.
- Anisotropic memory access pattern is found to be a key role in the performance on a GPU for the same algorithms. For a given algorithm, the memory access is consecutive in the x

direction whereas it is discontinuous in the *y* and *z* directions. This results in different performance on a GPU where the memory access plays a key role in the performance.

- A new hybrid method is proposed in terms of the anisotropic memory access pattern. The hybrid method is a combination of the best choice of different algorithms in the *x*, *y* and *z* directions and achieves the best performance.
- The subroutines in the CAA solver based on finite difference methods are categorized into five types with the aid of memory access patterns. Different parallel strategies are applied accordingly.
- A whole CAA code based on finite difference method is paralleled on multiple GPUs. The solver is efficient and robust on complex geometries. The performance is evaluated on a series of problems.
- The mesh resolution at the source region is found a key role in the propagation of sound waves computationally. In this study, the propeller noise source is introduced by monopoles and dipoles. The resolution of monopoles is represented by a Gaussian distribution and evaluated computationally.
- The validation of the propeller noise source model is performed by a comparison between the CAA results and the analytical solutions of Farassat's Formulation 1A [101] and Hanson's solution [99]. The validation of the propeller noise scattering off a cylinder is performed by a comparison between the CAA results and CESM results computationally.
- The thickness noise, loading noise of a propeller scattering off a cylinder with a realistic boundary layer is investigated by the CAA method for the first time.
- The refraction effect of the realistic boundary layer on a cylinder is investigated by the CAA method. It is found that the refraction effect of boundary layer is weak at M = 0.205 whereas it is significant at M = 0.75. When Mach number is greater than 0.3, the boundary layer plays an important role in the change of far-field directivity.
- The scattering of propeller noise scattering off a wing-body at a full scale is predicted computationally. The speed-up is evaluated on the large scale engineering case.

Some of the work included in this thesis has been presented in the 20th AIAA/CEAS Aeroacoustics Conference [93] and 22nd International Congress on Sound and Vibration [107], and has been published in the Journal of Computer Methods in Applied Mechanics and Engineering [92].

1.5 Structure of Thesis

The remainder of the thesis is organized into three main parts. The first part is devoted to the development of algorithms and the CAA solver on multiple GPUs and is distributed across chapters

2 through 4. The second part (chapter 5) is concerned with the investigation of propeller noise scattering off a cylinder computationally and the refraction effect of the boundary layer. The third part (chapter 6) applies the developed solver to predict the scattering phenomena of propeller noise off a wing-body with complex geometry at a full scale and to evaluate the true speed-up of the current solver in engineering problems. The performance is analysed in detail in comparison to the cases in the second part.

Chapter 2 gives a detailed description of the current CAA solver, called SotonLEE. The technical details include the governing equations, flowchart, numerical schemes, boundary conditions and index manipulations relevant to multi-block mesh and MPI operations. All the subroutines used in SotonLEE are categorized into five types according to different memory access patterns. Different parallel strategies will be applied to the subroutines with different memory access patterns.

In chapter 3, development of an efficient bidiagonal matrix solver on GPUs is described. A small CAA solver which runs in serial on a single CPU core is extracted from SotonLEE. This solver is mainly used as a workbench to develop the algorithms and strategy to solve bidiagonal matrices on a GPU, since it keeps the core algorithms of SotonLEE in a single mesh block in serial and abandons some advanced capabilities such as pre-processing of the mesh information. Three numerical methods are devoted to solving the bidiagonal matrix on GPUs. Furthermore, the anisotropic memory access pattern of all the algorithms are investigated. Finally, combinations of the numerical method, the hybrid methods, are proposed to overcome the deficiency introduced by the anisotropic memory access pattern. The hybrid method achieves the highest performance for large scale computation. The performance of the solvers is recorded and compared. In conclusion, a solving strategy is proposed with the numerical methods to offer a solution to the bidiagonal matrix on GPUs.

Chapter 4 is devoted to the development of other subroutines in SotonLEE and the optimization of the data exchange between multiple GPUs. These subroutines do not contribute much computational cost in comparison to the prefactored compact scheme computations. However, they might be the potential bottleneck if they are not well settled. Subsequently, a series of benchmark problems are implemented on multiple GPUs to evaluate the overall performance of the new solver, called SotonLEE_GPU.

In chapter 5, the acoustic refraction effect of the boundary layer close to an infinitely long cylinder is predicted. The infinitely long cylinder is assumed to be an ideal simplification of a fuselage and the boundary layer is obtained by a RANS simulation. The acoustic sources are a ring of spinning monopoles radiating thickness noise and a ring of spinning dipoles radiating loading noise, which imitate the noise sources of a propeller. The configuration is run with a low Mach number (0.205) setup and a high Cruise number (0.75) setup to investigate the basic features of the acoustic field

of a propeller scattering off a cylinder and the refraction effect of the boundary layer. At last, the acoustic refraction effect of the mean flow field is evaluated at more Mach numbers (0.3 and 0.4) to find out at which Mach number, the refraction of the mean flow-field starts to become important.

In chapter 6, the CAA solver with multi-GPU implementation capability is applied to investigate a large scale engineering case with a complex geometry, the propeller noise scattering off an aircraft. The research focus is not placed on the analysis and explanation on physical phenomena since the engineering geometry is complex and the analysis is difficult. The numerical methods and sound sources are kept the same as those in Chapter 5. The case with a complex geometry is mainly devoted to demonstrate the application to a large-scale engineering case and evaluate the true performance of the current solver in engineering problems. In addition, the speed-up is recorded to evaluate the true speed-up of the current solver in engineering problems. The causes of the relatively low performance of the current solver in this case with complex geometry in comparison to the cases in Chapters 4 and 5 is also analysed in detail.

Chapter 2: High-Order Scattering Solver

In this chapter, the existing CAA code, SotonLEE, which can be used for sound propagation and is used throughout this work, is introduced in detail. The technical details contain governing equations, flowchart, prefactored compact scheme, explicit filter, LDDRK, boundary conditions and operations relevant to MPI communication which include packing/unpacking and array mapping relevant to multi-block mesh. Furthermore, a small CAA scattering code which disables the relevant MPI communication and advanced pre-processing capabilities is extracted from SotonLEE. The flowchart of the basic solver is also given.

2.1 Governing Equations

The inviscid behaviour of small disturbance on a non-uniform, steady flow field can be described by LEE. LEE in 3D Cartesian coordinate can be written in the following form [5]:

$$\frac{\partial \rho'}{\partial t} = -\rho_0\left(\frac{\partial u'}{\partial x}+\frac{\partial v'}{\partial y}+\frac{\partial w'}{\partial z}\right) - u_0\frac{\partial \rho'}{\partial x} - v_0\frac{\partial \rho'}{\partial y} - w_0\frac{\partial \rho'}{\partial z} + s_1 \tag{1.16}$$

$$\frac{\partial u'}{\partial t} = -u_0\frac{\partial u'}{\partial x} - v_0\frac{\partial u'}{\partial y} - w_0\frac{\partial u'}{\partial z} - u'\frac{\partial u_0}{\partial x} - v'\frac{\partial u_0}{\partial y} - w'\frac{\partial u_0}{\partial z} - \frac{1}{\rho_0}\frac{\partial p'}{\partial x} + \frac{\rho'}{\rho_0^2}\frac{\partial p_0}{\partial x} + s_2 \tag{1.17}$$

$$\frac{\partial v'}{\partial t} = -u_0\frac{\partial v'}{\partial x} - v_0\frac{\partial v'}{\partial y} - w_0\frac{\partial v'}{\partial z} - u'\frac{\partial v_0}{\partial x} - v'\frac{\partial v_0}{\partial y} - w'\frac{\partial v_0}{\partial z} - \frac{1}{\rho_0}\frac{\partial p'}{\partial y} + \frac{\rho'}{\rho_0^2}\frac{\partial p_0}{\partial y} + s_3 \tag{1.18}$$

$$\frac{\partial w'}{\partial t} = -u_0\frac{\partial w'}{\partial x} - v_0\frac{\partial w'}{\partial y} - w_0\frac{\partial w'}{\partial z} - u'\frac{\partial w_0}{\partial x} - v'\frac{\partial w_0}{\partial y} - w'\frac{\partial w_0}{\partial z} - \frac{1}{\rho_0}\frac{\partial p'}{\partial z} + \frac{\rho'}{\rho_0^2}\frac{\partial p_0}{\partial z} + s_4 \tag{1.19}$$

$$\frac{\partial p'}{\partial t} = -u_0\frac{\partial p'}{\partial x} - v_0\frac{\partial p'}{\partial y} - w_0\frac{\partial p'}{\partial z} - \gamma p_0\left(\frac{\partial u'}{\partial x}+\frac{\partial v'}{\partial y}+\frac{\partial w'}{\partial z}\right) - \gamma p'\left(\frac{\partial u_0}{\partial x}+\frac{\partial v_0}{\partial y}+\frac{\partial w_0}{\partial z}\right) + s_5 \tag{1.20}$$

Here, u, v and w denote the velocity components in the x, y, z directions respectively. ρ and p denote density and pressure. γ, the ratio of specific heat, is treated as a constant (1.4 for air). The prime symbol denotes perturbation field whereas the subscript 0 represents background mean flow field. In this work, all the variables in LEE are non-dimensional with the following reference scales: a reference length L_0, a reference speed c_0, a reference density ρ_0, a reference pressure $\rho_0 c_0^2$ and a reference time L_0 / c_0. The terms s_1 to s_5 denote the source of disturbance. They are normally set to zero unless certain form of perturbation exists in the flow field and is modelled through these sources explicitly.

2.2 Flowchart

The current CAA solver, SotonLEE, is based on finite difference method using body-fitted, multi-block, structured mesh. The features of SotonLEE has been listed in Section 1.2.2. For a better understanding of the code behaviour, the flowchart of SotonLEE is given below with a description of a typical implementation on multiple CPUs using MPI:

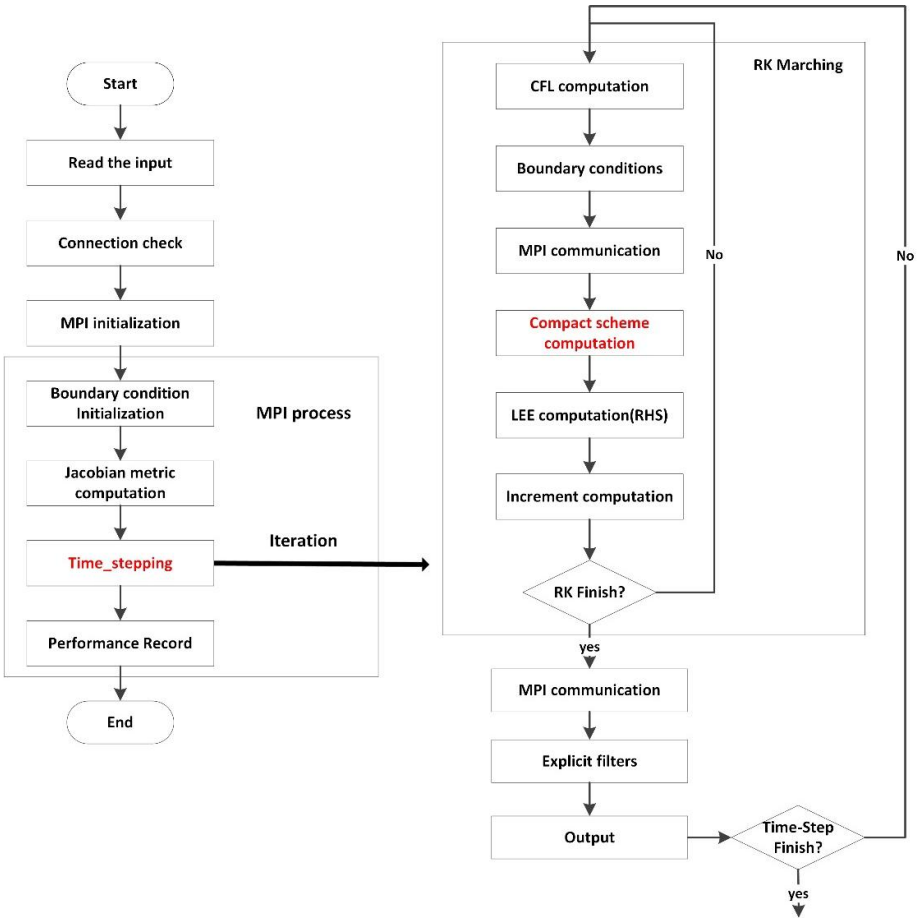


Figure 2.1 Flowchart of SotonLEE.

As shown in the flowchart above, when the program launches, it first performs some initialization operations including reading inputs, checking multi-block connection and launching MPI processes. Then, each MPI process enters into an iteration solver, which is denoted by *Time_stepping* in red in Figure 2.1, after it gets its own mesh blocks and performs the Jacobian metric computation and mean flow field manipulations. In a time step, an MPI process performs the LDDRK iteration, then explicit filtering and if necessary, output to hard disk. In the LDDRK iteration, there are CFL calculations, boundary treatments, spatial derivative and LEE RHS residual increment computations. When the iteration finishes, the program finalizes MPI processes and records performance metrics.

In the whole implementation, spatial derivative computations using the prefactored compact schemes are the most computing-intensive part, as shown in Table 1-3.

2.3 Prefactored Compact Schemes

The high-order, low-dissipative and low-dispersive prefactored compact schemes increase the maximum resolved wave number and reduce the total mesh element amount as stated in section 1.2.2. Furthermore, the prefactored compact schemes split the derivatives into the sum of a forward derivative and a backward derivative. This operation reduces the stencil size and simplifies the implementation close to boundaries [13, 14]. Meanwhile, this operation results in two bidiagonal matrices which are easier to solve on a CPU via a forward/backward sweep operation.

Given a computational domain with dimensions of (*Nx*, *Ny*, *Nz*), the fourth-order optimized prefactored compact scheme in the *x* direction can be expressed in a recursive form as shown below:

$$D_{i,j,k} = \frac{1}{2}\left(D^{F}_{i,j,k} + D^{B}_{i,j,k}\right) \tag{1.21}$$

$$\alpha D^{F}_{i+1,j,k} + \beta D^{F}_{i,j,k} = \frac{1}{\Delta x}\left[a_F f_{i+1,j,k} + b_F f_{i,j,k} + c_F f_{i-1,j,k}\right] \tag{1.22}$$

$$\beta D^{B}_{i,j,k} + \alpha D^{B}_{i-1,j,k} = \frac{1}{\Delta x}\left[a_B f_{i+1,j,k} + b_B f_{i,j,k} + c_B f_{i-1,j,k}\right] \tag{1.23}$$

in which, $D_{i,j,k}$, $D^{F}_{i,j,k}$ and $D^{B}_{i,j,k}$ denote the unknown primitive derivative, the forward derivative and the backward derivative on a mesh point respectively, whereas $f_{i,j,k}$ represents the known primitive field. The coefficients, α, β, a_F, b_F, c_F, a_B, b_B and c_B, are constants. For a given *j* and *k*, *i* varies in descending order from *Nx* - 1 to 2 in Eq. (2.7) and *i* varies in ascending order from 2 to *Nx* - 1 in Eq. (2.8). On the outer edge of a mesh block, a third-order four-point stencil explicit scheme (forward and backward combined) is applied:

$$D^{B}_{1,j,k} = \sum_{i=1}^{4} s_i f_{i,j,k} \tag{1.24}$$

$$D^{B}_{Nx,j,k} = \sum_{i=Nx-3}^{Nx} e_i f_{i,j,k} \tag{1.25}$$

$$D^{F}_{1,j,k} = \sum_{i=1}^{4} -e_{Nx+1-i} f_{i,j,k} \tag{1.26}$$

$$D^F_{Nx,j,k} = \sum_{i=Nx-3}^{Nx} -s_{Nx+1-i} f_{i,j,k} \tag{1.27}$$

On a block interface, a fourth-order eleven-point stencil explicit scheme (forward and backward combined) is applied:

$$D^F_{i,j,k} = \sum_{m=-5}^{5} b_m f_{i+m,j,k} \tag{1.28}$$

$$D^B_{i,j,k} = \sum_{m=-5}^{5} -b_{-m} f_{i+m,j,k} \tag{1.29}$$

where s_i, e_i and b_i are fixed coefficients. Eqs. (2.6) - (2.14) constitute a recursive system. This recursive system can be represented in matrix form:

$$\mathbf{A}^F \mathbf{d}^F = \mathbf{b}^F = \mathbf{B}^F \mathbf{f} \tag{1.30}$$

$$\mathbf{A}^F \mathbf{d}^F = \begin{Bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \beta & \alpha & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \beta & \alpha & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \beta & \alpha & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \beta & \alpha \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{Bmatrix} \begin{Bmatrix} D^F_{1,j,k} \\ D^F_{2,j,k} \\ D^F_{3,j,k} \\ \vdots \\ \vdots \\ D^F_{Nx-2,j,k} \\ D^F_{Nx-1,j,k} \\ D^F_{Nx,j,k} \end{Bmatrix} \tag{1.31}$$

$$\mathbf{b}^F = \begin{Bmatrix} b^F_{1,j,k} & b^F_{2,j,k} & b^F_{3,j,k} & \cdots & \cdots & b^F_{Nx-2,j,k} & b^F_{Nx-1,j,k} & b^F_{Nx,j,k} \end{Bmatrix}^T \tag{1.32}$$

$$\mathbf{B}^F \mathbf{f} = \begin{Bmatrix} -e_{Nx} & -e_{Nx-1} & -e_{Nx-2} & -e_{Nx-3} & \cdots & 0 & 0 & 0 \\ c_F & b_F & a_F & 0 & \cdots & 0 & 0 & 0 \\ 0 & c_F & b_F & a_F & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & b_F & a_F & 0 \\ 0 & 0 & 0 & 0 & \cdots & c_F & b_F & a_F \\ 0 & 0 & 0 & 0 & \cdots & -s_3 & -s_2 & -s_1 \end{Bmatrix} \begin{Bmatrix} f_{1,j,k} \\ f_{2,j,k} \\ f_{3,j,k} \\ \vdots \\ \vdots \\ f_{Nx-2,j,k} \\ f_{Nx-1,j,k} \\ f_{Nx,j,k} \end{Bmatrix} \tag{1.33}$$

$$\mathbf{A}^B \mathbf{d}^B = \mathbf{b}^B = \mathbf{B}^B \mathbf{f} \tag{1.34}$$

$$\mathbf{A}^B\mathbf{d}^B=\begin{Bmatrix}1&0&0&0&\cdots&0&0&0\\ \alpha&\beta&0&0&\cdots&0&0&0\\ 0&\alpha&\beta&0&\cdots&0&0&0\\ \vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\ \vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\ 0&0&0&0&\cdots&\beta&0&0\\ 0&0&0&0&\cdots&\alpha&\beta&0\\ 0&0&0&0&\cdots&0&0&1\end{Bmatrix}\begin{Bmatrix}D^B_{1,j,k}\\ D^B_{2,j,k}\\ D^B_{3,j,k}\\ \vdots\\ \vdots\\ D^B_{Nx-2,j,k}\\ D^B_{Nx-1,j,k}\\ D^B_{Nx,j,k}\end{Bmatrix} \tag{1.35}$$

$$\mathbf{b}^B=\begin{Bmatrix}b^B_{1,j,k} & b^B_{2,j,k} & b^B_{3,j,k} & \cdots & \cdots & b^B_{Nx-2,j,k} & b^B_{Nx-1,j,k} & b^B_{Nx,j,k}\end{Bmatrix}^T \tag{1.36}$$

$$\mathbf{B}^B\mathbf{f}=\begin{Bmatrix}s_1&s_2&s_3&s_4&\cdots&0&0&0\\ c_B&b_B&a_B&0&\cdots&0&0&0\\ 0&c_B&b_B&a_B&\cdots&0&0&0\\ \vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\ \vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\ 0&0&0&0&\cdots&b_B&a_B&0\\ 0&0&0&0&\cdots&c_B&b_B&a_B\\ 0&0&0&0&\cdots&-e_{Nx-2}&-e_{Nx-1}&-e_{Nx}\end{Bmatrix}\begin{Bmatrix}f_{1,j,k}\\ f_{2,j,k}\\ f_{3,j,k}\\ \vdots\\ \vdots\\ f_{Nx-2,j,k}\\ f_{Nx-1,j,k}\\ f_{Nx,j,k}\end{Bmatrix} \tag{1.37}$$

Eqs. (2.15) - (2.22) constitute the matrix form of the fourth-order optimized prefactored compact scheme. Solving the bidiagonal matrix on CPUs is quite straightforward and simple. Forward substitution is employed to solve the lower bidiagonal matrices whereas backward substitution is used to solve the upper bidiagonal matrix in serial. In a 3D domain, there are *Ny* × *Nz* upper and lower bidiagonal matrices to solve. These matrices can be solved independently by a shared memory model. A 3D computational domain exposes a massive degree of parallelism to the CPUs, thus making the algorithm highly efficient. The fourth-order optimized prefactored compact schemes in the *y* and *z* direction are solved in the similar way with varying index *j* and *k*, respectively.

2.4 Explicit Filters

A finite difference scheme always has a limited resolved wave number. The waves with high wave numbers which are not resolved with accepted fidelity are called spurious waves and must be removed. In CAA simulations, there are two ways to remove spurious waves: one way is using artificial damping [11] and the other is using spatial filtering [12, 120-122]. Artificial damping adds extra terms to the finite difference scheme computation and the introduced cost is comparable to that of the finite difference scheme. Another method is using spatial filtering. Spatial filtering can be performed every multiple time step so that the introduced cost is much lower in comparison to that of the finite difference scheme. In this work, explicit filters are used. A tenth order explicit filter [41, 42] is written in the following form:

$$\begin{aligned}\overline{f}_i = f_i - c_5 \times \left(f_{i-5} + f_{i+5}\right) - c_4 \times \left(f_{i-4} + f_{i+4}\right) - c_3 \times \left(f_{i-3} + f_{i+3}\right) \\ - c_2 \times \left(f_{i-2} + f_{i+2}\right) - c_1 \times \left(f_{i-1} + f_{i+1}\right) - c_0 \times f_i\end{aligned} \tag{1.38}$$

in which, $\overline{f}_i$ denotes the filtered field while f_i is the primitive field. Eq. (2.23) describes the filter in the interior points with a stencil size of 11. The interior stencil is symmetric. When *i < 6* or *i > Nx - 4*, the boundary stencils must be used. The boundary stencils are asymmetrical and use different formulations at varying positions:

at *i = 1*:

$$\overline{f}_1 = f_1 - \sum_{j=1}^{6} c_{1,j} \times f_j \tag{1.39}$$

at *i = 2*:

$$\overline{f}_2 = f_2 - \sum_{j=1}^{7} c_{2,j} \times f_j \tag{1.40}$$

at *i = 3*:

$$\overline{f}_3 = f_3 - \sum_{j=1}^{8} c_{3,j} \times f_j \tag{1.41}$$

at *i = 4*:

$$\overline{f}_4 = f_4 - \sum_{j=1}^{9} c_{4,j} \times f_j \tag{1.42}$$

at *i = 5*:

$$\overline{f}_5 = f_5 - \sum_{j=1}^{10} c_{5,j} \times f_j \tag{1.43}$$

The similar stencils apply to the points with *i > Nx - 5*. The filtering performance of the tenth-order explicit filter is given in Figure 1.3.

2.5 Low-Dissipation and Low-Dispersion Runge-Kutta Scheme

In collaboration with high-order, low-dissipative and low-dispersive finite difference schemes, requirements of low-dissipation, low-dispersion and high-order accuracy are also made on time marching schemes. The time marching schemes attempt to start at an initial time and advance a small time interval forward to obtain the solution at the next time step. In CAA simulations, the explicit time marching scheme is preferable since the time interval must be small enough to retain

the accuracy, low dissipation and low dispersion. In addition, the computational cost is much lower in comparison to an implicit time marching method if the same time interval is used. Solving methods on explicit schemes are also much simpler as implicit time schemes always require solving a globally dependent matrix [123, 124]. The explicit time marching schemes can also be categorized into two types: multi-step type [11, 125] and multi-stage type [40, 126]. A multi-step method utilizes the information at multiple previous steps for extrapolation to achieve high-order accuracy. In contrast, a multi-stage method discards the information at multiple previous steps. It performs computations at multiple intermediate time stages between the current and the next time steps to obtain high-order accuracy. Due to this difference, the multi-step method gains much higher computational efficiency since there is only one computation between two time steps whereas the multi-stage method has to compute the finite difference scheme, filtering, boundary condition, RHS residual and performs MPI data exchange within each intermediate time stage. However, if high-order accuracy must be retained, the memory volume required by a multi-step method is much larger than a multi-stage method. This is a key limitation in some applications, such as applications on GPUs, since the memory volume of a GPU is limited. The prototype of a multi-step method is Adams-Bashforth method [125], whereas Runge-Kutta method is a typical multi-stage method. In this research, the LDDRK with alternating 4/6 stages [40] has been used. The governing equation has the following form:

$$\partial \mathbf{U} / \partial t = F\left(\mathbf{U}\right) \tag{1.44}$$

A *p*-stage LDDRK scheme can be written in the following form:

$$\mathbf{K}_i = \Delta t F\left(\mathbf{U}^n + \overline{\beta_i}\mathbf{K}_{i-1}\right) \tag{1.45}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \mathbf{K}_p \tag{1.46}$$

in which, $\mathbf{K}_i$ is the incremental vector at an intermediate stage, while **U** is the solution vector. $\overline{\beta_i}$ is the constant coefficient. Details of the coefficients for the LDDRK with alternating 4/6 stages can be found in Hu's work [40].

2.6 Boundary Conditions

The boundary conditions determine the numerical solution if numerical methods and a CAA mesh are given. In this work, the inflow, outflow, symmetry and inviscid wall boundary conditions are utilized and will be introduced below.

2.6.1 Inflow Condition

The inflow condition is used to introduce the input of perturbation when the perturbation is not easy or convenient to show as a source term explicitly in the governing equations. In this work, the inflow condition is a buffer zone which radiates the disturbance outwards. A typical inflow condition used in the scattering of a spinning mode off a duct [34-37, 127] is shown below:

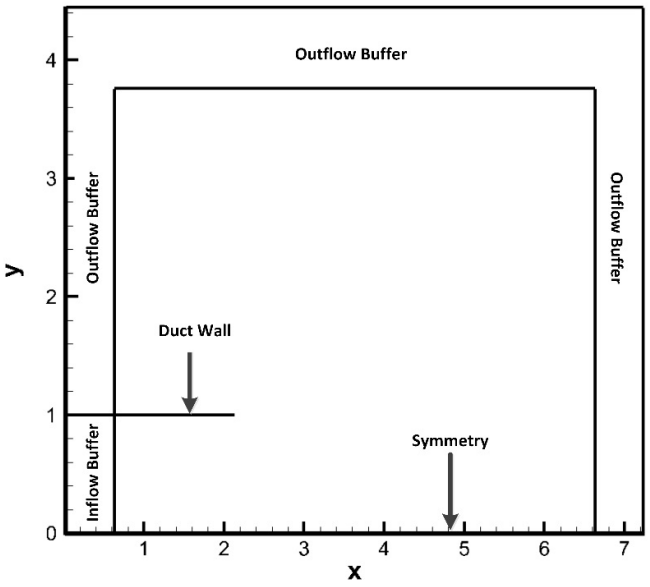


Figure 2.2 Boundary conditions in the scattering of a spinning mode off a duct case.

In Figure 2.2, the inflow condition which contains the spinning mode of the duct is located at the left bottom corner. In the inflow condition, the acoustic field is forced into a spinning duct mode. The acoustic field which enters into the inflow buffer is damped by the buffer. The inflow buffer condition is given in the following form:

$$\widetilde{\mathbf{U}}^{n+1} = \mathbf{U}^{n+1} - \sigma(x) \times \left(\mathbf{U}^{n+1} - \mathbf{U}_{target} \right) \tag{1.47}$$

$$\sigma(x) = \sigma_{\max} \left| 1 - x/L \right|^{\beta} \tag{1.48}$$

where $\widetilde{\mathbf{U}}$ denotes the damped solution vector and $\mathbf{U}_{target}$ is the acoustic spinning mode of the duct. $\sigma(x)$, the damping coefficient, varies smoothly throughout the buffer zone. x is the distance from the starting position of the inflow buffer, while σ_{max} and β are constant coefficients. In the inflow

condition, the acoustic waves are damped out and forced to be the acoustic spinning mode of the duct.

2.6.2 Outflow Condition

The outflow condition in this work refers to the artificial computational boundaries which truncate the flow region of interest into a limited domain. The outflow condition endeavours to reduce the domain with the minimum feedback on the flow/acoustic field. Based on the different phenomena and treatments, there are two types of outflow conditions: the non-reflect boundary conditions (NRBC) [128-130] and the absorbing layers [131, 132]. When the flow field near the boundary can be linearized into small disturbance about a uniform flow, NRBC performs manipulations on the boundaries so that no incoming waves exist. However, when the mean flow field is not uniform or the flow field near the boundary cannot be linearized into small perturbation to a uniform flow field, NRBC is not available [128-130]. The NRBC can also be extended into the non-uniform mean flow field. However, when the perturbation is not propagating normally to the boundary, it works less well [131, 132]. In this situation, the buffer zone that is one of the absorbing layers gains the best performance [53, 133]. The outflow buffer zone condition is nearly the same as the inflow buffer condition shown in Eqs. (2.32) – (2.33) except that $\mathbf{U}_{target}$ in the outflow buffer is normally set to zero to damp the waves. Another method is grid stretching [120, 131, 134, 135] which abandons boundary condition at the outer edge. The mesh element is stretched to a large distance close to the boundary so that the disturbance cannot be well resolved and is removed by explicit filters. A typical stretched grid close to the boundary is shown below:

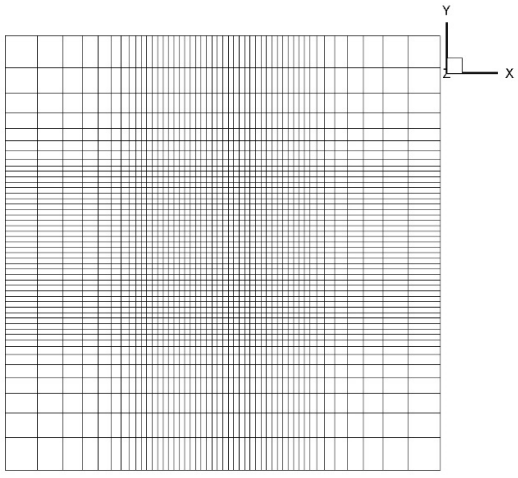


Figure 2.3 Grid stretching.

In Figure 2.3, the mesh elements close to the outflow boundary are stretched so that the perturbation waves become spurious waves which are removed by the explicit filters gradually. Furthermore, the buffer zone condition [53, 136, 137] can be utilized to attenuate the acoustic field further, together with the grid stretching. In this research, the buffer zone and grid stretching are used as the outflow boundary condition depending on different cases.

2.6.3 Inviscid Wall Condition

Inviscid wall condition is another key boundary condition in CAA simulations. As mentioned in sections 2.3 and 2.4, the high-order, low-dissipative and low-dispersive prefactored compact scheme together with explicit filter does not sustain spurious waves. However, the spurious waves always results from improper imposition of wall condition [138]. The improper imposition of inviscid wall condition even results in parasite waves which have very short wavelengths (grid to grid oscillations) and cannot be dampened or removed by spatial filtering effectively [138]. The parasite waves are confined locally in space which contaminate and even blow up the numerical solution occasionally. Therefore, proper imposition of inviscid wall condition plays an important role which does not excite the spurious waves, thereby improving the quality of the solution.

The inviscid wall condition requires that the velocity component normal to the wall and also its derivative with respect to time are zero. In this research, the implementation of inviscid wall condition is based on the inviscid wall condition of Tam and Dong [138] and that of Hixon [13, 139-141]. It is shown in following form:

$$\partial p/\partial n = \partial p/\partial \xi \times \partial \xi/\partial n + \partial p/\partial \eta \times \partial \eta/\partial n + \partial p/\partial \zeta \times \partial \zeta/\partial n = 0 \tag{1.49}$$

$$\partial \xi/\partial n = \partial \xi/\partial x \times \partial x/\partial n + \partial \xi/\partial y \times \partial y/\partial n + \partial \xi/\partial z \times \partial z/\partial n \tag{1.50}$$

$$\partial \eta/\partial n = \partial \eta/\partial x \times \partial x/\partial n + \partial \eta/\partial y \times \partial y/\partial n + \partial \eta/\partial z \times \partial z/\partial n \tag{1.51}$$

$$\partial \zeta/\partial n = \partial \zeta/\partial x \times \partial x/\partial n + \partial \zeta/\partial y \times \partial y/\partial n + \partial \zeta/\partial z \times \partial z/\partial n \tag{1.52}$$

where n is the modulus of the normal vector on wall, ξ, η and ζ are variables in the computational domain. $\partial \xi/\partial x$, $\partial \xi/\partial y$, $\partial \xi/\partial z$; $\partial \eta/\partial x$, $\partial \eta/\partial y$, $\partial \eta/\partial z$; $\partial \zeta/\partial x$, $\partial \zeta/\partial y$ and $\partial \zeta/\partial z$ are grid metrics whereas $\partial x/\partial n$, $\partial y/\partial n$ and $\partial z/\partial n$ are normal vector components of the wall. Given an inviscid wall simply at $y = 0$, an illustration of the implementation is described below:

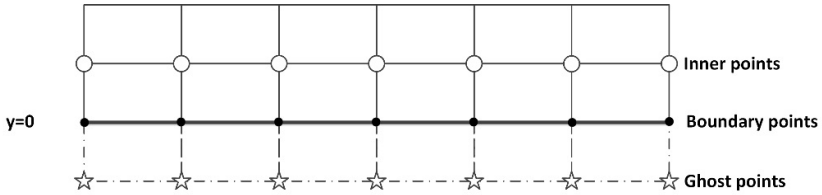


Figure 2.4 Region close to wall.

As shown in Figure 2.4, there are three types of points close to the inviscid wall, namely inner points, wall points and ghost points. Recall that in terms of Eqs. (2.7) and (2.8), the computation of the forward and backward derivatives at point *i* requires three points at *i - 1*, *i*, and *i + 1*. The inner points starting from 2 to *Nx - 1* are solved by Eqs. (2.7) and (2.8). At boundary points with *i = 1* and *i = Nx*, the forward and backward derivatives are solved by biased schemes in Eqs. (2.9) – (2.12) close to the wall. However, it is found that direct application of the biased schemes on an inviscid wall in a prefactored compact scheme is not stable. Therefore, a change is necessary to implement a prefactored compact scheme close to the wall. If the inviscid wall locates at *y = 0 (i = 1)* as shown in Figure 2.4, the forward and backward derivatives on both boundaries with *i = 1* and *i = Nx* are computed via a biased stencil in Eqs. (2.9) - (2.12). Then, a correction is made on the backward derivative at point *i = 1* so that the resultant pressure gradient with respect to normal direction is zero. Lastly, the inner points are solved via Eq. (2.7) by backward substitution and Eq. (2.8) by forward substitution. In 3D computations, there are six faces in a structured mesh block; the implementation of the inviscid wall condition can be concluded below:

1) calculation of the pressure derivative normal to the wall by using the biased stencils;
2) judgement of segment index;

If segment index is 1 or 2, use the formulation below:

$$\partial p/\partial \xi = -\left(\partial p/\partial \eta \times \partial \eta/\partial n + \partial p/\partial \zeta \times \partial \zeta/\partial n\right)/\left(\partial \xi/\partial n\right) \tag{1.53}$$

For segment 1, the forward derivatives at point *i* = 1 and *i* = *Nx* are determined by Eqs. (2.11) and (2.12), whereas the backward derivative at point *i* = *Nx* is computed by Eq. (2.10). However, the backward derivative at point *i* = 1 is not solved by Eq. (2.9); it is determined by enforcing $\partial p/\partial \xi$ = 0:

$$\partial p/\partial \xi = 0.5 \times \left(D_i^F + D_i^B\right) \tag{1.54}$$

$$D_i^B = 2 \times \partial p/\partial \xi - D_i^F \tag{1.55}$$

For segment 2, the backward derivatives at point *i* = 1 and *i* = *Nx* are determined by Eqs. (2.9) and (2.10), whereas the forward derivative at point *i* = 1 is computed by Eq. (2.11). However, the forward derivative at point *i* = *Nx* is not solved by Eq. (2.10); it is determined by:

$$D_i^F = 2 \times \partial p / \partial \xi - D_i^B \tag{1.56}$$

If segment index is 3 or 4, then replace Eq. (2.38) with the following equation:

$$\partial p / \partial \eta = -\left(\partial p / \partial \zeta \times \partial \zeta / \partial n + \partial p / \partial \xi \times \partial \xi / \partial n\right) / \left(\partial \eta / \partial n\right) \tag{1.57}$$

If the segment index is 5 or 6, replace Eq. (2.38) with:

$$\partial p / \partial \zeta = -\left(\partial p / \partial \xi \times \partial \xi / \partial n + \partial p / \partial \eta \times \partial \eta / \partial n\right) / \left(\partial \zeta / \partial n\right) \tag{1.58}$$

3) The inner points are solved by Eqs. (2.7) and (2.8).

2.6.4 Symmetry Condition

The symmetry condition is used when flow field/acoustic field is symmetric with respect to a geometry. When the geometry can be found, the symmetry condition can be utilized to reduce the computational cost artificially. In comparison to the inviscid wall condition, the symmetry condition always appears as a simple geometry, such as a flat plane. Therefore, a simple mirror operation is performed on five ghost points outside the physical domain. Then, the symmetric condition is treated as an interface between mesh blocks and therefore the explicit symmetric schemes in Eqs. (2.13) and (2.14) are utilized to obtain the derivatives. In this work, the method mentioned above is used.

2.7 Index Transformation, Buffer Packing and Unpacking

The index transformation is necessary for multi-block structured mesh especially when an O-mesh is used [142]. This situation is demonstrated by a typical O-H-mesh, which is shown below:

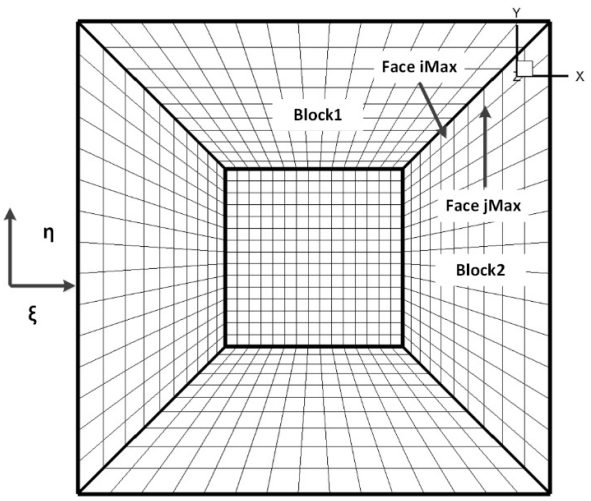


Figure 2.5 Sketch of a typical mesh with the O-H topology.

As shown in Figure 2.5, the core of the domain is an H-mesh with a surrounding O-mesh. The face iMax in block 1 is in the ξ direction in the computational domain whereas face jMax in block 2 is in the ξ direction. The face iMax in block 1 becomes face jMax rather than iMin in block 2. This kind of index transformation exists in the O-mesh and introduces extra computational cost into the code. In this work, the index transformation utilizes the definition of transform in CFD general notation system (CGNS) [142]. For example, at the interface between block 1 and block 2 in Figure 2.5,, the shorthand notation of transformation is denoted by a transform vector [-2, 1, 3]. The first component -2 means a minus increment occurs at *j* at the interface in block 2 when a positive increment occurs on the element index at *i* at the interface in block 1. The second component 1 denotes when a positive increment occurs on the element index at *j* at the interface in block 1, a positive increment occurs at *i* at the interface in block 2. In SotonLEE, the flow field close to the interface in an adjacent mesh block is mapped as ghost points in the current block. Every time the information on ghost points is required, the code first maps the flow field at the adjacent block into the ghost points, then performs computation. The index transformation is independent of MPI implementation but is only relevant to the multi-block mesh.

The packing and unpacking operations are only concerned with MPI implementation on multiple CPU cores. When data exchange of flow field close to the block interface occurs, the data which needs exchange in the array is first packed into a small buffer array and then sent out. The buffer array received from a neighbouring mesh block in another MPI process has to be unpacked. This kind of packing and unpacking operation is necessary for MPI implementation.

2.8 Subroutine Categories

As mentioned in section 1.2.3.4, memory access patterns play important roles in the performance of GPU codes. It is necessary to categorize iteration subroutines into five types according to different memory access patterns, following the method proposed by Elsen et al. [17]. Different types of subroutines are applied with different parallel strategies, which are discussed below.

First, subroutines such as prefactored compact scheme computations, which always require solving a recursive system (bidiagonal or tridiagonal matrix), are categorized as implicit stencil type. Second, subroutines such as explicit high-order filters, which demand regular information from neighbouring elements, are categorized as explicit stencil type. Third, subroutines such as LEE RHS residual computations, inflow and outflow conditions, which only depend on local information, are categorized as point-wise type. Fourthly, unstructured gather type subroutines, such as those packing/unpacking operations and index transformation appeared in the interface condition, are used to deal with the connectivity information on interfaces for adjacent blocks. This kind of subroutines require irregular access to the memory and are considered unstructured gather type. Lastly, reduction type subroutines are employed to collect the maximum, the minimum or the sum of some specific variables, such as the CFL calculation. This kind of subroutine performs a traversal across the whole domain. Different categories of subroutines are applied with different parallel strategies and algorithms on GPUs, as will be explained in detail in the next two chapters.

2.9 Summary

In this chapter, the high-order CAA code, SotonLEE, which can be used for sound propagation, has been introduced in detail. The description contains governing equations, flowchart, CAA numerical methods, boundary conditions, multi-block index transformation, and operations relevant to MPI. Finally, all the subroutines have been categorized into five types according to their memory access pattern which is important in GPU programming in subsequent chapters.

Chapter 3: Efficient Bidiagonal Matrix Solver on a GPU

This chapter is devoted to the solution of prefactored compact schemes on a GPU. The prefactored compact schemes give rise to bidiagonal matrices. Therefore, an efficient bidiagonal matrix solver is a critical ingredient and dominates the overall performance of applications on a GPU. Furthermore, previous researchers always developed numerical methods in the *x* direction and applied the methods directly in the *y* and *z* directions in 3D computations [85, 143-145]. The anisotropic memory access pattern, which degrades the overall performance of the numerical methods, was ignored. In this chapter, the anisotropic memory access pattern is also investigated. Finally, a solution of bidiagonal matrix on a GPU is proposed. It contains the implementation strategy and algorithms. The performance is recorded and analysed on a simple CAA case.

3.1 A Simple Serial Solver

SotonLEE is a large and complex suite of CAA programs. It employs finite difference method and body-fitted, multi-block, structured mesh. It has complex control structures, complex initialization operations and index transformations. Consequently, it is better to first investigate the core algorithms on a small serial solver which retains the same numerical methods as in SotonLEE. For shorthand notation, this small serial solver is hereafter called BenLEE since it serves as a workbench to investigate the algorithms used in SotonLEE. The BenLEE contains typical features in SotonLEE shown below:

- Written in FORTRAN90
- Single block uniform mesh
- Ashcroft and Zhang's fourth-order optimized prefactored compact scheme [14]1 and Hixon's sixth-order prefactored compact scheme [13]
- Tenth-order and sixth-explicit filters [41, 42]
- Hu's alternating 4-6 stages LDDRK time-marching scheme [40]
- Outflow buffer zone condition [131, 132]
- In double precision format
- In non-dimensional format

The flowchart of BenLEE is shown below:

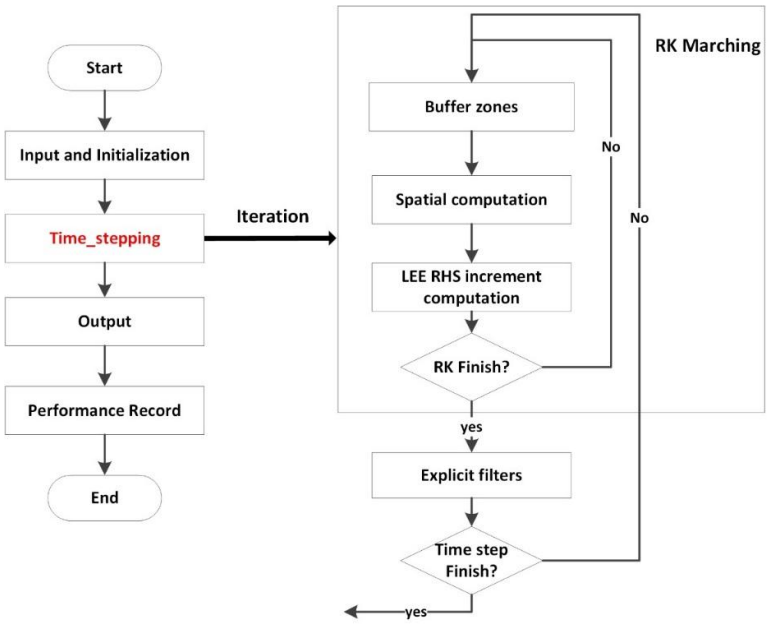


Figure 3.1 Flowchart of BenLEE.

BenLEE runs in serial on one CPU core. It first loads parameters from an input file, performs initialization on variables, and then loops iterations until the total time-step is reached. Lastly, it outputs results and prints wall-clock time. The main difference in *Time-stepping* between BenLEE and SotonLEE is relevant index transformation to multi-block mesh. In BenLEE, only one mesh block is allowed and there are no inviscid wall or symmetry conditions. Since the index transformation is not computationally intensive, ignorance of index transformation does not affect the profile of the solver.

To build BenLEE correctly, a 2D acoustic wave propagation problem [146] is selected as the benchmark problem. The physical domain is a rectangle with a dimension of [-50, 50] × [-50, 50] with outflow buffer zones around it. The width of each buffer zone is 20. The whole physical domain is shown in Figure 3.2 below:

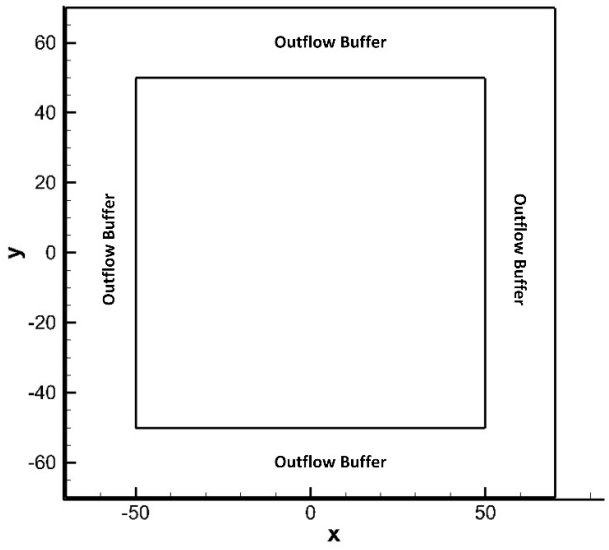


Figure 3.2 Physical domain in 2D propagation case.

The mesh is uniform. The LEE is non-dimensional with the following reference scales:

Length scale = Δx (grid spacing)

Velocity scale = c_∞ (ambient acoustic speed)

Time scale = $\Delta x / c_\infty$

Density scale = ρ_∞ (ambient density)

Pressure scale = $\rho_\infty c_\infty^2$

The 2D non-dimensional LEE on a uniform mean flow field is expressed as:

$$\frac{\partial \rho}{\partial t} + c_x \frac{\partial \rho}{\partial x} + \frac{\partial u}{\partial x} + c_y \frac{\partial \rho}{\partial y} + \frac{\partial v}{\partial y} = 0 \tag{2.1}$$

$$\frac{\partial u}{\partial t} + c_x \frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} + c_y \frac{\partial u}{\partial y} = 0 \tag{2.2}$$

$$\frac{\partial v}{\partial t} + c_x \frac{\partial v}{\partial x} + c_y \frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = 0 \tag{2.3}$$

$$\frac{\partial p}{\partial t} + c_x \frac{\partial p}{\partial x} + \frac{\partial u}{\partial x} + c_y \frac{\partial p}{\partial y} + \frac{\partial v}{\partial y} = 0 \tag{2.4}$$

in which, ρ, u, v and p are acoustic variables whose prime symbols are removed for simplicity. The Mach number of mean flow in the x direction, c_x is 0.5, whereas c_y is zero in the y direction. The acoustic field is initialized by a Gaussian pulse across the whole domain:

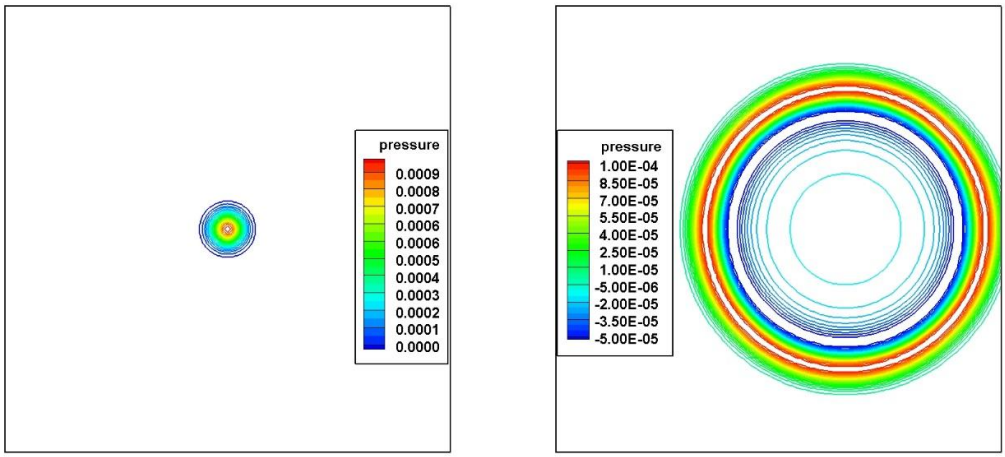
$$\rho = \varepsilon \exp\left[-(\ln 2)\times\left(x^2+y^2\right)/d^2\right] \tag{2.5}$$

$$p = \varepsilon \exp\left[-(\ln 2)\times\left(x^2+y^2\right)/d^2\right] \tag{2.6}$$

$$u=0 \tag{2.7}$$

$$v=0 \tag{2.8}$$

where ε denotes the acoustic amplitude, whereas d represents the half width of Gaussian pulse. Here ε is 0.001 and d is 3.0. Time step size Δt is 0.5. The pressure contours are shown below:



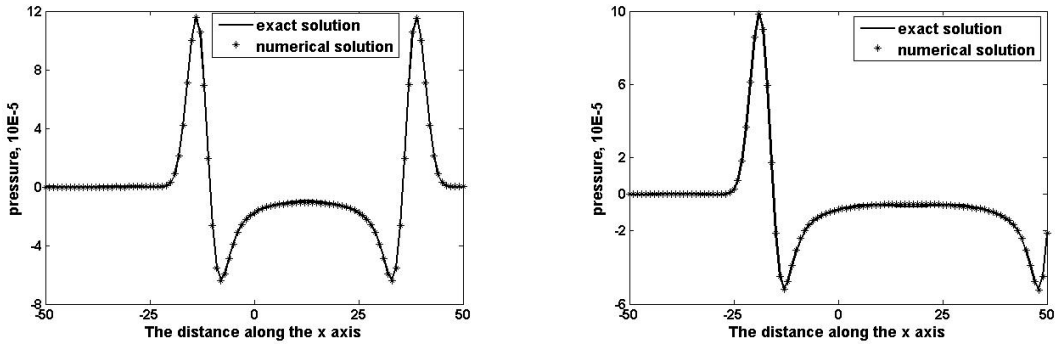
a) t = 0. b) t = 30, the 60th time step.

Figure 3.3 Acoustic pressure contours, c_x = 0.5.

As shown in Figure 3.3, BenLEE holds the waveform when an initial acoustic perturbation is propagating. The exact solution is given to evaluate the quality of the solver below:

$$p(x,y,t) = \rho(x,y,t) = \varepsilon_1 / 2\alpha_1 \times \int_0^\infty e^{-\xi/4\alpha_1} \times \cos(\xi t) J_0(\xi\eta)\xi d\xi \tag{2.9}$$

in which $\alpha_1 = \ln 2 / d^2$, $\eta = [(x - Mt)^2 + y^2]^{1/2}$ and J_0 is the 0th order Bessel function of the first kind. The acoustic pressure along the line at $y = 0$ is recorded in the numerical solution. A comparison between the numerical solution and exact solution is shown in Figure 3.4.



a) t = 25, the 50th time step. b) t = 35, the 70th time step.

Figure 3.4 Pressure waveforms along y = 0, c_x = 0.5.

Good agreement is found between the analytical solution and numerical result from Figure 3.4. In 3D computations, the physical domain is constructed by duplications of the 2D domain in the *z* direction. The buffer zone is also incorporated in the *z* direction. The simple 2D acoustic wave propagation demonstrates the correctness and fidelity of BenLEE. It will be used to develop the bidiagonal matrix solver on a GPU in subsequent sections.

3.2 Bidiagonal Matrix Solver on a GPU

The prefactored compact scheme gives rise to an upper bidiagonal matrix and a lower bidiagonal one. The bidiagonal matrix is a recursive system which globally depends on all the elements in the matrix. Therefore, it is not easy to find an efficient parallel algorithm for solving a bidiagonal matrix. As mentioned in Section 1.2.3.5, solving a tridiagonal matrix on a GPU has been investigated and good performance can be achieved. In this section, effort is extended to implementing a bidiagonal matrix on a GPU. Based on different mathematical forms, four methods are employed to solve a bidiagonal matrix. Each method is assigned a name for shorthand notation.

3.2.1 Natural Method

Natural method refers to the back substitution algorithm and is based on a recursive system form. It can be observed from Eq. (2.7) that for forward derivative $D^F_{i,j,k}$ to be solved, $D^F_{i+1,j,k}$ has to be known first. Also, in Eq. (2.8) $D^B_{i-1,j,k}$ has to be known first for $D^B_{i,j,k}$ to be solved. Therefore, data dependency occurs only in the *x* direction when the derivatives in the *x* direction are solved. In a 3D computational domain, systems like Eqs. (2.7) and (2.8) with different *j* or *k* indices can be solved independently. Eq. (2.7) can be solved by back substitution in a descending order of *i* from *Nx* - 1 to 2, whereas Eq. (2.8) is solved in an ascending order of *i* from 2 to *Nx* - 1. A CUDA thread is responsible

for solving a bidiagonal matrix, which makes the method 'embarrassingly parallel'. In parallel computing, embarrassingly parallel means that the problem is naturally parallel or little effort is required to partition the problem into smaller parallel tasks. This is a direct porting of the original algorithm on the GPU, thus it is called the Natural method. The Natural methods in the *y* and *z* direction are implemented in similar ways.

Since each CUDA thread is responsible for solving a recursive system, data dependency only occurs in a thread and there is no data dependency between threads. No data reuse occurs between recursive systems. Therefore, no shared memory is utilized in the implementation of the Natural method.

3.2.2 Parallel Cyclic Reduction Method

The PCR method was popular in the 1980s [147-149] and research has shown it to be one of the fastest generic algorithms for a tridiagonal system on a GPU [85, 87, 143, 150]. The PCR method is applied to a bidiagonal system in this study. The core of the PCR method is the application of Gaussian elimination in parallel with variant elimination strides. First, the PCR method on a simple lower bidiagonal system is demonstrated. Then, the procedure is summarized.

If **A** is a ninth-rank square matrix as given below in the matrix form:

$$\mathbf{A}=\begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{Bmatrix} \tag{2.10}$$

the last row is unrelated to other rows and is thus removed together with the last column. The matrix **A** can be expressed as:

$$\mathbf{A}=\begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 1 \end{Bmatrix} \tag{2.11}$$

At the first step, all the lower bidiagonal elements are eliminated by the diagonal elements in the row above. The stride between two rows where the elimination operation is performed is 1. It gives rise to a new matrix:

$$\begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.01 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.01 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.01 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.01 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.01 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.01 & 0 & 1 \end{Bmatrix} \tag{2.12}$$

At the second step, the row stride is set to 2. The lower diagonal elements in the new matrix are eliminated by the diagonal elements which are 2 rows above in a similar way as in step one:

$$\begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -0.0001 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -0.0001 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -0.0001 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -0.0001 & 0 & 0 & 0 & 1 \end{Bmatrix} \tag{2.13}$$

At the third step, the row stride is set to 4. After elimination, the matrix becomes an identity matrix:

$$\begin{Bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{Bmatrix} \tag{2.14}$$

When the matrix is manipulated, the RHS vectors experience the same operation. In the code, for less computational operations, the diagonal elements are normalized at the initialization phase so that the RHS vectors are directly the solution when the elimination finishes. In real practice, the matrix rank N is rarely a power of two. Given a lower system with N elements in which $2^M < N \leqslant 2^{M+1}$, the bidiagonal matrix can be expressed as a recursive system in the form:

$$r_i^h D_{i-2^h} + D_i = g_i^h \tag{2.15}$$

in which, *i* denotes the element index, whereas *h* represents the elimination step varying from 0 to *M* + 1. $D_{i-2^h} = 0$ when $i - 2^h \le 0$.

In the PCR kernel execution configuration, each recursive system is put into a CUDA block. *Nx* threads are launched in a block and each thread is responsible for solving Eq. (3.15) with a specific '*i*'. There are *Ny* blocks in the *y* direction and *Nz* blocks in the *z* direction. The implementation procedure of the PCR method in the *x* direction for a bidiagonal matrix on a GPU can be summarized below:

1) At the initialization phase, the primitive variables are loaded from global memory to the shared memory. The LHS coefficient r_i^0 is initialized by the variable in the constant memory of the GPU whereas the RHS vector g_i^0 is initialized in the shared memory.

$$g_i^0 = \frac{1}{\beta \times \Delta x}\left[a_B f_{i+1} + b_B f_i + c_B f_{i-1}\right] \tag{2.16}$$

$$r_i^0 = \alpha / \beta \tag{2.17}$$

2) The LHS coefficient r_i^h and the RHS vector g_i^h are computed in the shared memory and a local synchronization barrier is made to ensure all the threads in a CUDA block have finished the computation.

$$r_i^h = r_i^{h-1} \times \left(-r_{i-2^h}^{h-1}\right) \tag{2.18}$$

$$g_i^h = g_i^{h-1} + g_{i-2^h}^{h-1} \times \left(-r_{i-2^h}^{h-1}\right) \tag{2.19}$$

3) If *h = M + 1*, then $D_i = g_i^{M+1}$ is the solution and is written back to the global memory; otherwise, phase two must be repeated.

When a computational domain is given with *Nx* × *Ny* × *Nz* elements, *Nx* × *Ny* × *Nz* threads are launched to solve the domain simultaneously. *Ny* × *Nz* thread blocks are launched, each block with *Nx* threads solving a recursive system concurrently in the *x* direction. The recursive systems in the *y* and *z* directions are solved in similar ways. The implementation leads to highly efficient operations in the *x* direction. However, when the PCR is applied in the *y* direction, primitive variables have to be loaded from the global memory to the shared memory at the initialization phase. The memory stride between two neighbouring threads is a multiple of *Nx* which causes severe redundant memory access. This introduces a performance penalty in the *y* direction. The similar performance penalty occurs when PCR is implemented in the *z* direction.

3.2.3 MatMul Method

The term MatMul stands for matrix multiplication which is the core of the MatMul method. Tutkun and Edis [90] applied this method to Lele's compact scheme [12], which gave rise to a tridiagonal system. Following Tutkun and Edis's work, the MatMul method is optimized and applied to a bidiagonal system. It can be observed from Eqs. (2.15) and (2.19) that matrices $\mathbf{d}^F$ and $\mathbf{d}^B$ can be obtained in a straightforward way by inversion of the coefficient matrices $\mathbf{A}^F$ and $\mathbf{A}^B$. Tutkun and Edis's work can be formulated in equations below:

$$\mathbf{d} = inv(\mathbf{A}) \times \mathbf{b} = inv(\mathbf{A}) \times (\mathbf{B} \times \mathbf{f}) \tag{2.20}$$

in which **d** is the derivative matrix, **A** is a tridiagonal matrix, and **b** is a RHS vector. **B** is a sparse matrix, whereas **f** is a known primitive field variable vector. By optimization, the two multiplication operations can be aggregated into one shown in Eq. (3.21) below:

$$\mathbf{d} = (inv(\mathbf{A}) \times \mathbf{B}) \times \mathbf{f} = \mathbf{C} \times \mathbf{f} \tag{2.21}$$

For prefactored compact schemes, it can be represented as:

$$\mathbf{d}^F = inv(\mathbf{A}^F) \times \mathbf{B}^F \times \mathbf{f} = \mathbf{C}^F \times \mathbf{f} \tag{2.22}$$

$$\mathbf{d}^B = inv(\mathbf{A}^B) \times \mathbf{B}^B \times \mathbf{f} = \mathbf{C}^B \times \mathbf{f} \tag{2.23}$$

$$\mathbf{d} = (\mathbf{d}^F + \mathbf{d}^B) = \mathbf{C}^F \times \mathbf{f} + \mathbf{C}^B \times \mathbf{f} = \mathbf{C} \times \mathbf{f} \tag{2.24}$$

The matrix **C** is a dense matrix. It does not change during iterations and can be obtained by calling the LAPACK library at the initialization phase. The derivative vector **d** is solved by a simple matrix multiplication in Eq. (3.24). Since for all vectors **f** with different *j* or *k*, coefficient matrix **C** stays the same, the whole 3D domain can be computed by a simple matrix-matrix multiplication.

$$\mathbf{D} = \left[\mathbf{d}_{1,1}, \quad \cdots, \quad \mathbf{d}_{j,k}, \quad \cdots, \quad \mathbf{d}_{Ny,Nz}\right] = \mathbf{C} \times \left[\mathbf{f}_{1,1}, \quad \cdots, \quad \mathbf{f}_{j,k}, \quad \cdots, \quad \mathbf{f}_{Ny,Nz}\right] = \mathbf{C} \times \mathbf{F} \tag{2.25}$$

in which, **F** is a 3D matrix composed of primitive variables. The matrix multiplication solver on the GPU comes from the CUDA FORTRAN Programming Guide and Reference [69] and is accommodated to 3D matrix multiplication. It makes use of shared memory to reduce the access to global memory therefore is a highly efficient matrix solver.

In Eq. (3.25), each element in matrix **D** is a product of a row of **C** and a column of **F**. An illustration of the implementation of the matrix-matrix multiplication on a GPU is shown below:

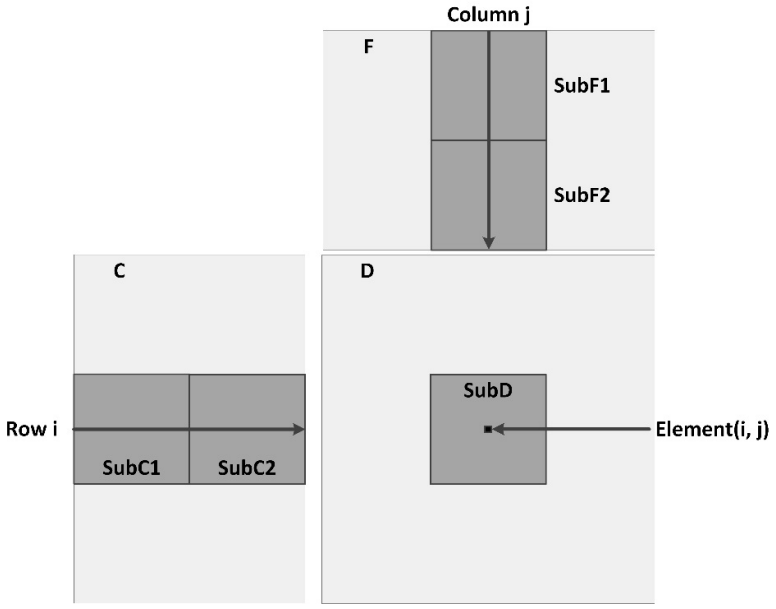


Figure 3.5 Matrix multiplication by accumulating product of sub-matrices.

The whole matrix D can be categorized into a number of sub-matrices with uniform size. The tasks of computation of sub-matrices are performed independently in parallel. When a sub-matrix in **D** is computed, it can be decomposed into the following form:

$$\mathbf{D}_{sub} = \mathbf{C}_{sub} \times \mathbf{F}_{sub} = \mathbf{C}_{sub1} \times \mathbf{F}_{sub1} + \mathbf{C}_{sub2} \times \mathbf{F}_{sub2} \tag{2.26}$$

in which, $\mathbf{D}_{sub}$, $\mathbf{C}_{sub}$ and $\mathbf{F}_{sub}$ are sub-matrices of **D**, **C** and **F** respectively. Given a $\mathbf{D}_{sub}$ with dimensions of (*B*, *B*), then a 2D thread block containing B^2 threads is launched. Each thread has unique index *(ti, tj)* corresponding to the element index *(i, j)*. The element of *D(i,j)* is computed by the product of row *i* in matrix **C** and column *j* in matrix **F**:

$$D(i,j) = \sum_{k=1}^{Nx} C(i,k) \bullet F(k,j) = \sum C_{sub1}(i,k) \bullet F_{sub1}(k,j) + \sum C_{sub2}(i,k) \bullet F_{sub2}(k,j) \tag{2.27}$$

The computation of a sub-matrix in **D** can be summarized into three stages below:

1) At the first stage, a thread with index *(i, j)* loads an element *(i, tj)* in row *i* of a sub-matrix in **C** (like $\mathbf{C}_{sub1}$) and an element *(ti, j)* in column *j* of a sub-matrix in **F** (like $\mathbf{F}_{sub1}$) from global memory to the fast shared memory. Then, a synchronization in a CUDA block is performed to ensure that all the load operations complete.
2) The matrix-matrix multiplication is performed in Eq. (3.27) in the current sub-matrices. The current product is accumulated to the product from previous matrix multiplication (if available).

3) A local synchronization is performed to ensure all the multiplications and summations complete. If $\mathbf{C}_{sub}$ and $\mathbf{F}_{sub}$ are the last sub-matrices in **C** and **F**, then $\mathbf{D}_{sub}$ is the solution, and the results are written back into global memory. Otherwise, the next sub-matrices in **C** and **F** are switched to by updating $tj = tj + B$ and $ti = ti + B$ and repeating stage 1.

The matrix multiplications in the *y* and *z* directions are similar. The difference is that the RHS matrix **F** should be composed of vectors **f** in the *y* direction and *z* direction, respectively. In addition, it requires an extension of the 2D thread index to 3D one when a 3D computation is performed. These two changes do not impose significant modification upon the algorithm mentioned above, so they are not described in detail.

In addition, it is noted that the MatMul method in fact is a general method for solving a linear system, described by Eq. (3.25). It is reasonable to conclude that some higher order compact schemes [12, 39] which give rise to penta-diagonal matrices can obtain a higher speed-up via the MatMul method.

3.2.4 Hybrid Method

Implementations of the Natural method, the PCR method and the MatMul method on a typical 2D mesh and 3D mesh are profiled. The results are shown below in Table 3-1.

Table 3-1 Wall-block time taken in Natural, PCR and MatMul implementations.

Kernel names	2D			3D		
	Natural	PCR	MatMul	Natural	PCR	MatMul
Prefactored scheme in the *x* direction	76.4%	26.9%	37.1%	72.8%	19.6%	21.5%
Prefactored scheme in the *y* direction	20.0%	37.4%	41.5%	9.5%	25.4%	23.8%
Prefactored scheme in the *z* direction	Null	Null	Null	8.8%	25.1%	27.0%
Other kernels	1.1%	17.9%	11.6%	5.8%	17.8%	21%
Buffer zone computation	0.2%	3.0%	2.5%	0.7%	2.9%	3.5%
Memory set	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

Note that the 2D implementations are run on a mesh with a dimension of 128 × 128, whereas the 3D implementations are on a mesh with a dimension of 64 × 64 × 64.

In Table 3-1, it is found that a considerable performance gap exists between the Natural method in the *x*, *y* and *z* directions. The computational cost in the *x* direction dominates the overall performance of the applications. The computational cost in the *x* direction in the Natural method is roughly 8 times of that in the *y* and *z* direction in 3D computations whereas it is nearly 4 times in 2D computations. The main cause of low performance of the Natural method in the *x* direction is

attributed to the redundant access to the global memory which will be explained in Section 3.4. The only difference in the Natural method between different directions is the memory access pattern since the mesh size and the arithmetic operations are the same in both/all directions. In the *x* direction, the memory stride between two neighbouring threads is *Nx*, whereas that is 1 in the *y* and the *z* direction. In contrast, the PCR and MatMul methods achieve high efficiency in the *x* direction. Thus, the application of the PCR method or the MatMul method in the *x* direction only, and preservation of the Natural method in the *y* and the *z* direction should result in improved performance. This hybrid method is a mixed-grained parallel approach where the degree of parallelism is fine in the *x* direction but coarse in the *y* and *z* direction.

3.3 Performance

BenLEE contains three categories of subroutines in terms of memory access pattern, namely the implicit stencil type, explicit stencil type and point-wise type. The implicit type refers to the subroutine which performs prefactored compact scheme computations, whereas the explicit type denotes the subroutine performs explicit finite difference scheme computations. The point-wise type of subroutines contains the buffer zone condition, and time-marching in governing equations. It can be observed from Table 3-1 that the implicit stencil type of subroutines dominate the overall performance. The explicit stencil type and point-wise type do not contribute considerably. Furthermore, the algorithms on explicit stencil type of subroutines and point-wise type of subroutines are the same when the implementations of the implicit stencil vary. Therefore, the performance difference between different implementations is largely caused by the implicit stencil computations. The strategy on explicit stencil type and point-wise type of subroutines will be explained in chapter 4.

The performance is obtained by comparing the wall-clock time on a Tesla M2050 GPU with that on a single Intel Xeon E5620 CPU core with a frequency of 2.40 GHz. The compiler used is PGI 13.3 with –fast –Munroll –Mcuda = cc20 flags for both CPU and GPU implementations, in which the compiler flags denote optimization of codes in terms of the fastest implementation speed, unrolling loops and current GPUs with CUDA computing capability of 2.0, respectively. The implementation on the CPU is the back substitution algorithm, whereas the implementations on a GPU utilize Natural, PCR, MatMul and hybrid methods respectively. The wall-clock time and speed-ups of BenLEE implementing different numerical methods on prefactored compact scheme on a GPU are shown in tables below, and a summary of the speed-up is shown in Fig. 3.6 below. It is emphasized here that the wall-clock time and speed-ups are based on the total simulation time in which the I/O time is not included. This comparison also applies to SotonLEE. Table 3-2Table 3-3 summarize the performance of BenLEE on a GPU in 2D computations, while the performance in 3D computation is

tabulated in Table 3-4– Table 3-5. For 3D case studies in BenLEE, the maximum domain size on a Tesla M2050 GPU is 256 × 128 × 128, restricted by the global memory volume capacity of the GPU.

Table 3-2 Wall-clock time and speed-ups of BenLEE (2D) with Natural, PCR and MatMul methods.

Mesh Size	Serial	Natural		PCR		MatMul	
	Time	Time	Speed-up	Time	Speed-up	Time	Speed-up
32×32	0.06	0.50	0.1	0.05	1.2	0.02	3.2
64×64	0.30	1.12	0.3	0.07	4.5	0.04	7.2
128×128	2.29	2.91	0.8	0.19	12.0	0.22	10.4
256×256	9.50	5.91	1.6	0.60	15.9	1.13	8.4
512×512	82.72	16.83	4.9	2.63	31.5	8.05	10.3

Note that 'Mesh Size' refers to the amount of mesh points. 'Serial' denotes the serial implementation of BenLEE on a single CPU core, whereas 'Natural', 'PCR' and 'MatMul' represent BenLEE implemented by the Natural, PCR and Matmul method on a GPU, respectively. The total number of iterations in 2D computation is 90. The unit of time is second (s).

Table 3-3 Wall-clock time and speed-ups of BenLEE (2D) with Hybrid methods.

Mesh Size	Serial	PCR_Hybrid		MatMul_Hybrid	
	Time	Time	Speed-up	Time	Speed-up
32×32	0.06	0.13	0.5	0.11	0.5
64×64	0.30	0.24	1.2	0.23	1.3
128×128	2.29	0.65	3.5	0.67	3.4
256×256	9.50	1.49	6.4	1.75	5.4
512×512	82.72	3.81	21.7	6.38	13.0

Note that 'PCR_Hybrid' denotes the hybrid method which employs PCR in the *x* direction whereas 'MatMul_Hybrid' represents the hybrid method with MatMul in the *x* direction.

Table 3-4 Wall-clock time and speed-ups of BenLEE (3D) with Natural, PCR and MatMul methods.

Mesh Size	Serial	Natural		PCR		MatMul	
	Time	Time	Speed-up	Time	Speed-up	Time	Speed-up
32×32×32	5.0	1.0	4.9	0.3	16.0	0.1	50.2
64×64×64	46.1	7.2	6.4	1.6	28.2	1.1	41.4
128×128×128	526.5	57.9	9.1	13.3	39.8	14.9	35.4
256×128×128	1158.2	114.4	10.1	29.4	39.4	38.5	30.1
128×256×128	1159.0	106.5	10.9	28.2	41.1	39.8	29.2
128×128×256	1186.4	107.3	11.1	28.4	41.8	39.2	30.3

Table 3-5 Wall-clock time and speed-ups of BenLEE (3D) with Hybrid methods.

Mesh Size	Serial	PCR_Hybrid		MatMul_Hybrid	
	Time	Time	Speed-up	Time	Speed-up
32×32×32	5.0	0.3	15.7	0.2	20.6
64×64×64	46.1	1.4	33.0	1.3	35.1
128×128×128	526.5	11.0	47.9	12.3	42.7
256×128×128	1158.2	22.2	52.2	30.9	37.5

128×256×128	1159.0	21.8	53.1	24.5	47.4
128×128×256	1186.4	21.6	54.9	24.3	48.9

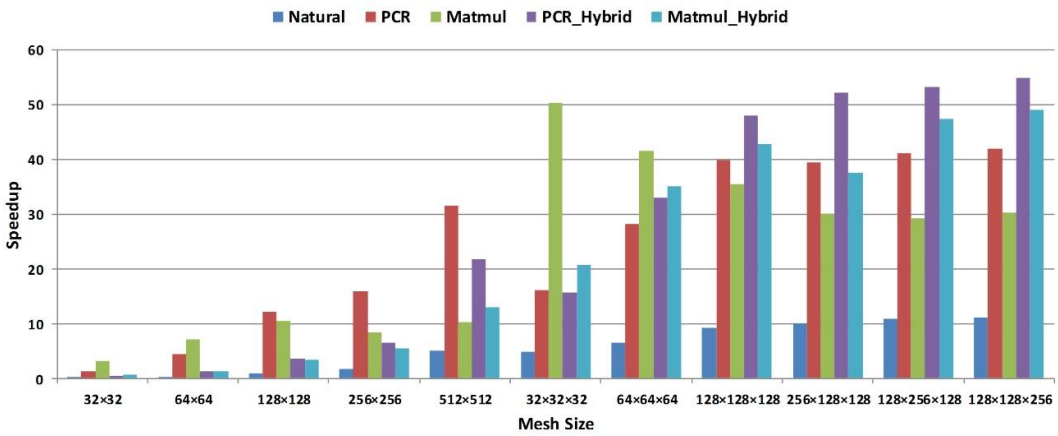


Figure 3.6 Summary of speed-ups of BenLEE on a GPU.

It can be observed from Figure 3.6 that all the mesh sizes are equal to powers of 2. However, in practical computations, the mesh size does not need to be a power of 2 because all the matrices will be padded congruently to be a multiple of 16 in the *x* direction both in the CPU and GPU codes. This is mainly used to enforce the memory to align a cache line of 128 bytes. The code is in double precision; therefore, the mesh size has to be a multiple of 16.

3.4 Analysis on Performance

The speed-up of the PCR method increases with the increase of the system size of the recursive system in Figure 3.6. This illustrates that the PCR method can achieve high performance on a large system. The PCR achieves the highest speed-up when the system dimension is greater than 128 in 2D computations whereas the MatMul is better for $N < 128$. In 3D computations, the MatMul achieves the highest speed-up when $N < 128$ and the speed-up drops as N increases. When $N > 128$, the performance of the MatMul is surpassed by the PCR and the PCR_Hybrid achieves the highest speed-up. For a detailed analysis of the performance, the wall-clock time spent in each direction is recorded when the system size is set to 128 × 128 × 128. The computational time spent in one direction is obtained by disabling computations in the other two directions. The result is shown in Table 3-6.

Table 3-6 The computational time spent in each direction.

Implementations	Wall-clock time (in seconds)		
	In the *x* direction	In the *y* direction	In the *z* direction
Natural	50.0	5.7	5.5
PCR	5.2	6.2	7.2
MatMul	5.9	6.3	6.8

It can be seen that the performance of the Natural method in the *x* direction is low; almost 1/10 of that in the *y* and the *z* direction. The system size and the arithmetic operations are isotropic in all directions. The only difference comes from the global memory access pattern in the different directions. The memory stride of neighbouring threads of the Natural method in the *x* direction is *Nx,* and each thread has to access global memory *Nx* times. This results in severely redundant memory access. The memory stride of the Natural method is 1 in the *y* and the *z* direction which results in coalesced memory access. Therefore, it is the anisotropic memory access pattern which gives rise to large anisotropy of the performance of the Natural method in the different directions.

Also, the redundant memory access and the coalesced memory access contribute to the big performance gap in the *x* direction among the Natural, PCR and MatMul methods. The memory stride of the PCR and MatMul is 1 which makes efficient coalesced memory access. The performance of the PCR and MatMul is nearly 10 times of that of Natural method in the *x* direction. In addition, it can be observed that the performance of the PCR method in the *y* and the *z* direction is lower than that in the *x* direction. This is attributed to the redundant memory access to the global memory during the initialization phase and the final phase of the PCR, since the memory strides are *Nx* in the *y* direction and *Nx* × *Ny* in the *z* direction. The cost of redundant global memory access of PCR in the *y* and the *z* direction is much lower than that of the Natural method in the *x* direction because each thread only needs to access the global memory once in the PCR method in the *y* and the *z* direction, whereas *Nx* times are necessary for the Natural method in the *x* direction. The 3D MatMul method achieves coalesced memory access in all three directions. The performance should be the same in all three directions. However, there is still a performance loss when the MatMul is applied from the *x* direction to the *y* and the *z* direction. The performance of the Natural method in the *y* and the *z* direction should be the same, as should the performance of the PCR in the *y* and the *z* direction. It is found that the cost is different when memory access occurs in different directions. The cost is always larger when the memory access to the *z* direction occurs than that when memory access to the *y* direction occurs.

In large scale 3D computations, the PCR and MatMul methods have been found to be efficient in the *x* direction. On the other hand, high efficiency is achieved by the Natural method in the *y* and the *z* direction. Three causes are found. First, the high efficiency of the Natural method in the *y* and

the *z* direction results from the coalesced memory access mentioned above. Second, for the PCR method, there is redundant memory access in the y and the z direction at the initial and final phases which partly offsets the performance. Third, the MatMul method introduces far more arithmetic operations than the PCR and the Natural methods if the system size is large. This trend is significant when the system dimension is larger than 128, as shown in Figure 3.6. It can be observed that the performance of the MatMul method decreases and that of the PCR increases with the increase of the mesh size monotonously. When the mesh size is larger than 128, the PCR is better than the MatMul method in 3D computations.

Therefore, the hybrid method is proposed, based on the anisotropic computational costs of the Natural, PCR and MatMul methods. It is a combination of the numerical methods in different directions. The hybrid method only achieves the best performance when the system size is larger than 128 in 3D computations, as shown in Figure 3.6.

Finally, the following strategy is formulated with the numerical methods: the PCR is always used for 2D computations; the MatMul method is used for 3D computations if the mesh dimensions *Nx*, *Ny*, and *Nz* are all smaller than 128; otherwise, the PCR_Hybrid is used. The strategy, together with the numerical methods, constitutes the solution to the bidiagonal matrix solver on a GPU.

3.5 Summary

In this chapter, a simple serial CAA scattering solver, BenLEE, has been developed as a workbench for the prefactored compact schemes which give rise to bidiagonal matrices. Three numerical methods, namely, the Natural method, the PCR method and the MatMul method, were derived for a bidiagonal matrix on a GPU. Then the performance and analysis followed. Furthermore, a hybrid method has been proposed to improve the overall computational performance in 3D computations. For 2D computations, the highest speed-up achieves about 30, whereas the highest speed-up is roughly 55 in 3D computations in double precision.

Chapter 4: Development of SotonLEE on GPUs

As mentioned in chapter 3, SotonLEE is a complex program which contains five types of subroutines in terms of memory access patterns during iterations. The five types of subroutines are: implicit stencil type, explicit stencil type, point-wise type, unstructured gather type and reduction type. The implicit type, which refers to the computation of prefactored compact schemes, has been investigated in detail in chapter 3. The following chapter is devoted to the parallel strategies of the four other types of subroutines on a GPU. Furthermore, the parallel strategy on multiple GPUs by using MPI + CUDA is also offered.

4.1 Explicit Stencil Type

The explicit type of subroutines refer to the computations of explicit filters. In SotonLEE, a tenth-order explicit filter is utilized to suppress the spurious waves. It is described below:

$$\begin{aligned}\overline{f}_i = f_i - c_5 \times \left(f_{i-5} + f_{i+5}\right) - c_4 \times \left(f_{i-4} + f_{i+4}\right) - c_3 \times \left(f_{i-3} + f_{i+3}\right) \\ - c_2 \times \left(f_{i-2} + f_{i+2}\right) - c_1 \times \left(f_{i-1} + f_{i+1}\right) - c_0 \times f_i\end{aligned} \tag{3.1}$$

in which $\overline{f}_i$ denotes the filtered field whereas f_i is the primitive field; c_i is the constant coefficient and Eq. (4.1) describes the filter in the inner points. The inner stencil is symmetric. When the points lie at $i < 6$ or $i > Nx - 4$, the boundary stencil has to be employed. The boundary stencils are asymmetric and utilize different formulations at varying positions. For points at $i < 6$, the filtering operations are shown below:

$$\overline{f}_1 = f_1 - \sum_{j=1}^{6} c_{1,j} \times f_j \tag{3.2}$$

$$\overline{f}_2 = f_2 - \sum_{j=1}^{7} c_{2,j} \times f_j \tag{3.3}$$

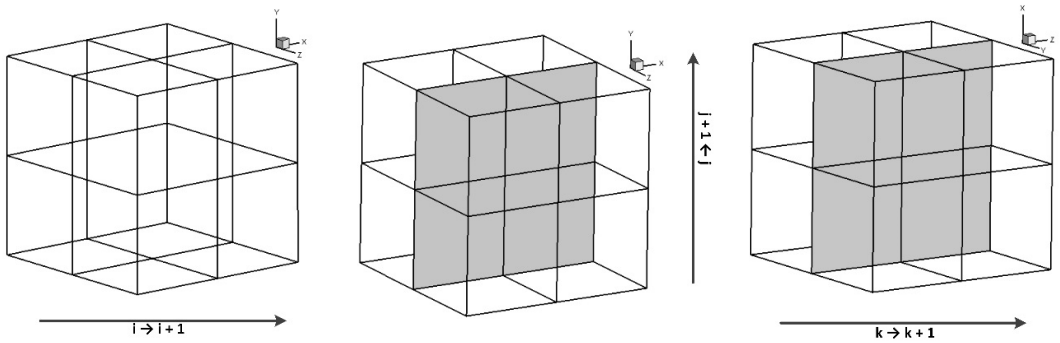
$$\overline{f}_3 = f_3 - \sum_{j=1}^{8} c_{3,j} \times f_j \tag{3.4}$$

$$\overline{f}_4 = f_4 - \sum_{j=1}^{9} c_{4,j} \times f_j \tag{3.5}$$

$$\overline{f}_5 = f_5 - \sum_{j=1}^{10} c_{5,j} \times f_j \tag{3.6}$$

The coefficients $c_{1,j}$ to $c_{5,j}$ denote the coefficients on the boundary points. The similar stencils apply to the mesh points with $i > Nx - 4$. The computation in the *y* or the *z* direction occurs in a similar way with varying index *j* or *k*.

As mentioned in section 1.2.3.4, the maximum performance of explicit stencil computation can be achieved by the 'slicing method' or can be exploited by the 'tiling method' [25]. These two methods load a piece of global memory into the fast shared memory and perform the explicit stencil computations in *x*, *y* and *z* directions simultaneously. The two methods mitigate the global memory access and thereby obtain high performance. However, in SotonLEE, the filtering in the *x*, *y* or *z* direction is performed alternately at each time-step. Therefore, the slicing method which is efficient in 3D stencil computation simultaneously does not fit well to the explicit filtering in SotonLEE. Furthermore, it can be observed that the stencil size is large from Eq. (4.1). So it is more efficient to use the 2D tiling method to reduce the global memory access due to the limited volume of shared memory. The 2D tiling method in the explicit filters is shown in Figure 4.1 below:



a) In the *x* direction. b) In the *y* direction. c) In the *z* direction.

Figure 4.1 The anisotropic 2D tiling method in explicit filters.

As shown in Figure 4.1, the 2D tiling method is anisotropic in the *x*, *y* and *z* directions. When the explicit filtering operation is performed in the *x* direction, the 2D tiling method in the *x* direction is used. The threads in the 2D tiling method in the *x* direction are organized into many lines. Given a 3D domain with dimensions of (*Nx*, *Ny*, *Nz*) and a CUDA block with a dimension of *Bx*, there are (*Nx/Bx)* × *Ny* × *Nz* thread lines, each of which contains *Bx* threads. The memory stride between two neighbouring threads is 1; therefore, the access to the global memory is coalesced and high performance is achieved.

The 2D tiling method in the *y* direction is a 2D rectangle in the *xy* plane. There are (*Nx/Bx*) × (*Ny/By*) × *Nz* 2D rectangles in total, each of which contains $\underline{Bx}$ × *By* × 1 threads. The filtering operation is in the *y* direction. The *Bx* threads in the *x* direction are mainly used to reduce the redundant memory access. Each time an element in the *y* direction is required, a continuous global memory segment

is loaded. The continuous memory segment by each load from global memory is not abandoned but is shared in the *x* direction because the memory stride between two neighbouring threads is 1. Therefore, the redundant memory access is reduced and high performance can be achieved in the *y* direction.

The 2D tiling method in the *z* direction is similar to that in the *y* direction. The threads are organized into a 2D rectangle in the *xz* plane. There are (*Nx/Bx)* × *Ny* × (*Nz/Bz)* rectangles if the dimensions of a CUDA block are *Bx* × 1 × *Bz*. Also, the *Bx* threads in the *x* direction are largely used to eliminate the redundant memory access induced by the global memory access in the *z* direction. The memory stride between two neighbouring threads is 1 and high performance is achieved in the *z* direction.

Another important feature of the explicit filter is that the stencil operations vary across many points close to the boundary due to the large stencil size, as shown in Eqs. (4.2) – (4.6). Therefore, the filtering operation on boundary stencils gives rise to branching of execution which reduces the performance on a GPU since the threads are implemented in a bunch if they have the same instructions on a GPU. For the highest performance, a unified expression is the best choice. Eqs. (4.2) – (4.6) can be aggregated into one equations:

$$\overline{f}_i = f_i - \sum_{j=1}^{j=10} c_{i,j} \times f_j \tag{3.7}$$

$$c_{i,j} = 0 \text{ if } j > i+5$$

$c_{i,j}$ is organized into a matrix **C** shown below:

$$\mathbf{C} = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & c_{1,6} & 0 & 0 & 0 & 0 \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} & c_{2,6} & c_{2,7} & 0 & 0 & 0 \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} & c_{3,6} & c_{3,7} & c_{3,8} & 0 & 0 \\ c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} & c_{4,6} & c_{4,7} & c_{4,8} & c_{4,9} & 0 \\ c_{5,1} & c_{5,2} & c_{5,3} & c_{5,4} & c_{5,5} & c_{5,6} & c_{5,7} & c_{5,8} & c_{5,9} & c_{5,10} \end{bmatrix} \tag{3.8}$$

By using Eqs. (4.7) and (4.8), only two branches are present in the filtering operations, namely inner and boundary filtering operations. The boundary stencil accesses the corresponding row of matrix **C** according to its index *i*. The coefficient matrix is stored in constant memory with low access latency on a GPU.

When the explicit filter is implemented, the primitive variables are first loaded into shared memory. Then, the computation is performed in terms of the index. When the computation completes, the result is written back to the global memory.

4.2 Point-wise Type

The point-wise-type subroutines can be expressed in the following form:

$$A[i][j][k] = B[i][j][k] + C[i][j][k] \tag{3.9}$$

in which each element only depends on its local information. There is no data dependency between neighbouring elements. Subroutines of this type contain the computation of RHS residual of governing equations and buffer zone conditions. These subroutines are embarrassingly parallel; therefore, no shared memory is used. No particular attention is paid to the implementation on a GPU. The 2D tiling method is used to eliminate the redundant memory access.

4.3 Unstructured Gather Type

The unstructured gather-type subroutines contain the index transformation between two neighbouring mesh blocks and the pack/unpack operations induced by multiple MPI processes. When a multi-block O-mesh is utilized, the index transformation between two neighbouring mesh blocks is necessary. The mesh elements close to the block interface in the neighbouring mesh block are mapped into ghost points of current mesh block by using index transformation. An increase in index *i* in current block may correspond to a decrease in *j* in the neighbouring block. Therefore, the global memory access pattern is irregular and much attention should be paid to the index calculation.

When multiple MPI processes are used, packing/unpacking operations occur in the vicinity of the mesh interface. The packing operation packs the information in the vicinity of current mesh block into a small buffer, whereas the unpacking operation unpacks the information in a small buffer into the ghost points. The packing and unpacking can be regarded as a transformation operation between two matrices with different dimensions that occurs in a similar way to the index transformation. They also perform irregular memory access to the global memory. No special algorithm is required to implement the unstructured gather type subroutines on a GPU, whereas the index calculation requires much attention.

4.4 Reduction Type

The reduction type subroutines perform a traversal on the physical domain and collect the maximum/minimum values on all the elements. On multiple GPUs, the reduction can be decomposed into three phases: first, reduction on a local CUDA block; second, reduction on a mesh block which always contains multiple CUDA blocks; lastly, reduction on the whole physical domain

which contains many mesh blocks. The reduction operations in the first and second phases are performed on GPUs, whereas the last phase is achieved by the MPI reduction operation [55].

Harris [151] performed an exhaustive research on the reduction operations on a GPU and a series of algorithms have been developed and verified. The kernel 3 with sequential addressing is selected in this research in phases one and two since other kernels, which are optimized further, unroll the loops, therefore imposing difficulty in code reading. Furthermore, the reduction operation does not contribute much to the overall performance. Therefore, kernel 3 is selected and the procedure is illustrated in Figure 4.2 below:

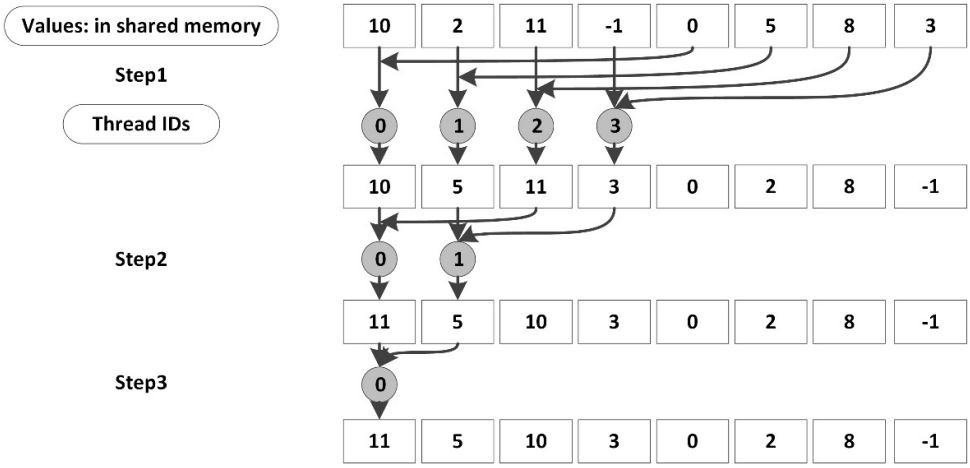


Figure 4.2 Reduction with sequential addressing.

In Figure 4.2, a system with eight elements is supposed. The data on which the reduction will be performed is first loaded from global memory into shared memory, or the data obtained from intensive computation has already resided in the shared memory. In the first step of reduction, a thread in the first half finds the local maximum value between itself and the corresponding thread that is half the total thread amount away. At the second step, a thread in the first quarter repeats the reduction in step one among the threads in the first half. At the last step, the global maximum value is found and is put in the first element. Supposing that there are $2^h < H \leqslant 2^{h+1}$ systems, each of which has a dimension of $2^M < N \leqslant 2^{M+1}$ and the data has been already in the shared memory, the reduction procedure in the whole physical domain can be concluded as below:

1) At the n^{th} step, the thread stride in a CUDA block is set to 2^{M+1-n}. A thread in the first half of 2^{M+1-n} finds the local maximum value between itself and the thread with the index 2^{M+1-n} far away, and keeps the local maximum value in its shared memory.

2) If the step *n* is less than *M + 1*, step 1 is repeated; otherwise, the procedure completes and the global maximum value is kept in thread 1. The result is written back to global memory. By now, the reduction on a CUDA block completes.
3) The same procedures in steps 1 and 2 are employed to obtain the maximum value across a mesh block. The data on which the reduction will be performed becomes an array, composed of the local maximum values obtained from step 2 across multiple CUDA blocks. Then, the global maximum value across a mesh block is found and copied back to RAM.
4) Lastly, an MPI reduction operation is utilized on the master process to collect the maximum value distributed in all the mesh blocks across the physical domain.

The operations in steps 1 and 2 find the local maximum value on a CUDA block. When operations in step 2 complete in all CUDA blocks, operations in step 3 are performed to get the maximum value in a mesh block. When the operations in step 3 in all mesh blocks complete, operations in step 4 collect the global maximum value in the whole domain. By using this method, the maximum value across the whole physical domain is found.

4.5 Minimization of Data Transfer

When multiple GPUs are utilized, there are data exchanges between GPUs. When data exchange occurs, data is copied through the PCI Express whose bandwidth (8 GB/s on PCI-express × 16 Gen2) [152] is far less than peak bandwidth of a GPU (144 GB/s on Tesla C2050) [153]. Consequently, the data transfer has to be avoided to the minimum extent. The flowchart of SotonLEE with an MPI + CUDA implementation with necessary data transfers is shown below:

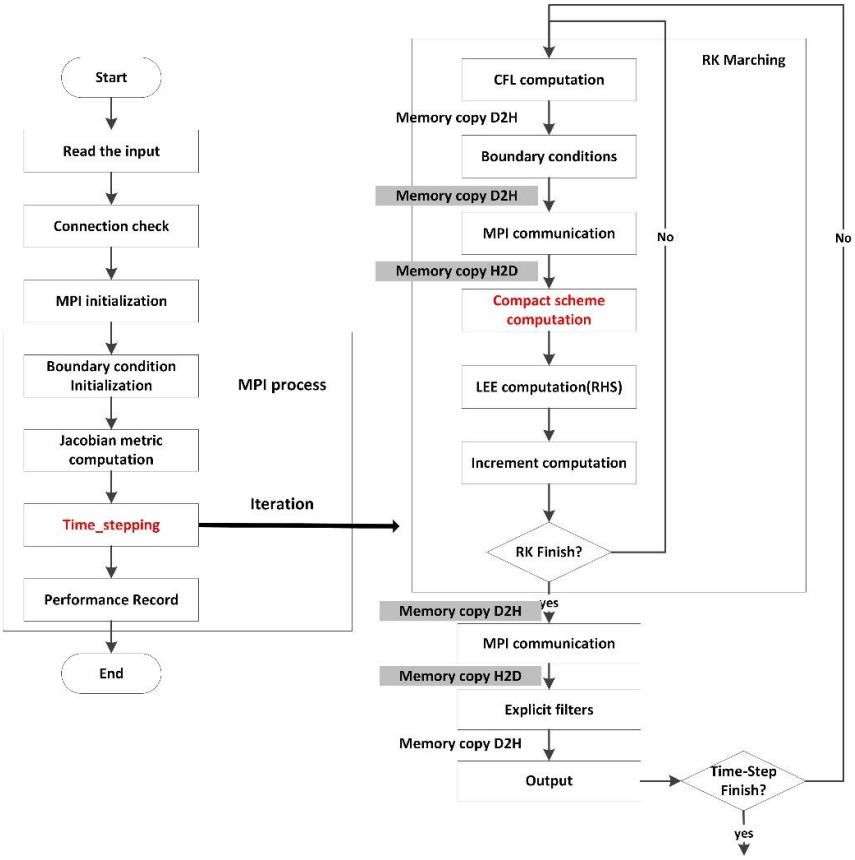


Figure 4.3 Flowchart of SotonLEE with MPI + CUDA implementation.

Note: H2D is the shorthand notation of host to device, while D2H denotes device to host. The memory copy in grey denotes the memory copies occurring in mesh block interface condition computations.

Normally, some data transfers are necessary for subroutines, such as the interface condition, and cannot be avoided. The data transfer occurs regularly in four scenarios: First, before the iteration in SotonLEE starts, data has to be copied from RAM to GPU memory and the data has to be written back from the GPU memory to RAM when the iteration completes. These data transfers occur only once outside the iteration. Therefore, they contribute little to the overall performance penalty for long-time iterations. Second, the data transfer occurs when both of the manipulations on interface condition occur and the two neighbouring mesh blocks are distributed across different CPU cores. The interface condition occurs in iterations and therefore contributes considerably to the overall performance penalty. When data transfer occurs, arrays have to be copied from the GPU memory to RAM, then copied to a network fabric buffer, and are finally sent by an MPI process. When the arrays from a neighbouring mesh block are received, the arrays have to be copied from the network fabric buffer to RAM, then copied to the GPU memory. The cost of this kind of communication is high. Third, the reduction operation requires a traversal across the physical domain therefore the

data transfer is also necessary. However, in SotonLEE, the CFL calculation is not performed as frequently as the interface condition occurs. Furthermore, the data transfer in the CFL calculation only involves a single variable rather than an array. Consequently, the CFL calculation does not contribute much to the overall performance penalty. Lastly, the output of transient acoustic field requires significant data transfers during iterations. The output always writes multiple arrays across the whole domain and occurs every iteration step in some cases, such as outputs of data to FW-H solver. Therefore, a substantial performance penalty occurs when the output in iteration occurs frequently. Other data transfers have to be eliminated.

In terms of contribution to the overall performance penalty, the data transfers outside the iteration contribute little and are necessary; therefore, they are ignored. Furthermore, the contribution of the CFL calculation within the iteration is small and is also ignored. In addition, the output operation does not perform any computational work and has to be performed from RAM to hard disk. This research focuses on the optimization of the data transfers in interface conditions denoted in grey in Figure 4.3. As shown in Figure 4.3, the data transfer has been minimized so that it only occurs at four scenarios mentioned above.

The memory transfers between multiple GPUs contain memory copy between system RAM and GPU memory through PCI express; memory copy between system buffer used for the GPU and the buffer for the Infiniband; and MPI send/receive operations. When the MPI + CUDA approach is utilized to manage multiple GPUs across multiple CPU cores, the data transfer procedure is performed below:

Table 4-1 Data transfer between two GPUs by MPI + CUDA approach.

```
MPI Rank 0:
istat = cudaMemcpy( hBuffer, dBuffer, bufferSize, cudaMemcpyDeviceToHost )
call mpi_iSend( hBuffer, bufferSize, mpi_double_precision, 1, tag, mpi_comm_world, sReq, mpiErr )
call mpi_wait(sReq, sStatus, mpiErr)
MPI Rank 1:
call mpi_iRecv( hBuffer, bufferSize, mpi_double_precision, 0, tag, mpi_comm_world, rReq, mpiErr)
istat = cudaMemcpy( dBuffer, hBuffer, bufferSize, cudaMemcpyDeviceToHost )
call mpi_wait(rReq, rStatus, mpiErr)
```

Note that hBuffer denotes a buffer in the system RAM whereas dBuffer represents a buffer on a GPU. A tag is used to uniquely identify a message. A send and a corresponding receive operation have to share the same tag. The data transfer can be diagrammed as shown in Figure 4.4:

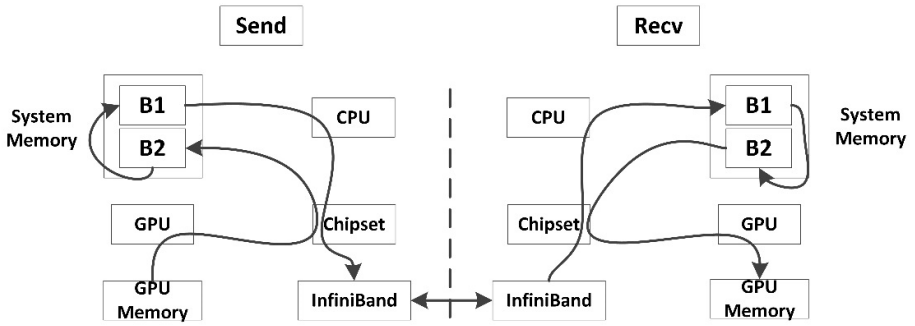


Figure 4.4 Data transfer by using MPI + CUDA approach. MPI rank 0 sends a message and MPI rank 1 receives a message. The boxes which are transmitted by the curves denote the units the data travels through when the memory copy operation occurs.

As shown in Figure 4.4, when a send operation occurs in MPI rank 0, there is a receive operation in MPI rank 1. In rank 0, the GPU buffer is first copied to a buffer B2 for the GPU in system RAM through PCI express. Then, the buffer B1 for the infiniband copies B2 and is sent by infiniband. In rank 1, once the buffer B1 for the infiniband receives a message, the buffer B2 for the GPU copies it and then copies it into the GPU memory through the PCI express. This kind of data exchange is somewhat complex and inefficient. It results from the private address space of each GPU and CPU. For instance, a variable on GPU 0 is not visible to GPU 1 and all the CPUs. With unified virtual addressing (UVA) supported by GPUDirect [154-156], the system memory and the memory of all GPUs in a node are combined into one addressing space. Therefore, the memory addresses are visible for CPUs and GPUs. In the MPI implementation with GPUDirect technology, the data transfer can be simplified. Furthermore, with GPUDirect remote direct memory access (RDMA), the procedure can be simplified further as:

Table 4-2 Data transfer between two GPUs via GPUDirect RDMA.

MPI Rank 0: call mpi_iSend(dBuffer, bufferSize, mpi_double_precision, 1, tag, mpi_comm_world, sReq, mpiErr) call mpi_wait(sReq, sStatus, mpiErr)
MPI Rank 1: call mpi_recv(dBuffer, bufferSize, mpi_double_precision, 0, tag, mpi_comm_world, rReq, mpiErr) call mpi_wait(rReq, rStatus, mpiErr)

With GPUDirect RDMA, the data transfer is simplified and the buffer on a GPU can be directly copied to infiniband. When the same data exchange occurs, the data exchange using the GPUDirect RDMA is diagrammed as shown in Figure 4.5 below:

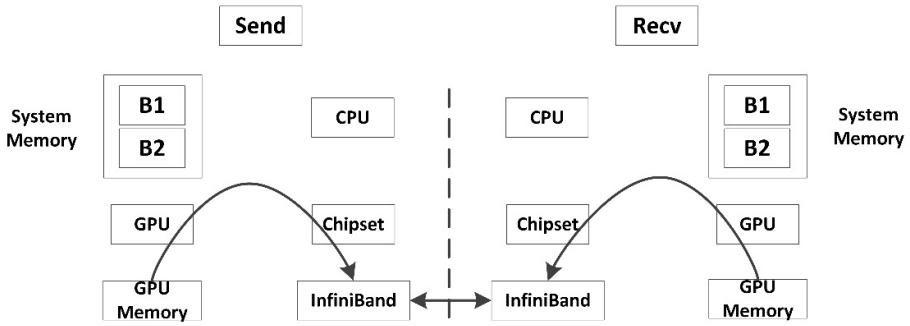


Figure 4.5 Data transfer by using GPUDirect RDMA. MPI rank 0 sends a message and rank 1 receives a message. The boxes which are transmitted by the curves denote the units the data travels through when the memory copy operation occurs.

In short, using GPUDirect RDMA gives rise to two advantages: first, it is much easier to program in comparison to the primitive MPI + CUDA; second, some acceleration techniques are used to improve the efficiency of data exchange [154-156]. Most of all, the GPUDirect RDMA is supported and available on IRIDIS4 and Emerald GPU cluster. Therefore, it is utilized to replace the previous MPI + CUDA communication in this research. The final flowchart is simplified and shown below:

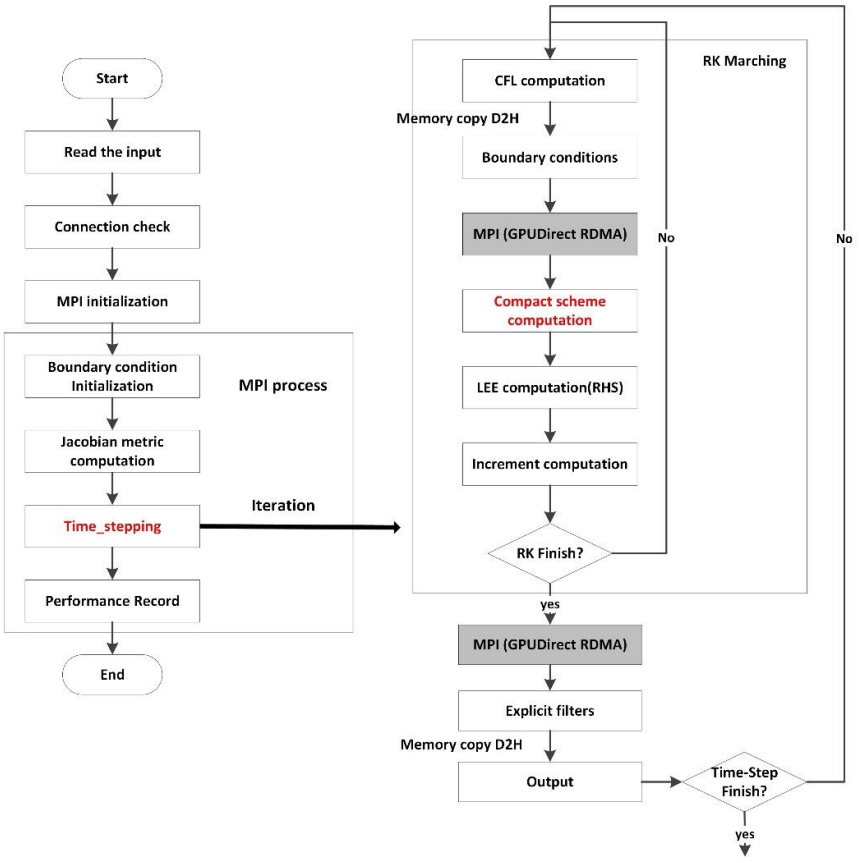


Figure 4.6 Flowchart of SotonLEE with GPUDirect RDMA implementation.

4.6 Performance

In this section, a number of benchmark cases which have successfully run with MPI and been validated in previous research, are reproduced to scale the performance of the new SotonLEE which runs on multiple GPUs across multiple nodes. Most of these cases were coded and used by various students at ANTC and included in the SotonLEE suite of codes. These benchmark cases contain three 2D/2.5D cases and two 3D cases. The 2D/2.5D cases include radiation of spinning modes out of an unflanged duct [34, 127, 157-159]; scattering of an initial acoustic distribution off a cylinder [146]; and acoustic radiation from an engine exhaust duct [5, 160]. The 3D engine bypass cases contain a spinning modal radiation out of a generic engine bypass duct [36] and sound radiation from a bypass duct with bifurcations [37]. The analysis of the performance on these cases will be given and summarized altogether in section 4.6.6.

4.6.1 Radiation of a Spinning Mode Out off an Unflanged Duct

This case simulates a spinning mode radiating off a semi-infinite duct. The geometry is simple. A schematic of the physical domain with boundary conditions is shown below:

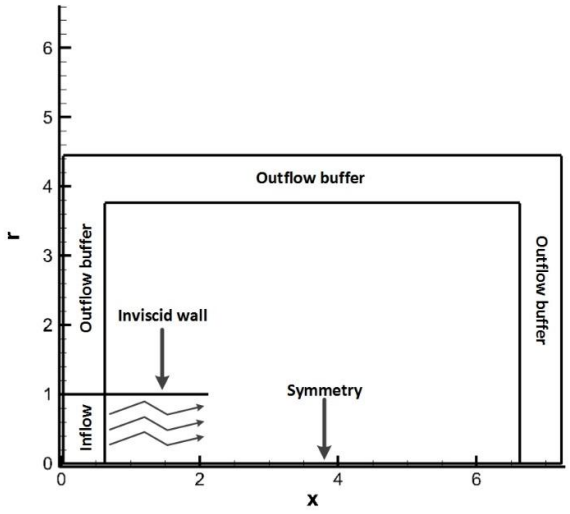


Figure 4.7 Schematic of the physical domain with boundary conditions.

As shown in Figure 4.7, the physical domain is two dimensional and the governing equation is LEE in cylindrical coordinate system. Normally, the 3D LEE in cylindrical coordinate system takes the form below:

$$\partial \rho '/\partial t + \nabla \bullet \left(\rho '\mathbf{V}_0 + \rho_0 \mathbf{V} ' \right) = 0 \tag{3.10}$$

$$\partial u '/\partial t + \mathbf{V}_0 \bullet \nabla u ' + \mathbf{V} ' \bullet \nabla u_0 + \partial p '/\rho_0 \partial x = 0 \tag{3.11}$$

$$\partial v '/\partial t + \mathbf{V}_0 \bullet \nabla v ' + \mathbf{V} ' \bullet \nabla v_0 + \partial p '/\rho_0 \partial r = 0 \tag{3.12}$$

$$\partial w '/\partial t + \mathbf{V}_0 \bullet \nabla w ' + \mathbf{V} ' \bullet \nabla w_0 + \partial p '/\rho_0 r \partial \theta = 0 \tag{3.13}$$

$$\partial p '/\partial t + \mathbf{V}_0 \bullet \nabla p ' + \gamma \left(p_0 \nabla \bullet \mathbf{V} ' + p ' \nabla \bullet \mathbf{V}_0 \right) = 0 \tag{3.14}$$

x, *r* and ϑ denote the coordinates in the axial, radial and azimuthal directions respectively. The incident acoustic disturbance radiating from the inflow buffer is a spinning duct mode which is harmonic dependent and can be written in the following form:

$$p ' = \sum_m p '_m \left(x, r \right) e^{i\left(kt - m\theta \right)} \tag{3.15}$$

$$u ' = \sum_m u '_m \left(x, r \right) e^{i\left(kt - m\theta \right)} \tag{3.16}$$

$$v ' = \sum_m v '_m \left(x, r \right) e^{i\left(kt - m\theta \right)} \tag{3.17}$$

$$w ' = \sum_m w '_m \left(x, r \right) e^{i\left(kt - m\theta \right)} \tag{3.18}$$

Therefore, the relation below holds:

$$\partial w '/\partial \theta = - \left(m/k \right) \times \partial w '/\partial t \tag{3.19}$$

$$\partial^2 p '/\partial t \partial \theta = mkp ' \tag{3.20}$$

Eqs. (4.19) and (4.20) are utilized to remove the partial derivatives with respect to ϑ in Eqs. (4.10) – (4.14). Furthermore, assuming that the mean flow field is axisymmetric and isentropic with no swirl, Eqs (4.10) to (4.14) are simplified into the following form:

$$\partial \rho '/\partial t = - \partial \left(\rho ' u_0 + \rho_0 u ' \right)/\partial x - \partial \left(\rho ' v_0 + \rho_0 v ' \right)/\partial r + m \rho_0 w '_t /kr - \left(\rho ' v_0 + \rho_0 v ' \right)/r \tag{3.21}$$

$$\partial u '/\partial t = - u_0 \, \partial u '/\partial x - v_0 \, \partial u '/\partial r - \left(u ' + u_0 \rho '/\rho_0 \right) \partial u_0 /\partial x - \left(v ' + v_0 \rho '/\rho_0 \right) \partial u_0 /\partial r - \partial p '/\rho_0 \partial x \tag{3.22}$$

$$\partial v '/\partial t = - u_0 \, \partial v '/\partial x - v_0 \, \partial v '/\partial r - \left(u ' + u_0 \rho '/\rho_0 \right) \partial v_0 /\partial x - \left(v ' + v_0 \rho '/\rho_0 \right) \partial v_0 /\partial r - \partial p '/\rho_0 \partial r \tag{3.23}$$

$$\partial w '_t /\partial t = - u_0 \, \partial w '_t /\partial x - v_0 \, \partial w '_t /\partial r - mkp '/\rho_0 r - w '_t \, v_0 /r \tag{3.24}$$

$$p ' = c_0^2 \rho ' \tag{3.25}$$

in which $w'_t = \partial w/\partial t$. Eqs. (4.21) – (4.25) are called 2.5D LEE since they describe the 3D acoustic behaviour on a 2D domain and all the derivative terms with respect to ϑ are removed. The 2.5D LEE has been utilized on some applications [37]. The incident wave is defined as [37]:

$$p' = \mathrm{Re}\left\{\varepsilon J_m\left(k_r r\right)\exp\left[i\left(kt - k_x x - m\theta\right)\right]\right\} \tag{3.26}$$

$$u' = k_r p'/\left(k - k_r M_\infty\right) \tag{3.27}$$

$$v' = \mathrm{Re}\left\{\frac{a}{k - k_x M\infty}\frac{dJ_m\left(k_r r\right)}{dy}\exp\left[i\left(kt - k_x x - m\theta + \pi/2\right)\right]\right\} \tag{3.28}$$

$$w'_t = \mathrm{Re}\left\{-\frac{mkaJ_m\left(k_r r\right)}{r\left(k - k_x M\infty\right)}\exp\left[i\left(kt - k_x x - m\theta + \pi/2\right)\right]\right\} \tag{3.29}$$

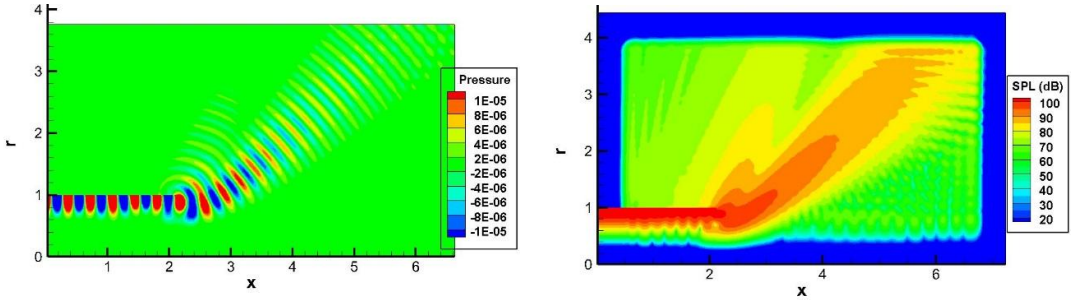
$$k_x = \left(k/\beta^2\right)\left(-M_\infty \pm \sqrt{1 - \xi^{-2}}\right) \tag{3.30}$$

$$\beta = \sqrt{1 - M_\infty^2} \tag{3.31}$$

$$\xi = k/k_r\,\beta \tag{3.32}$$

where ε is the non-dimensional acoustic amplitude, which is 10^{-4} in this case to ensure small changes in density. m denotes the order of the spinning mode, whereas k_r and k_x represent the wavenumbers in the radial direction and in the axial direction respectively. J_m is the Bessel function of the first kind with order m. The reference length is the cylinder radius, which is 1.0 m, and the reference velocity and density scales are set as the ambient sound speed c_∞, 340 m/s and density ρ_∞, 1.225 kg/m^3. The background mean flow is stationary. In this simulation, m = 13, k = 23 with a frequency of 1245 Hz. The grid is uniform. The time-step size is set to 0.27333 × 10^{-2} and the simulation lasts for 50 acoustic periods. A more detailed description of the case can be found in Zhang et al. [34].

The total amount of the mesh points is 32,208 distributed in 12 blocks. Since the computational scale of this case is not large, only one GPU is employed. The pressure and SPL contours are shown below:



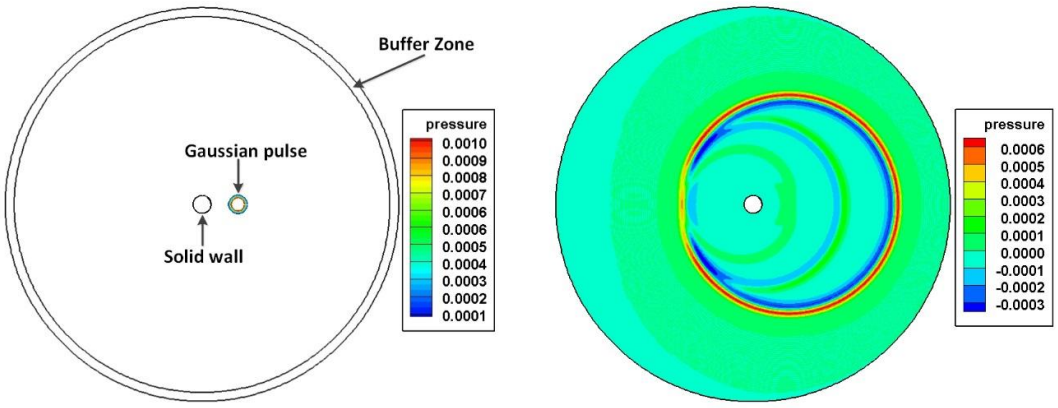
a) Acoustic pressure contour at the 50th cycles. b) SPL contours.

Figure 4.8 Acoustic radiation out of a semi-infinite duct.

The result shown in Figure 4.8 is found to be identical to that implemented on CPUs, by a direct comparison of the output files, since the total amount of mesh points is not large. The application runs 17 times faster on a GPU than on a CPU core. The analysis of the performance will be given in section 4.6.6.

4.6.2 Scattering of a 2D Gaussian Pulse by a Cylinder

This case is used to demonstrate the bidiagonal matrix solver for 2D computations on a multi-block curvilinear mesh. In this benchmark case, the sound field is initialized by a Gaussian pulse expressed by Eqs (3.5) – (3.8). The amplitude ε is 0.01 and d is 0.2. The acoustic computation is also in non-dimensional format. The reference length is the cylinder diameter, which is 1.0 m, and the reference velocity and density scales are set as the ambient sound speed c_∞, 340 m/s and density ρ_∞, 1.225 kg/m^3. The background mean flow is stationary and the sound source is positioned at (2.0, 0, 0) m, which is shown in Figure 4.9.



a) The computational domain. b) The non-dimensional pressure contour at the 1,000th time step.

Figure 4.9 Schematics of 2D cylinder scattering case.

In this case, two blocks make up the computational domain with 1.92×10^5 mesh points. The inner mesh block is the computational domain and contains 1.83×10^5 mesh points, whereas the outer block is the outflow buffer and only contains 1.0×10^4 mesh points. The performance is dominated by the inner mesh block. The non-dimensional time step size is set to 0.006 and the non-dimensional pressure contours at the 1,000th time step are shown in Figure 4.9. The application runs 77 times faster on a GPU than on a CPU core.

4.6.3 A Spinning Mode Scattering off a 2.5D Engine Bypass Duct

The case simulates a high-order acoustic spinning mode radiating off a 2D engine bypass duct by using the 2.5D LEE. This case is mainly used to demonstrate the performance of the solver when a complex geometry is used for 2D computations. The schematic of the computational configuration is shown below:

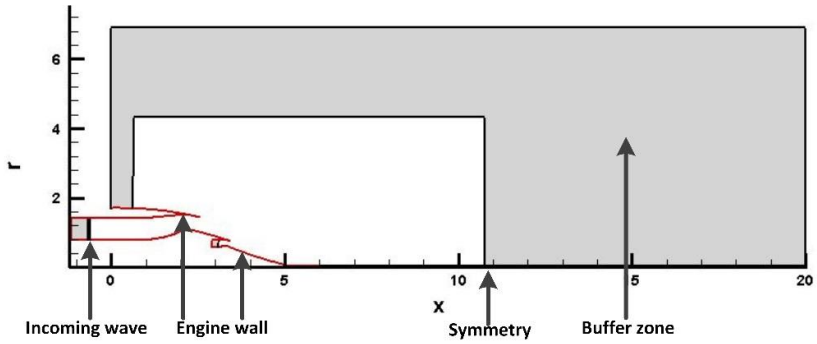
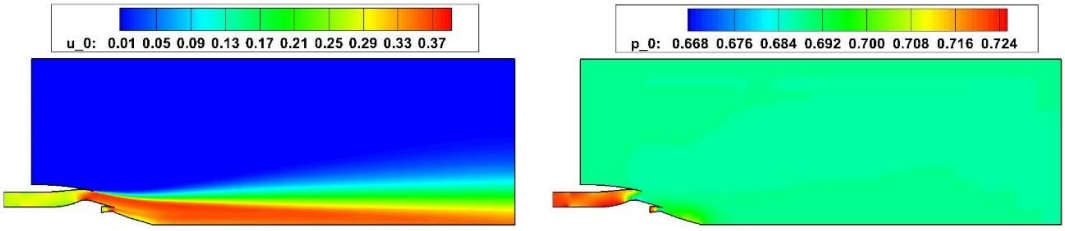


Figure 4.10 Physical domain with boundary conditions in 2.5D bypass case. The red curves denote inviscid solid engine wall, while the grey parts denote the outflow buffer zones which surround the physical domain.

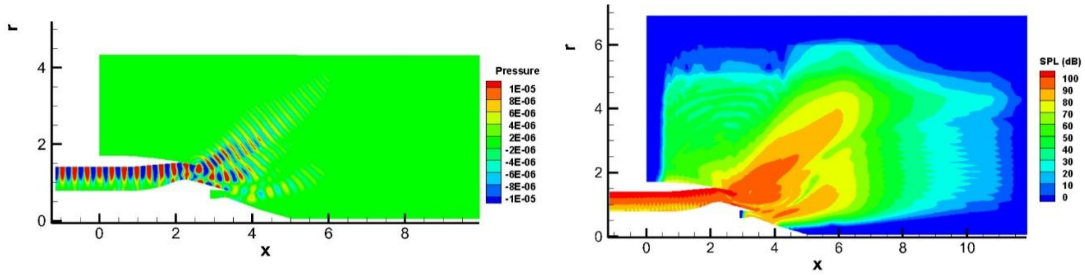
This case is similar to the semi-infinite duct case but simulates the radiation of the acoustic disturbance from a more complex geometry. A detailed description of the geometry can be found in Huang et al [36]. The incident wave is a spinning mode with an order of 12 at the frequency of 1500 Hz and is injected via the inflow buffer at the beginning end of the bypass duct. The bypass duct is axisymmetric and the central axis is aligned with r = 0.05 m with symmetry boundary condition. The mesh blocks in grey are damped through buffer zones, shown in Figure 4.10. The reference density ρ_∞ is 1.225 kg/m^3 and the reference temperature is 299.2 K. The reference length is 1 m and the reference velocity scale is 346 m/s. The background mean flow is obtained by a RANS simulation.



a) Mach number in the *x* direction. b) Non-dimensional pressure.

Figure 4.11 The background mean flow field for bypass duct radiation.

The total domain contains 22 mesh blocks with a total of 40,877 mesh points. The acoustic pressure field at the 50th cycle and the SPL are reproduced, as shown below in Figure 4.12. The application runs 21 times faster on a GPU than on a CPU core.



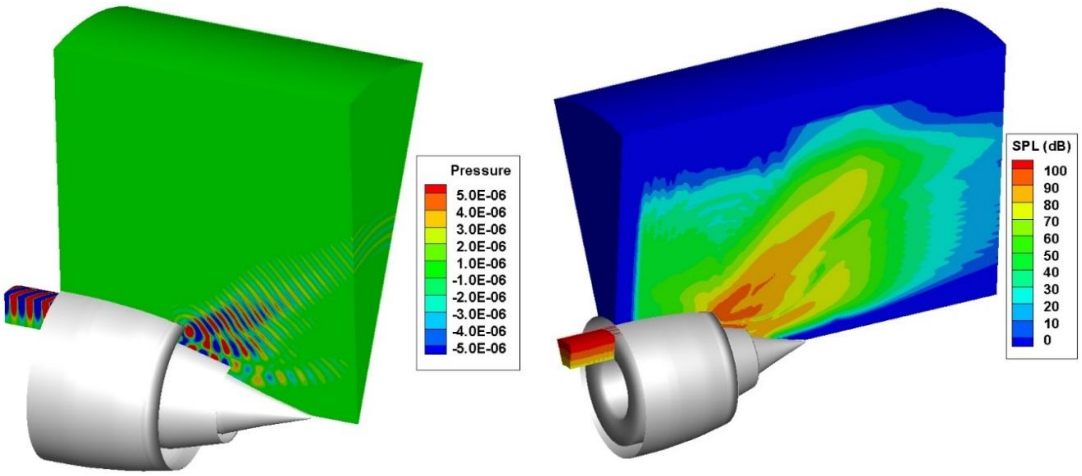
a) Acoustic pressure at the 50th cycle. b) SPL contours.

Figure 4.12 Acoustic pressure and SPL radiated by a 2.5D bypass duct.

4.6.4 A Spinning Modal Radiation Out of a Generic Engine Bypass Duct

This case simulates a spinning modal radiation out of a generic engine bypass duct [160]. It is mainly used to demonstrate the performance of the CAA solver when multiple GPUs are employed for 3D computations with a complex geometry. The computation is in non-dimensional format. The sound source is a spinning mode with the spinning order of 12 with a frequency of 1500 Hz, propagating in a generic 3D engine bypass duct. In this case, the 2D domain created in the previous case which is shown in Figure 4.10 is duplicated in the azimuthal direction to obtain the 3D physical domain. The 3D physical domain is set up as 1/12 of a complete 3D domain as the spinning order of the sound source is 12. Periodic boundary condition is applied on the boundaries in the circumferential direction. The background mean flow outside the bypass duct is stationary with a reference density of 1.225 kg/m^3 and a reference temperature of 299.2 K. The reference velocity scale is 346 m/s whereas the reference length is set to 1 m. The physical domain contains a total of 22 mesh blocks with 3.27x10^5 elements. A detailed description of the case can be found in Zhang et al [5]. The background mean flow was also obtained through the duplication of the 2D RANS simulation shown

in Figure 4.10 in the circumferential direction. The 3D acoustic pressure at the 50th cycle and the SPL contours are shown in Figure 4.13. The whole 3D spinning modal radiation patterns can be obtained by duplicating the 3D CAA domain 12 times in the circumferential direction and results are shown in Figure 4.14. The application runs 40 times faster on two GPUs in comparison to the performance running with two MPI processes.



a) Acoustic pressure at 50th cycle. b) SPL contours.

Figure 4.13 Sound propagation out of a 3D bypass duct.

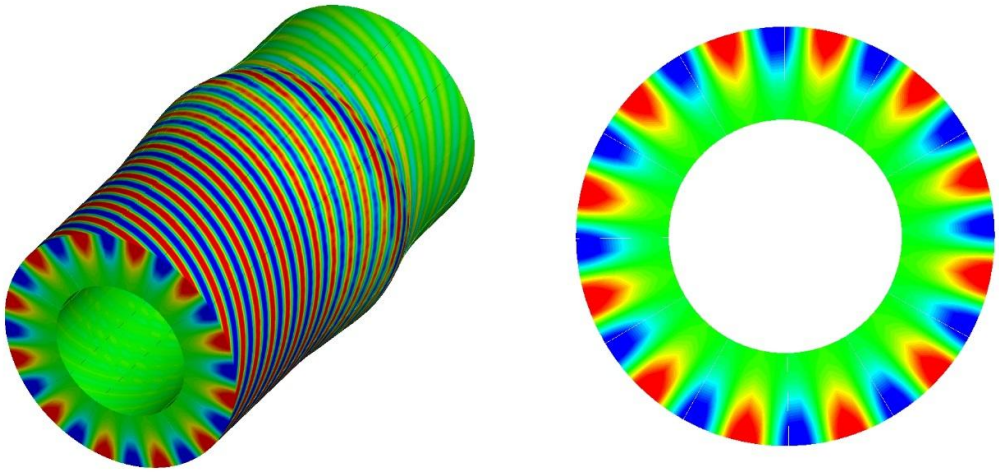
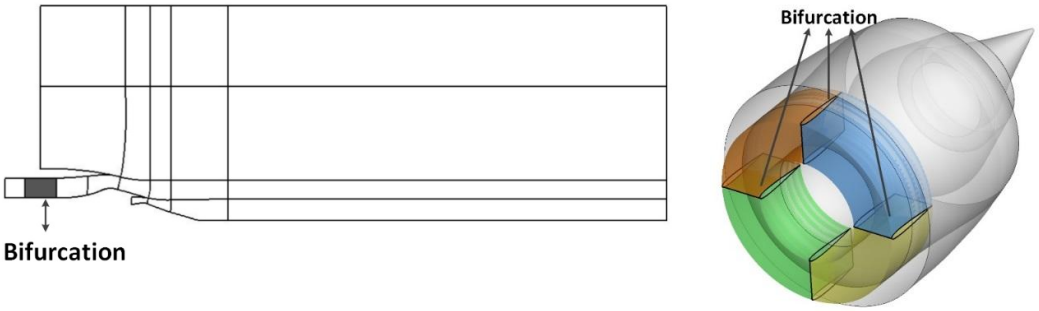


Figure 4.14 The spinning modal pattern in a generic 3D bypass duct, m =12, n = 1.

4.6.5 A Spinning Modal Propagation Out of a 3D Engine Bypass with Bifurcations

This case is a 3D case with a larger computational scale in comparison to the fourth one. The physical domain composes of a total of 22 mesh blocks with 1.2 million mesh points. The bifurcation refers to the structures connecting the outer wall and the inner wall of the bypass duct, which is

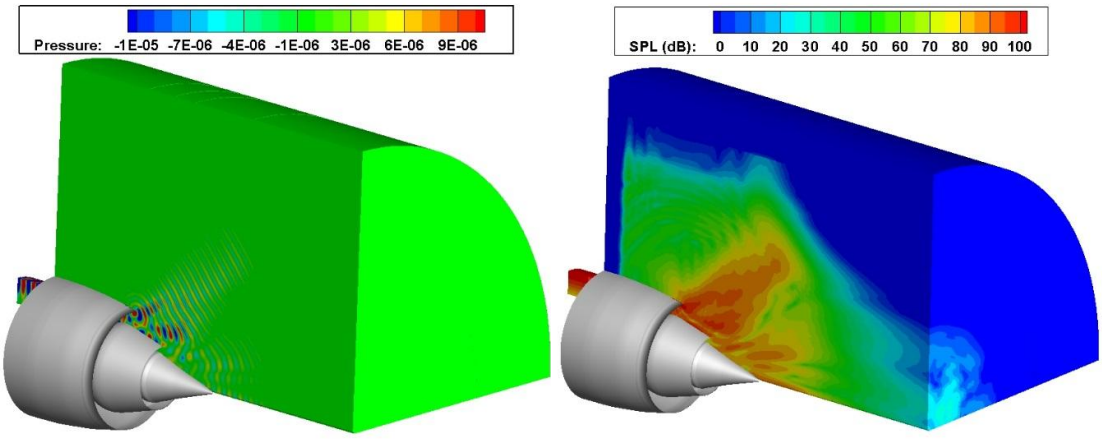
shown in Figure 4.15 b). A total of four bifurcation parts distribute evenly in an annual duct; the computational domain is a quarter of the complete 3D bypass duct.



a) Schematic of the physical domain. b) Bifurcations.

Figure 4.15 Computational setup for sound radiation out of an engine bypass duct with bifurcations.

The simulation uses the same 2D mean flow and mesh settings as the propagation in the clean duct. In 3D acoustic computation, the physical domain is extended to a quarter of a whole 3D domain which is shown in Figure 4.16. The boundary condition of the bifurcation is set as inviscid wall. The simulation is run in non-dimensional format. The boundary conditions, the reference values and the background mean flow field are the same as those in the previous clean bypass duct case. The setup of the present test case is shown in Figure 4.15. The pressure and SPL contours are shown in Figure 4.16. The complete 3D modal radiation patterns are recovered by duplication four times in the circumferential direction in Figure 4.17 and Figure 4.18. The speed-up of the application on two GPUs achieves 54 in comparison with the performance run by two MPI processes.



a) Acoustic pressure at 50th cycle. b) SPL contours.

Figure 4.16 Sound radiation out of an engine bypass duct with bifurcations.

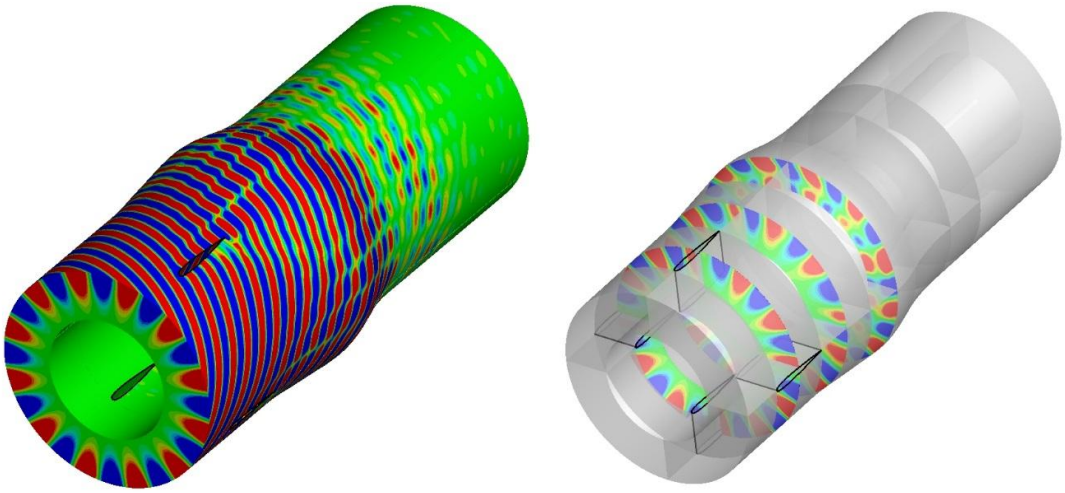
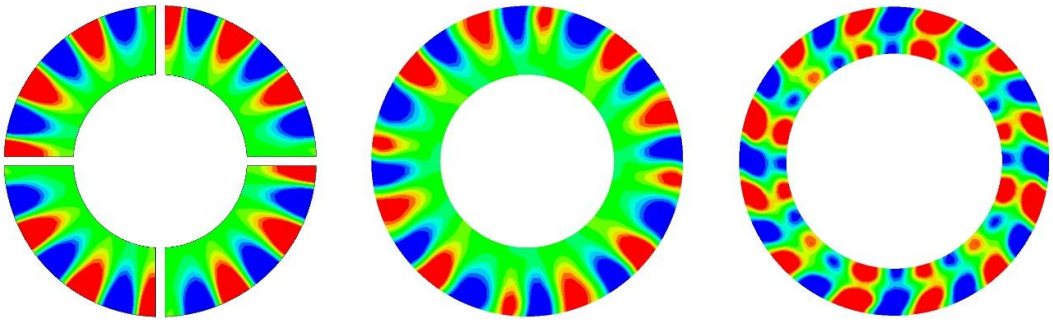


Figure 4.17 Modal radiation patterns of the engine bypass duct with bifurcations.



a) x = 0.0025. b) x = 1.0525. c) x = 2.0525.

Figure 4.18 The slices of modal pattern along the bypass duct with bifurcations.

4.6.6 Performance

The speed-ups of the five benchmark cases above are summarized in Table 4-3 below:

Table 4-3 The speed-ups of the benchmark cases.

Case name	Total mesh points	Maximum block size	Speed-up
2.5D duct	32,208	12,231	17.0
2.5D bypass	40,877	3,721	21.2
2D cylinder	192,881	183,261	77.7
3D bypass	327,016	29,768	39.6
3D bifurcation	1,267,187	115,351	54.3

First, it can be observed from Table 4-3 that the speed-ups achieved are reasonable and consistent with the performance of the bidiagonal matrix solver obtained in chapter 3. The lowest speed-up is 17.0, whereas the highest is 77.7 for 2D computations. Second, it can be observed that the speed-

up increases with the increase of the total amount of mesh points respectively in the 2D and 3D computations. The lowest speed-up occurs in the 2.5D duct case with the minimum number of mesh points, whereas the 3D bifurcation case with the largest number of mesh points achieves the second largest speed-up 54.3. The maximum speed-up emerges in the 2D cylinder case whose total number of mesh points is less than the 3D bifurcation case. This point does not conflict with the conclusion made in the second point. The wall-clock time of a case is dominated by that in the block with the maximum size and the speed-up increases monotonously with the increase of the size of the largest mesh block. Therefore, the 2D cylinder case with the largest mesh block achieves the highest speed-up. In addition, the cache size plays an important role in the performance of the codes on CPUs. The largest mesh block in the 2D cylinder case is 183,261, which runs on a single CPU core and results in severe cache miss. Therefore, a decrease in the performance on the CPU occurs, which results in a higher speed-up. In short, the overall speed-up is dictated by both the largest mesh block and the total number of the mesh points.

4.7 Summary

In total, there are five types of subroutines in terms of the memory access patterns in the current solver SotonLEE. In this section, the parallel strategies of the explicit stencil type, point-wise type, unstructured gather type and the reduction type of subroutines have been investigated and implemented on GPUs. Furthermore, the data transfers between GPUs have been mitigated and optimized by using GPUDirect. Lastly, some benchmark cases which have been validated in previous research were reproduced by implementation on multiple GPUs. The speed-up was summarized. The speed-up achieves 54 when the total amount of the mesh point is over 1.2 million in the 3D computation.

Chapter 5: Acoustic Scattering off a Cylinder

Chapters 2 to 4 are devoted to the development of the CAA solver, SotonLEE, on multiple GPUs. The following two chapters will focus on the application of SotonLEE to a typical large-scale engineering case, the propeller noise scattering off an aircraft. Before the application of SotonLEE_GPU to the engineering case, a simplified geometry is used to verify the numerical methods, acoustic source models and investigate acoustic refraction effect of the boundary layer in this chapter. A single cylinder is utilized as an ideal fuselage. The propeller noise sources are approximated by a ring of monopoles and a ring of dipoles respectively to investigate the thickness noise and the loading noise.

As mentioned in Section 1.2.4.2, scattering of propeller noise off a cylinder has been investigated by experimental [113, 117, 118] and analytical methods [113-117, 119]. In this study, the same phenomenon is investigated by the CAA method for the first time. In comparison to analytical methods, the boundary layer, obtained from a RANS simulation, is realistic, varying in both radial and flow direction. In addition, the purpose of this chapter is mainly to validate the CAA methods used to predict the propeller noise scattering off a cylinder, which will be employed to predict a more complex, large-scale engineering case, propeller noise scattering off a wing-body at a full scale. The validation contains two points: source model of propeller noise and PPW of the mesh. The investigation refers to the refraction effect of boundary layer on the scattering of propeller noise off a cylinder.

5.1 Source Model

This chapter examines the scattering of propeller noise off a cylinder. First, the numerical model of the propeller noise source used in the current research has to be added in SotonLEE and validated in free space. As mentioned in section 1.2.4.2, propeller noise sources can be modelled explicitly by the RHS terms in LEE [10, 49, 95, 96]. The comparison between the numerical solution from LEE implementation and analytical solution is performed. The analytical solution comes from Farassat's Formulation 1A [100, 101, 161] in the time domain and Hanson's solution [99, 162] in the frequency domain.

5.1.1 Analytical Solution

The general form of acoustic analogy can be described by FW-H equations [38, 95]. The FW-H equation can be expressed in the following form:

$$\left(\frac{\partial^2}{\partial t^2}-c^2\nabla^2\right)\left\{\left(\rho-\rho_0\right)c^2H\left(f\right)\right\}=\frac{\partial^2\left\{T_{ij}H\left(f\right)\right\}}{\partial x_i\partial x_j}-\frac{\partial\left\{L_i\delta\left(f\right)\right\}}{\partial x_i}+\frac{\partial\left\{Q\delta\left(f\right)\right\}}{\partial t} \tag{4.1}$$

where *H(f)* is the Heaviside step function whereas the control surface is denoted by *f = 0*. The volume space within the control surface is denoted by *f < 0,* whereas *f > 0* represents the volume space outside the control surface. The vector normal to the control surface is $\mathbf{n}=\nabla f$. ∇^2 denotes the Laplace operator. When the density perturbations are small and the observation distance is far away, the term $c^2(\rho - \rho_0)$ can be replaced by *p'*, which is acoustic pressure. *Q* and *L* denote distribution of mass and linear momentum on the control surface. T_{ij} is the Lighthill's stress tensor. The solution of the FW-H equation can be expressed as:

$$\begin{aligned}4\pi p'=&\frac{\partial}{\partial t}\int_{f=0}\left[Q/r\left(1-M_r\right)\right]_{ret}dS\\&-\frac{\partial}{\partial x_i}\int_{f=0}\left[L_i/r\left(1-M_r\right)\right]_{ret}dS\\&+\frac{\partial^2}{\partial x_i\partial x_j}\int_{f>0}\left[T_{ij}/r\left(1-M_r\right)\right]_{ret}dV\end{aligned} \tag{4.2}$$

$$Q=\rho_0U_in_i \text{ with } U_i=u_i+\left[\left(\rho/\rho_0\right)-1\right]\times\left(u_i-v_i\right) \tag{4.3}$$

$$L_i=P_{ij}n_j+\rho u_i\left(u_n-v_n\right) \text{ with } P_{ij}=\left(p-p_0\right)\delta_{ij}-\tau_{ij} \tag{4.4}$$

$$T_{ij}=\rho u_iu_j+\left(p'-c^2\rho'\right)\delta_{ij}-\tau_{ij} \tag{4.5}$$

$$r=\left|\mathbf{x}-\mathbf{y}\right| \tag{4.6}$$

where u_i denotes the velocity of the flow field whereas v_i is the velocity of the control surface. *r* denotes the distance between the source and observer and $M_r = M_i \cdot r_i/r$. The source element in $[\ ]_{ret}$ denotes an evaluation at the retarded time:

$$\tau_{ret}=t-\left|\mathbf{x}-\mathbf{y}\left(\tau_{ret}\right)\right|/c \tag{4.7}$$

The observer position and time is denoted by (**x**, *t*), whereas the source is described by (**y**, *τ*). δ_{ij} is the Kronecker Delta function. τ_{ij} is the viscous stress tensor.

The three terms at the RHS in Eq. (5.2) denote thickness noise, loading noise and quadrupole noise respectively. When the Mach number at the blade tip is low, the contribution of the quadrupole sources can be ignored, leading to a linear acoustic formulation that only contains thickness noise and loading noise [96]. The solution to the thickness noise is:

$$4\pi p'_T(\mathbf{x},t) = \int_{f=0}\left[\rho_0\left(\dot{U}_n\right)\Big/\left(r\left(1-M_r\right)^2\right)\right]_{ret} dS + \int_{f=0}\left[\rho_0 U_n\left(r\dot{M}_r + c\left(M_r - M^2\right)\right)\Big/\left(r^2\left(1-M_r\right)^3\right)\right]_{ret} dS \tag{4.8}$$

$$\left(\dot{U}_n\right) = \left(U_i \bullet \dot{n}_i\right) = \dot{U}_i \bullet n_i + U_i \bullet \dot{n}_i \tag{4.9}$$

$$M_r = M_i \bullet r_i / r \tag{4.10}$$

$$\dot{M}_r = \dot{M}_i \bullet r_i / r \tag{4.11}$$

The dot on the variables denotes the time derivative with respect to the source time τ. The key point in Eq. (5.8) is the determination of the retarded time [96, 100, 161] which can be obtained by the iteration of Eq. (5.7). For spinning monopoles with a constant strength and a given angular frequency, the acoustic pressure at a given time $t = t_0$ can be identified via Eq. (5.8). By using the similar procedure, the solution to the loading noise due to the second term in Eq. (5.2) is:

$$4\pi p'_L(\mathbf{x},t) = \frac{1}{c}\int_{f=0}\left[\dot{L}_r\Big/\left(r\left(1-M_r\right)^2\right)\right]_{ret} dS + \int_{f=0}\left[\left(L_r - L_M\right)\Big/\left(r^2\left(1-M_r\right)^2\right)\right]_{ret} dS + \frac{1}{c}\int_{f=0}\left[L_r\left(r\dot{M}_r + c\left(M_r - M^2\right)\right)\Big/\left(r^2\left(1-M_r\right)^3\right)\right]_{ret} dS \tag{4.12}$$

$$L_r = L_i \bullet r_i / r \tag{4.13}$$

$$\dot{L}_r = \dot{L}_i \bullet r_i / r \tag{4.14}$$

$$L_M = L_i \bullet M_i \tag{4.15}$$

Eqs. (5.8) and (5.12) constitute the solution of FW-H equation in the time domain. For spinning dipoles with a constant strength and a given frequency, the acoustic pressure at a given time $t = t_0$ can also be identified. For compact sources such as rotating monopoles and dipoles, Eqs. (5.8) and (5.12) can be reduced to:

$$4\pi p'_T(\mathbf{x},t) = \left[\rho_0 U_n\left(r\dot{M}_r + c\left(M_r - M^2\right)\right)\Big/\left(r^2\left(1-M_r\right)^3\right)\right]_{ret} \tag{4.16}$$

$$4\pi p'_L(\mathbf{x},t) = \left[\left(\dot{L}_r \bullet r - cL_M\right)\Big/\left(cr^2\left(1-M_r\right)^2\right)\right] + \left[L_r r\left(\mathbf{r}\bullet\dot{\mathbf{M}} + c\left(1-M^2\right)\right)\Big/\left(cr^3\left(1-M_r\right)^3\right)\right]_{ret} \tag{4.17}$$

Also, the solution of the thickness noise and the loading noise to Eq. (5.1) can also be obtained in the frequency domain by using Hanson's method [99, 162]:

$$p'_T(\mathbf{x},t) = \sum_{m=-\infty}^{\infty} P'_{TmB}(\mathbf{x}) e^{-imB\omega t} \tag{4.18}$$

$$P'_{TmB}(\mathbf{x}) = \rho_0 B \int_{f=0} e^{-imB\phi_s} \frac{1}{2\pi} \int_0^{2\pi} U_n \left(-imB\omega G_{mB}\right) e^{imB\phi_0} \bullet d\phi_0 dS \tag{4.19}$$

$$p'_L(\mathbf{x},t) = \sum_{m=-\infty}^{\infty} P'_{LmB}(\mathbf{x}) e^{-imB\omega t} \tag{4.20}$$

$$P'_{LmB}(\mathbf{x}) = B \int_{f=0} e^{-imB\phi_s} \frac{1}{2\pi} \int_0^{2\pi} L_i(\phi - \phi_s) \frac{\partial G_{mB}}{\partial y_i} e^{imB\phi} d\phi dS \tag{4.21}$$

$$G_{mB} = e^{imB\omega r/c_0} / 4\pi r \tag{4.22}$$

$$r = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} \tag{4.23}$$

in which $p'_T(\mathbf{x}, t)$ denotes the thickness noise, whereas $P'_{TmB}(\mathbf{x})$ represents the m^{th} harmonic of the Fourier transform of $p'_T(\mathbf{x}, t)$. B is the total number of blades, whereas ω is the angular frequency of blade rotation. G_{mB} is the Green's function in the frequency domain. Eqs. (5.18) – (5.21) offer the solution of FW-H equations in the frequency domain.

If the acoustic sources are monopoles or dipoles, then the term $e^{-imB\phi_s}$ can be ignored. Eqs. (5.19) and (5.21) can be expressed in the discrete form in terms of φ:

$$\begin{aligned} P'_{TmB}(\mathbf{x}) &= \rho_0 B \times \frac{1}{2\pi} \sum_{j=1}^{N} U_n (-imB\omega G_{mB}) e^{imBj2\pi/N} \frac{2\pi}{N} \\ &= -i\rho_0 B^2 U_n \omega m / N \left(\sum_{j=1}^{N} G_{mB} e^{imB2\pi j/N} \right) \end{aligned} \tag{4.24}$$

$$\begin{aligned} P'_{LmB}(\mathbf{x}) &= B \frac{1}{2\pi} \sum_{j=1}^{N} L_i \left(\frac{j2\pi}{N} \right) \frac{\partial G_{mB}}{\partial y_i} e^{imB2\pi j/N} \frac{2\pi}{N} \\ &= \frac{B}{N} \left(\sum_{j=1}^{N} L_i \left(\frac{j2\pi}{N} \right) \partial G_{mB} / \partial y_i \, e^{imBj2\pi/N} \right) \end{aligned} \tag{4.25}$$

$$\phi = j2\pi / N, \; j = 1, \cdots N \tag{4.26}$$

Comparisons will be made between the solution of FW-H equation obtained by Farassat's Formulation 1A, Hanson's method and the numerical solution in next sub-section.

5.1.2 Numerical Solution

5.1.2.1 Introduction of Source Model

In the numerical model, the sources are introduced via the RHS terms in LEE denoted by s_1 to s_5 in Eqs. (2.1) to (2.5). The propeller noise sources are assumed compact, to be a ring of spinning monopoles due to the thickness noise and a ring of spinning dipoles due to the loading noise. This way of source introduction into LEE was proposed by Dierke et al. and realized in the CAA code PIANO [10]. Garrec and Reboul [49] also employed this source model in Euler equations to predict the open rotor noise scattering off a wing-body. In this study, the same source model is used. However, the introduction of dipole source into momentum equation is found not stable in SotonLEE. An equivalent way is used to replace the dipole source which will be introduced in detail in section 5.2.5. A sketch of the ring of the spinning source is shown below:

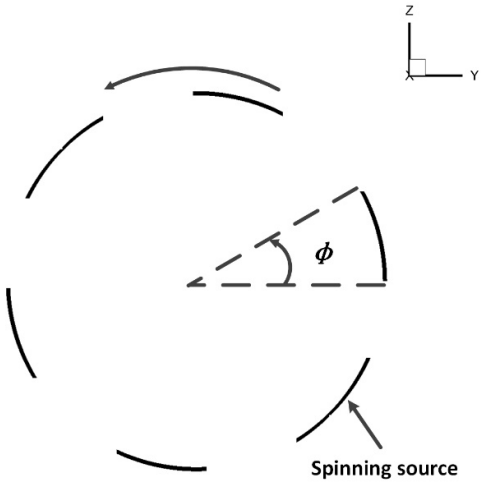


Figure 5.1 Sketch of ring model.

Viewed by an observer downstream as shown in Figure 5.1, the ring source rotates in the counter clock-wise direction. The source ring is placed in the *yz* plane. If the centre of the source ring is given with the coordinate (x_c, y_c, z_c), then the position of the i^{th} monopole/dipole on the ring can be identified by the relations below:

$$\begin{aligned} x_i(t) &= x_c \\ y_i(t) &= y_c + R\cos(\omega t + \phi_i) \\ z_i(t) &= z_c + R\sin(\omega t + \phi_i) \end{aligned} \tag{4.27}$$

where R denotes the radius of the ring and φ_i denotes the relative position between the sources on the ring. The velocity of a source on the ring can be derived from Eq. (5.27):

$$\begin{aligned} u_i(t) &= 0 \\ v_i(t) &= -R\omega\sin\left(\omega t + \phi_i\right) \\ w_i(t) &= R\omega\cos(\omega t + \phi_i) \end{aligned} \tag{4.28}$$

Given the amplitude of the monopole is $Q = \rho_0 U_n$, the source terms are introduced by the source terms s_1 and s_5 of LEE in Eqs. (2.1) – (2.5). The source terms s_5 and s_1 in LEE become:

$$s_5 = s_1 = \sum_{i=1}^{B} \rho_0 U_n \delta\left(x - x_i(t), y - y_i(t), z - z_i(t)\right) \tag{4.29}$$

However, in reality, the amplitude of the monopole is determined by the blade volume. According to the relation in reference [96]:

$$\rho_0 \mathbf{U}\bullet\mathbf{n}\left|\nabla f\right|\delta\left(f\right) = \partial\left(\rho_0\left(1 - H\left(f\right)\right)\right)\Big/\partial t \tag{4.30}$$

For the propeller noise, the wavelengths of the first and the second BPFs are large in comparison to blade dimensions, such as the blade thickness. Consequently, blades are assumed compact sources. For a compact source like a monopole, the source term s_5 and s_1 in LEE become:

$$s_5 = s_1 = \partial\left(\sum_{i=1}^{B} \rho_0 V_B \delta\left(x - x_i(t), y - y_i(t), z - z_i(t)\right)\right)\Big/\partial t \tag{4.31}$$

in which V_B is the volume of a blade. If the synthetic thrust and drag of the blade section are projected into the x, y and z directions and are denoted as F_x, F_y and F_z, the sources of loading noise in the momentum equations can be denoted as:

$$s_2 = -F_x \delta\left(x - x_i(t), y - y_i(t), z - z_i(t)\right) \tag{4.32}$$

$$s_3 = -F_y \delta\left(x - x_i(t), y - y_i(t), z - z_i(t)\right) \tag{4.33}$$

$$s_4 = -F_z \delta\left(x - x_i(t), y - y_i(t), z - z_i(t)\right) \tag{4.34}$$

In reality, the Dirac Delta function cannot be realized by using the finite difference method and therefore is smoothed using the Gaussian distribution:

$$\delta\left(x - x_i(t), y - y_i(t), z - z_i(t)\right) = \frac{1}{\left(2\pi\right)^{3/2}\sigma^3}\exp\left(\frac{-1}{2\sigma^2}\left(\left(x - x_i\left(t\right)\right)^2 + \left(y - y_i\left(t\right)\right)^2 + \left(z - z_i\left(t\right)\right)^2\right)\right) \tag{4.35}$$

where σ is a parameter. In definition of the Gaussian distribution, the point in the extent of 3σ covers the 99.73% of the total energy.

5.1.2.2 PPW Determination at the Source Region

PPW plays a key role in a CAA simulation. It can be utilized to describe the resolution property of a finite difference scheme and it can also be employed to quantify the resolution of a mesh if the wavelength of an acoustic wave is given. In this section, the PPW refers to the resolution property of the mesh, since the fourth-order optimized prefactored compact scheme is used throughout the study.

The requirement of PPW at the source region on a given mesh is validated for a single monopole at the origin, radiating an acoustic wave with a frequency of 120 Hz. Simulations on meshes with different PPWs are performed. Results of SPL and acoustic pressure are shown below:

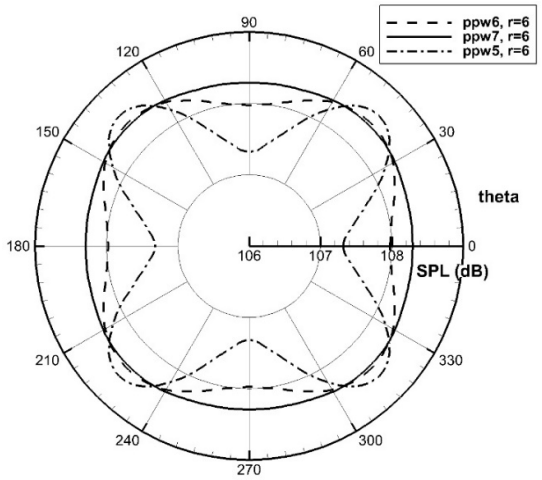


Figure 5.2 SPL directivities on the ring plane with r = 6.

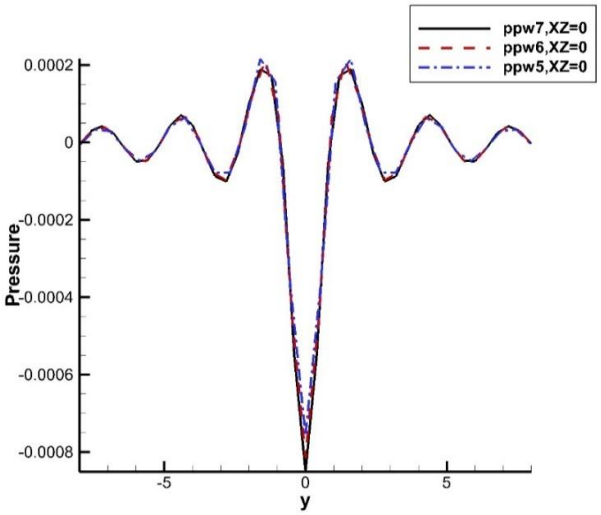


Figure 5.3 Acoustic pressure along the line x = 0, z = 0.

It can be observed from Figure 5.2 that the SPL directivity shows a strong anisotropy when the PPW of the mesh is set to 5, though the PPW of fourth-order optimized prefactored compact scheme is 3.7 in theory, as mentioned in section 1.2.2. The peaks of SPL occur at the angles of 45°, 135°, 225° and 315°, whereas valleys appear at the angles of 0°, 90°, 180° and 270°. The SPL of acoustic waves at the peaks is over-predicted, while waves at the valleys are under-predicted. The gap between the peaks and valleys achieves roughly 1 dB. A similar scenario occurs when PPW is set to 6, but the difference between the peaks and valleys is much smaller, 0.25 dB. When PPW is set to 7, the SPL directivity shows the isotropy in all directions. In addition, the acoustic waveform sampled on the line with $x = 0$, $z = 0$ also dictates that the amplitude achieves the largest value with PPW = 7. The line with $x = 0$, $z = 0$ corresponds to the angles of 0° and 180°, as shown in Figure 5.2.

In addition, the anisotropy of the SPL directivity actually results from the resolution of the acoustic source rather than that in the propagating region according to the PPW analysis of optimized prefactored compact schemes in Section 1.2.2. Since the ideal Dirac Delta function cannot be realized by using the finite difference method, it is replaced by the Gaussian distribution which, however, introduces numerical errors into the numerical simulation.

Recall that the only parameter in Eq. (5.35) is σ which determines the resolution of the monopole source on a given mesh. For a ring of 6 spinning monopoles with a radius of 1 m and the RPM of 1200, the acoustic wave along the line $x = 0$, $z = 5$ m with different σ values on a mesh with PPW of 7 is shown in Figure 5.4:

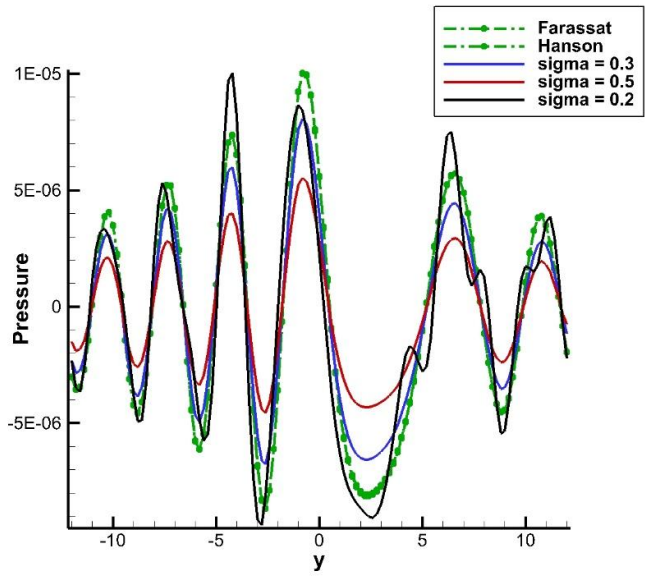


Figure 5.4 Acoustic signature at line with $x = 0$, $z = 5$, $t = 68$ (0.2 s with dimensional value).

In Figure 5.4, it can be observed that replacement of ideal Dirac Delta function by the Gaussian distribution introduces errors into numerical simulations. The waveforms with σ = 0.3 and σ = 0.5 have smaller amplitudes but the same phase in comparison with the analytical solution. That error is mainly the dissipation. The waveform with σ = 0.2 contains not only dissipation error but also dispersion error which is not negligible. The Gaussian distribution smooths the Dirac Delta function via a sphere distribution with a finite radius of about 3σ (99.75% of the total energy). σ with values of 0.3, 0.5 and 0.2 corresponds to Gaussian distributions in spheres with radii of 0.9 m, 1.5 m and 0.6 m which are resolved by 4.5, 7.5 and 3 mesh points respectively. The PPW of the fourth-order optimized prefactored compact scheme is 3.7. Therefore, the Gaussian distribution with σ = 0.3 and above are well resolved. If the errors are mainly the dissipation, the acoustic waveform can be calibrated by an amplification on the numerical simulation. Since the governing equations used, LEE, is a linear system, the amplification factor is obtained by the ratio of pressures between the analytical solution and numerical solution. Once the amplification factor is obtained, it is employed to modulate the source strength, including monopoles and dipoles, in LEE. The scaled numerical waveforms are shown in Figure 5.5 below:

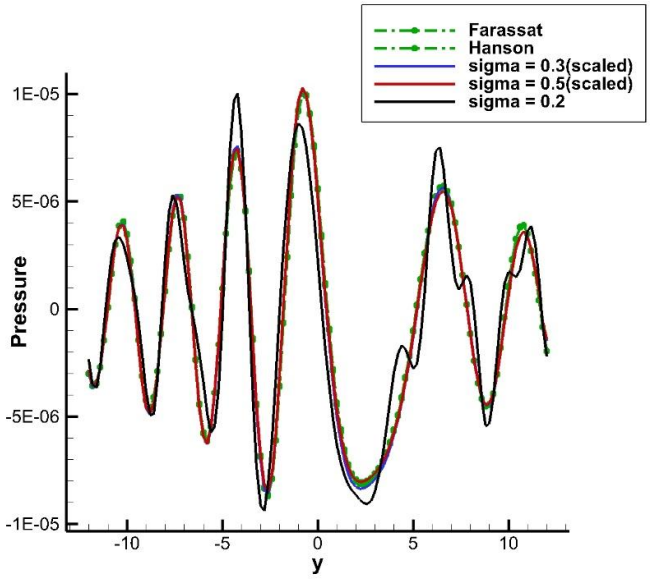


Figure 5.5 Scaled acoustic signature at line with *x* = 0, *z* = 5, *t* = 68 (0.2s with dimensional value).

It can be observed that modulated waveforms with σ = 0.3 and σ = 0.5 achieve good approximation of the exact analytical solution. The dissipation and dispersion errors are negligible. The waveform with σ = 0.2 cannot be scaled to have a good approximation of the analytical solution.

In conclusion, the replacement of an ideal Dirac Delta function with a Gaussian distribution will introduce errors in CAA simulations. The errors are mainly caused by the value of σ on a given mesh.

If the dispersion error can be ignored and only dissipation error exists, the numerical solution can be calibrated. The calibration is necessary for the compact source simulation.

From Hanson's solution, the 1st, 2nd and 3rd harmonics of the power spectral density (PSD) of the acoustic wave radiated by a ring of 6 spinning monopoles with a radius of 1 m and an RPM of 1200 are shown below:

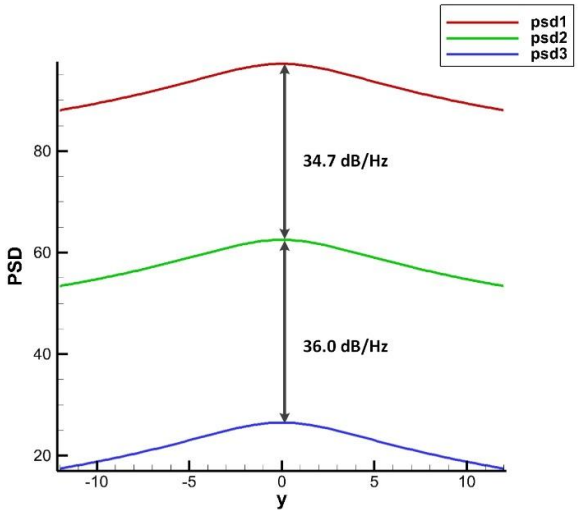


Figure 5.6 PSDs of the first three harmonics of a ring of spinning monopoles.

It can be observed that a gap of roughly 35 dB/Hz exists between two neighbouring harmonics. The contributions of the second and higher order harmonics can be ignored. Consequently, in the following numerical simulation, the mesh is only necessary to resolve the first harmonic with the frequency of 120 Hz.

5.2 Low Mach Number Setup

5.2.1 Sound Source

The noise source is first a ring of 6 rotating monopoles distributed evenly as illustrated in Figure 5.7 to study the thickness noise. Viewed by an observer downstream, the ring at the RHS of the cylinder rotates in the counter clock-wise direction at 1200 RPM. The first harmonic of BPF is 120 Hz. The phases of all the monopoles are set to 0, whereas the non-dimensional amplitude is set to 0.01 to ensure a linear propagation. A ring with the origin at (0, 0, 0) and a radius of 5.2 m is utilized to collect the near-field SPL on the ring plane, whereas another ring with the origin at (0, 0, -4) m and

a radius of 5.2 m is used on the ground. Then, the rotating monopoles are replaced by a ring of 6 rotating dipoles with the same BPF to study the loading noise.

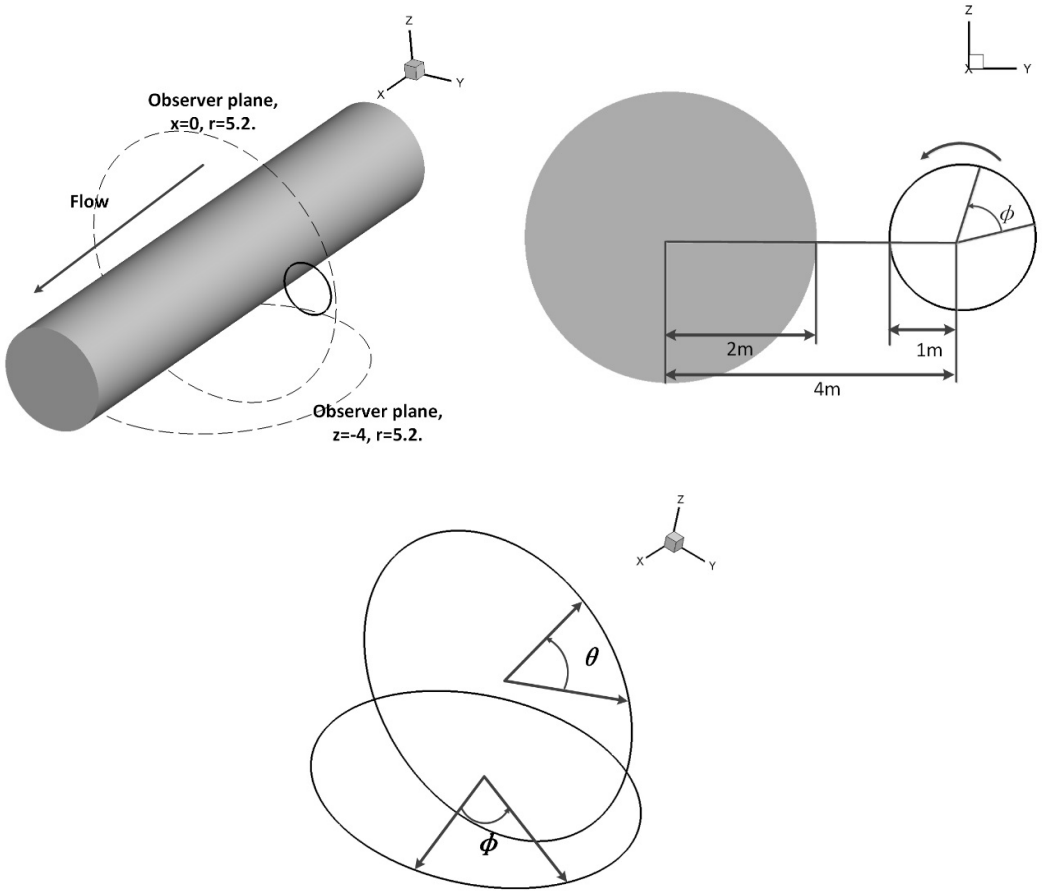


Figure 5.7 Sketch of spinning sources and scattering cylinder.

5.2.2 Computation Setup

In the LEE simulation, the 3D physical domain is 20 m long in the flow direction and 9 m in the radial direction. The boundary condition at the cylinder surface is slip wall, while zones with highly stretched grids are used at the outer boundary to attenuate the perturbations. The boundary conditions on a profile of the physical domain in the stream-wise direction are shown below in Figure 5.8:

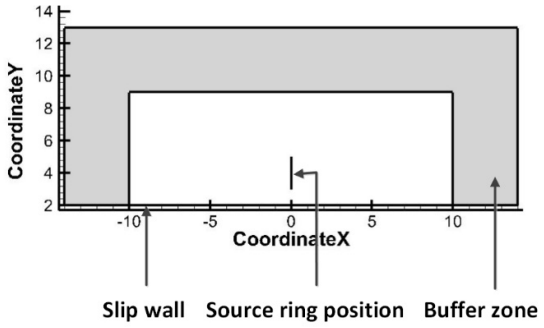
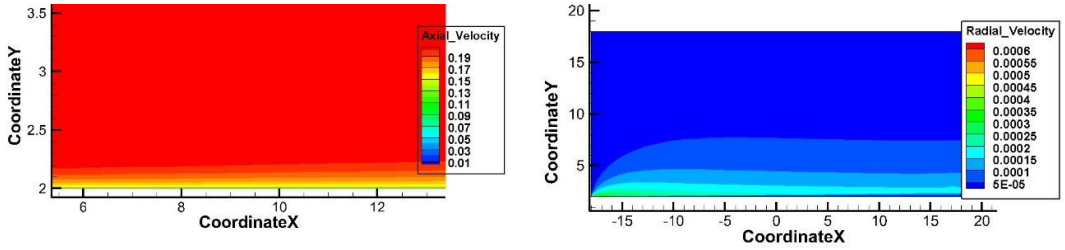


Figure 5.8 Sketch of the physical domain with boundary conditions.

PPW are set to 7 along the stream-wise and the radial directions. The mesh in the azimuthal direction is determined by setting the PPW to 7 at r = 5 m which is the maximum position of source on the ring in the radial direction. The final mesh has 1.2 million mesh points. Non-dimensional variables are used in the computation. The reference length scale L is 1 m, speed scale, c_∞, is 340.3 m/s, and density scale, ρ_∞ is 1.225 kg/m^3. The standard air properties at sea level are used.

5.2.3 Mean Flow

When the refraction effect of the boundary layer is considered, the background mean flow field is obtained from a RANS simulation. The Mach number of the free stream is 0.205. The estimated 99% turbulence boundary layer thickness on the end of cylinder surface is 0.194m, which is quite thin in comparison to the acoustic wavelength of 2.83m.



a) Mach number of the axial velocity. b) Mach number of the radial velocity.

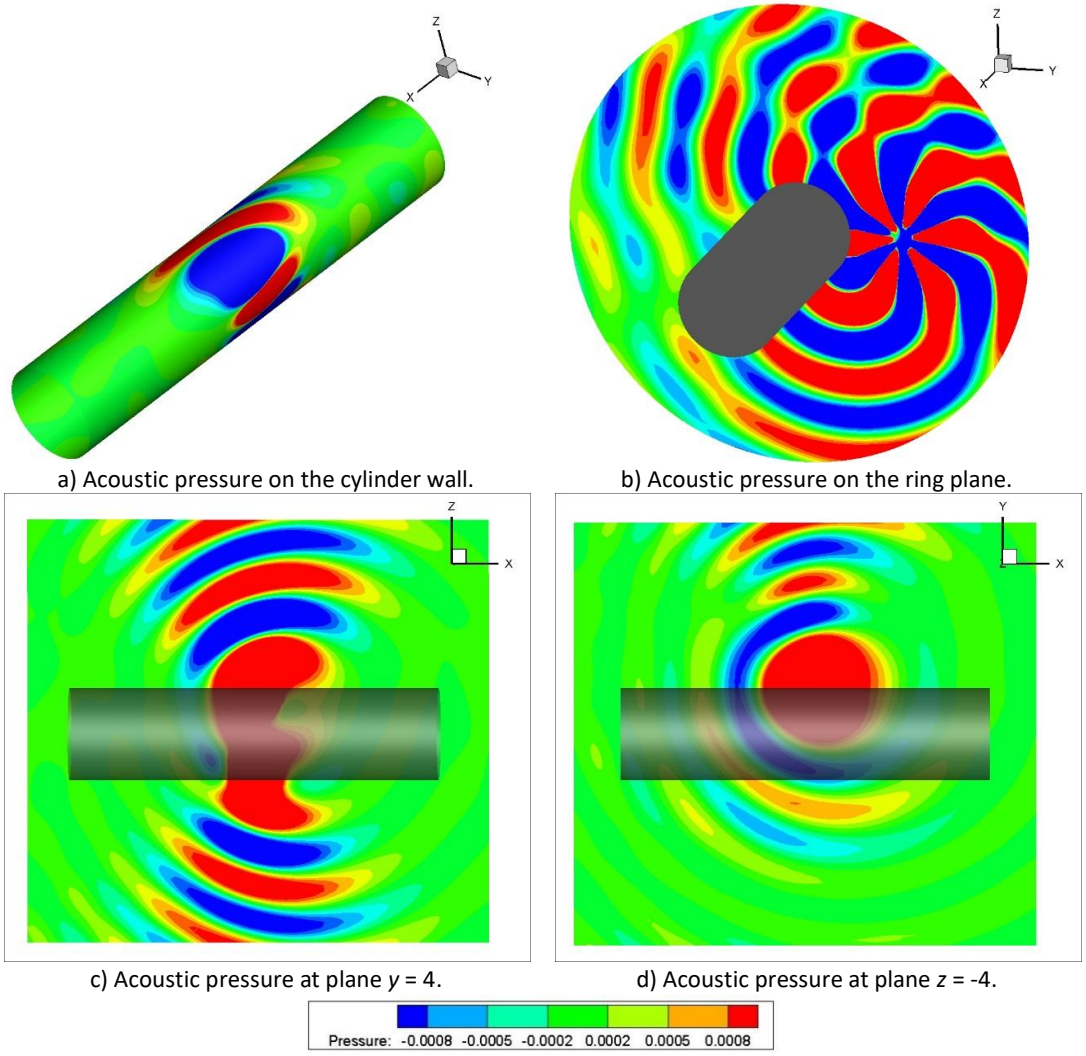
Figure 5.9 Mean flow field obtained from RANS.

5.2.4 Results of Thickness Noise

5.2.4.1 Validation

In this study, the simulation results from LEE with uniform flow and complex equivalent source method (CESM) [107, 108] are first compared to validate the numerical methods and source models

in SotonLEE since a program of CESM has been available and used to predict the scattering of propeller noise in ANTC. As the ESM as mentioned in Section 1.2.4, CESM solves the inhomogeneous convective acoustic equation by a surface method. It is an extension of ESM to the complex plane. In comparison to the traditional ESM, the sources in CESM have a directivity which can be controlled by the ratio of the imaginary and real parts of the source. Therefore, the refraction effect of the non-uniform mean flow is ignored in CESM. The detailed study on CESM can be found in the work of Gounot and Musafir [163], and Hou et al. [108]. The profiles of near-field instantaneous sound pressure field computed by LEE are shown in Figure 5.10 and those of SPL are shown in Figure 5.11.



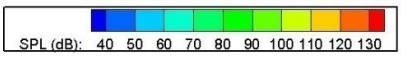
a) Acoustic pressure on the cylinder wall.

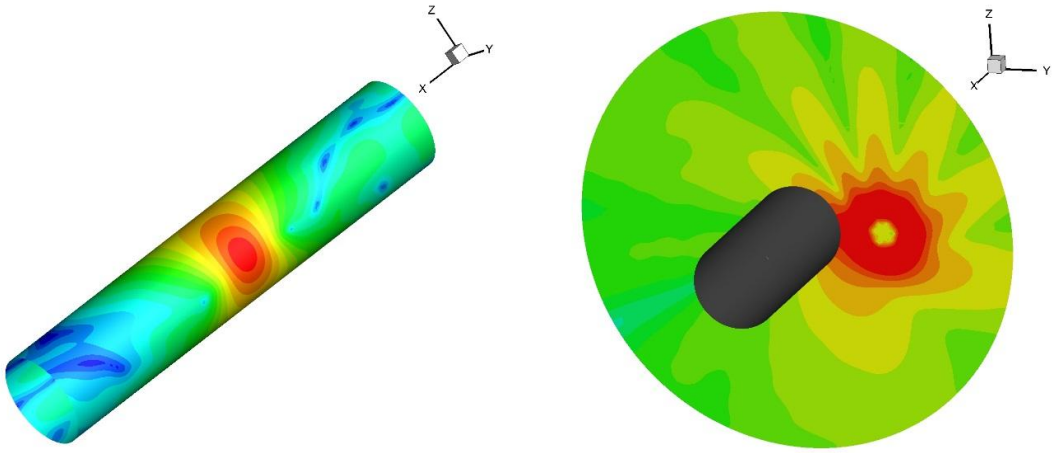
b) Acoustic pressure on the ring plane.

c) Acoustic pressure at plane y = 4.

d) Acoustic pressure at plane z = -4.

Figure 5.10 Instantaneous pressure contours simulated by LEE in uniform flow.

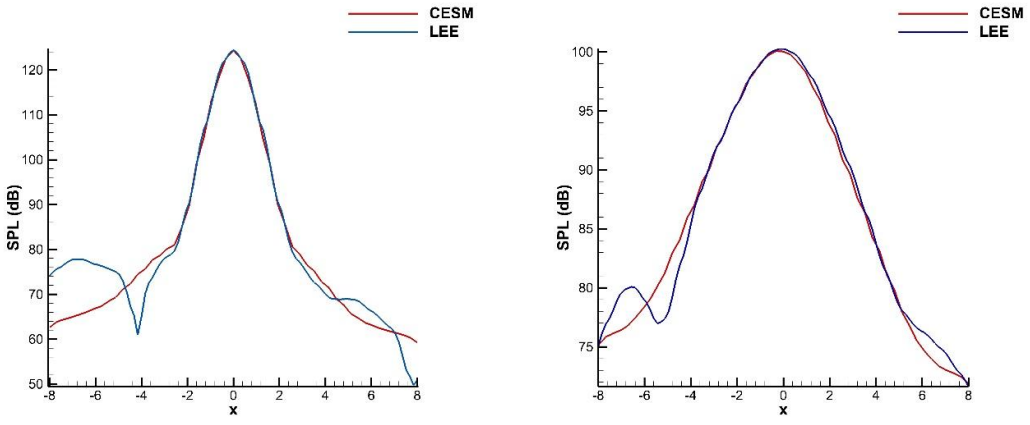




a) SPL on the cylinder wall. b) SPL on the ring plane.

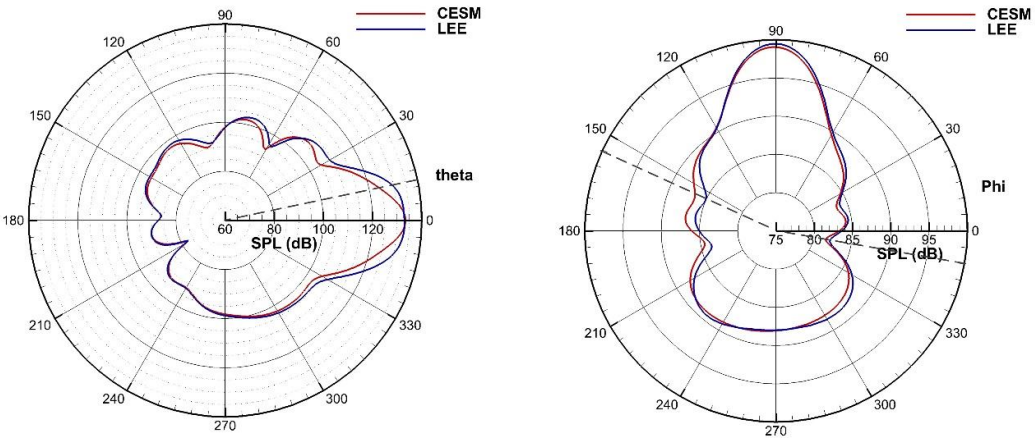
Figure 5.11 SPL contours simulated by LEE in uniform flow.

Sound pressure is propagating spirally in Figure 5.10 b), which illustrates the rotation of the sound sources. The convection effect of mean flow field with low Mach number inclines the propagation of the acoustic waves upstream, as shown in Figure 5.10 c). The sound pressure contour on a plane under the cylinder is shown in Figure 5.10 d). The propagation of acoustic waves concentrates near the ring plane and the RHS of the cylinder where the propeller is located. The concentration of acoustic wave propagation near the ring plane is also demonstrated by Figs.Figure 5.10 a) andFigure 5.11 a). The SPL contour on the ring plane is shown in Figure 5.11 b). It shows a strong directivity on the ring plane and multiple peaks and valleys of SPL distribute due to the scattering of the cylinder wall. Qualitative comparisons of the SPL on the cylinder wall and the far-field directivity are given in Figure 5.12 andFigure 5.13 shown below:



a) SPL on the cylinder wall, $\vartheta = 0$. b) SPL on the cylinder wall, $\vartheta = 1.5\pi$.

Figure 5.12 SPL values on the cylinder wall along the stream-wise direction.



a) SPL directivity at x = 0, r = 5.2. b) SPL directivity at z = -4 m, r = 5.2.

Figure 5.13 Far-field SPL directivities.

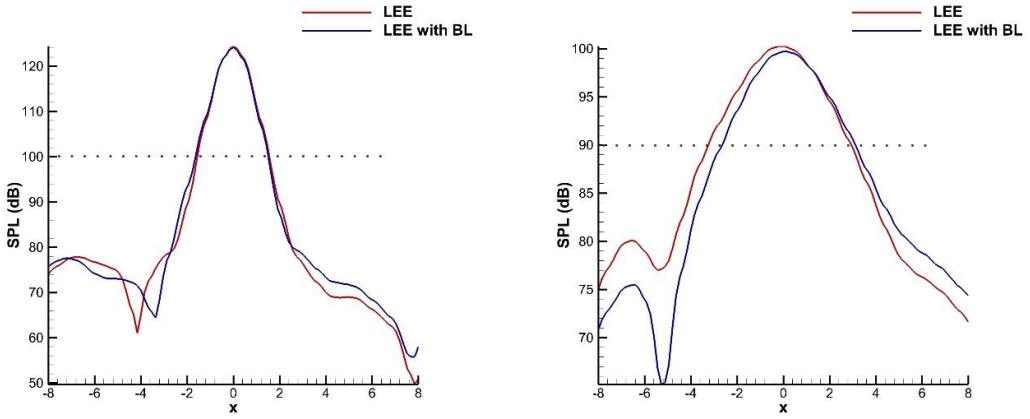
In Figure 5.12 a), the SPL on the cylinder wall line at ϑ = 0, which is defined in Figure 5.7, concentrates on the ring plane and behaves like a Gaussian distribution along the stream-wise direction. The difference between the peaks of SPL values obtained from CESM and LEE is 0.1 dB. The SPL drops sharply and differences between CESM and LEE occur when the observer is far away from the ring plane. This is attributed to the reflection of acoustic waves from the buffer zone in LEE. When observed on the line at $\vartheta = 1.5\pi$, which corresponds to the bottom of the cylinder wall, the peaks of SPL values are 100.08 dB and 100.28 dB for CESM and LEE respectively. The difference between CESM and LEE is less than 0.2 dB.

At the far-field as it is shown in Figure 5.13, two rings of observers are used to collect the SPL directivities which are shown in Figure 5.7. ϑ is defined in the counter clock-wise direction in the yz plane with ϑ = 0 corresponding to the positive y axis. The far-field SPL directivity also behaves like a Gaussian distribution with respect to the angle ϑ on the ring plane as shown in Figure 5.13 a) and the far-field SPL drops sharply when the angle goes far away from ϑ = 0. The peak achieves 133.4 dB and 133.1 dB at ϑ = 0 for CESM and LEE, whereas the valley is under 80 dB at $\vartheta = 210^{o}$. In addition, the directivities of CESM agree well with those of LEE. The maximum difference of SPL, 5.4 dB, between CESM and LEE occurs at $\vartheta = 12.43^{o}$ which is denoted by dash line in Figure 5.13 a). When observed from the ring under the cylinder (from the ground), it is found that the behaviour of the Gaussian distribution still exists with respect to φ. The peaks achieve 99.0 dB and 99.5 dB at $\varphi = 90^{o}$ for CESM and LEE. Furthermore, the SPL of LEE coincides well with that of CESM with the maximum difference of 1.5 dB at $\varphi = 155.1^{o}$ upstream and 0.6 dB at $\varphi = 351.3^{o}$ downstream.

In conclusion, the solution of current acoustic scattering solver, SotonLEE, agrees well with that of CESM, which validates the source model, boundary conditions and numerical methods of LEE. It can and will be utilized to predict the scattering of propeller noise off a cylinder.

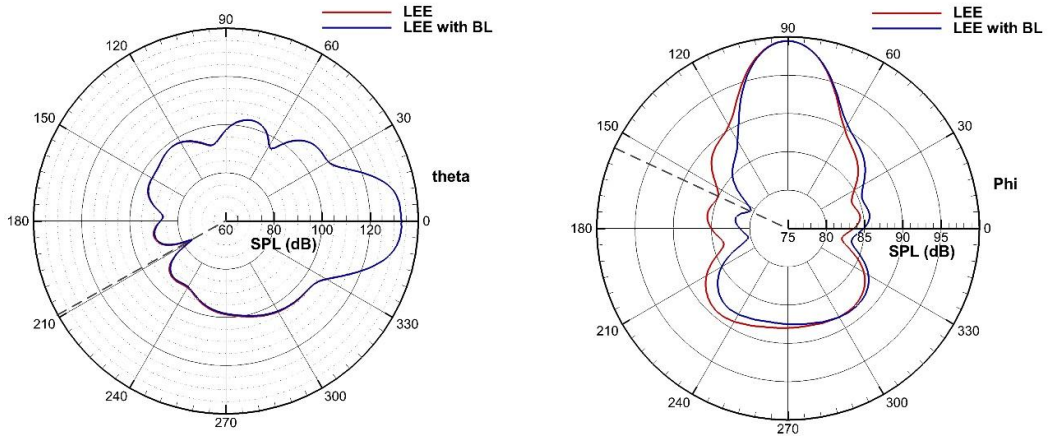
5.2.4.2 Refraction Effect of Boundary Layer

The result of propeller noise scattering off a cylinder with BL is compared with that in uniform mean flow to investigate the refraction effect of boundary layer. Quantitative comparisons of the SPL on the cylinder wall and the far-field directivity are given in Figure 5.14 andFigure 5.15 shown below:



a) SPL on the cylinder wall, φ = 0. b) SPL on the cylinder wall, φ = 1.5π.

Figure 5.14 SPL values on the cylinder wall along the stream-wise direction.



a) SPL directivity at x = 0, r = 5.2. b) SPL directivity at z = -4 m, r = 5.2.

Figure 5.15 Far-field SPL directivities.

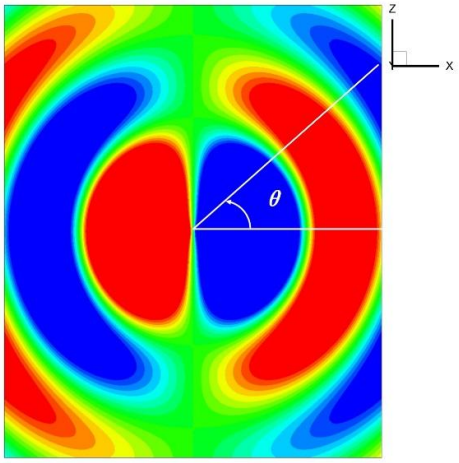
As it is shown in Figure 5.14 a) and b), the boundary layer reduces the SPL on the cylinder wall upstream of the ring plane and increases that downstream. This phenomenon was also demonstrated by results obtained from experiments [113, 117, 118] and analytical methods [113-

117, 119], and now it is recovered by the current CAA methods. In addition, the reduction upstream and increase downstream is stronger at the bottom than at the propeller side since the propagation path through the boundary layer at the bottom is longer. The maximum reduction achieves 12 dB.

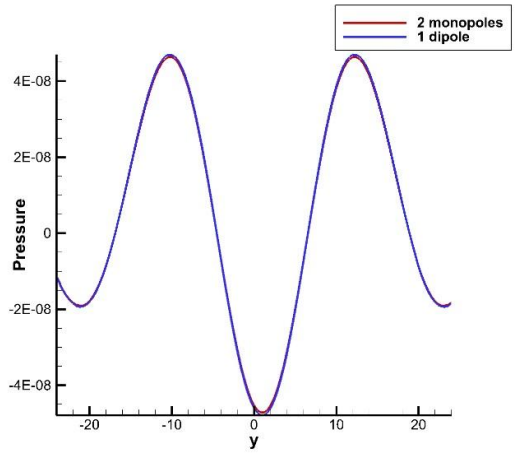
At far-field, the directivities obtained from LEE and LEE with BL coincide well on the ring plane as shown in Figure 5.15 a). The refraction effect of boundary layer can be ignored on the ring plane, which is also demonstrated at x = 0 in Figure 5.14 a) and b). The boundary layer reduces the SPL upstream of the ring plane and increases that downstream on the ground as shown in Figure 5.15 b), which is similar to the change of SPL on the cylinder wall. The maximum reduction achieves 4.5 dB. The causes will be explained in section 5.5.

5.2.5 Results of Loading Noise

The thickness noise is investigated using a ring of spinning monopoles. The loading noise should be investigated using a ring of spinning dipoles. However, direct use of dipoles in SotonLEE is found not stable close to the solid wall in the current solver. It is found that spinning dipoles excite parasite waves at the initialization phase, which propagate across the whole domain and cannot be damped efficiently by the buffer zone in SotonLEE. Consequently, an equivalent way to simulate the dipole noise is introduced in the current solver. Since each dipole is composed of a pair of monopoles [164, 165] out of phase separated by a small distance d with kd << 1, two rings of monopoles [49] are used to simulate a ring of spinning dipoles. First, the replacement of a stationary dipole by a pair of monopoles is validated in SotonLEE. For a stationary dipole with a radiating frequency of 120 Hz, two monopoles out of phase separated by 0.1 m with the same strength and frequency are used. The instantaneous acoustic pressure contours at t = 0.2 s are shown below:



a) Pressure contours radiated by a pair of monopoles at the surface with y = 0.

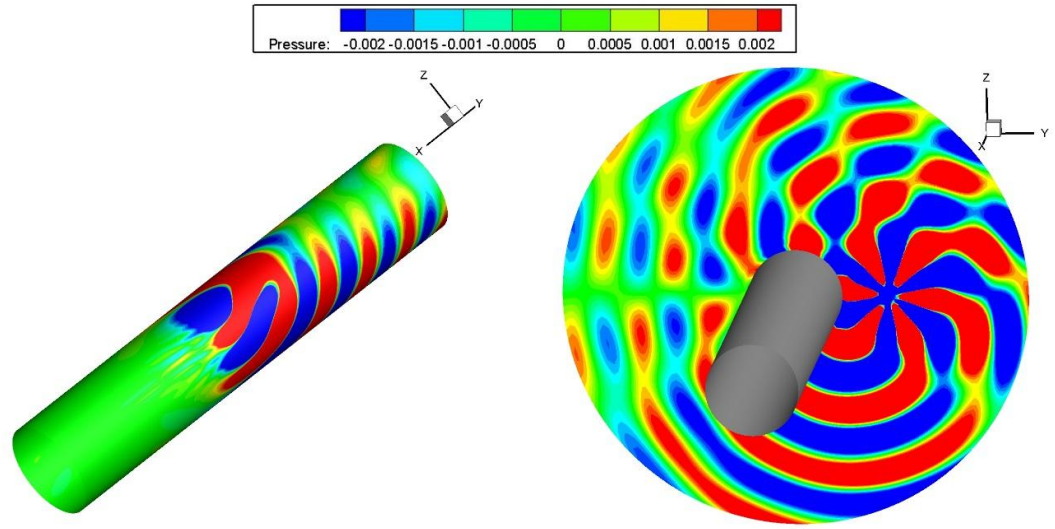


b) Acoustic pressure radiated by a pair of monopoles and a dipole at the line with x = -4, z = 10.

Figure 5.16 Acoustic pressure radiated by a pair of monopoles.

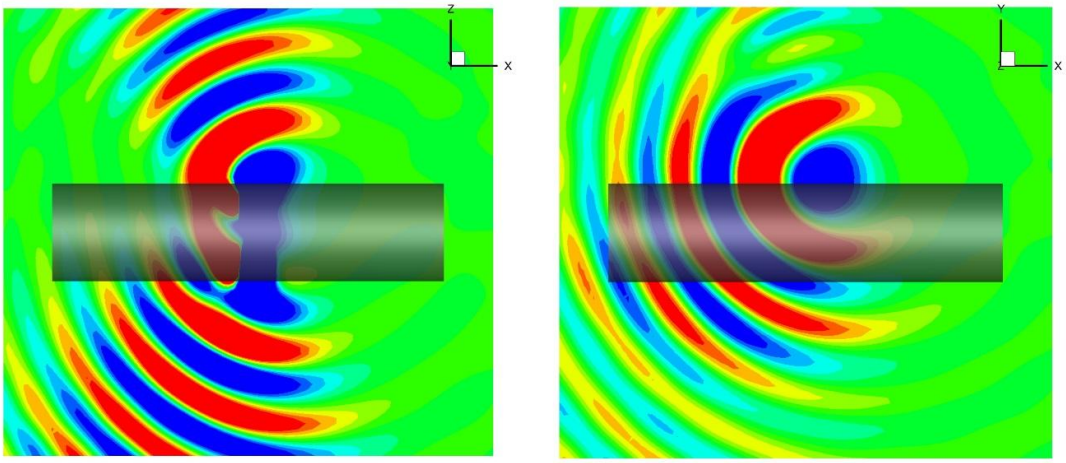
It can be observed from Figure 5.16 a) that the pressure radiated by the pair of monopoles exhibits the same radiation pattern as a dipole. The radiation is strongest at $\vartheta = 0^o$ and $\vartheta = 180^o$, whereas no sound is radiated at $\vartheta = 90^o$ and $\vartheta = 270^o$. ϑ is defined in Figure 5.16 a). The acoustic pressure generated by a dipole and a pair of monopoles at the line with $x = -4$, $z = 10$ from the source shown in Figure 5.16 b) demonstrates the equivalence of the two kinds of sources.

As a dipole is a vector in space, the components of a dipole in the cylindrical coordinate system contain the lift and drag components of the blade section. A dipole with the value of (F_x, F_r, F_ϑ) contains the lift of F_x and the drag of F_ϑ of the blade section. In practice, the contribution of F_r is small in comparison to those of F_x and F_ϑ and therefore is ignored. The values of F_x and F_ϑ can be determined by experiments or numerical simulations. In this study, the ratio of F_x / F_ϑ is set to 4 [166] which is a bit larger than the value in reference [10]. A similar analysis to that of the scattering of the ring of monopoles is applied to the dipole results. A comparison between LEE with uniform flow and LEE with BL is made. The profiles of instantaneous sound pressure field are shown in Figure 5.17, whereas those of SPL are shown in Figure 5.18 below:



a) Acoustic pressure on the cylinder wall.

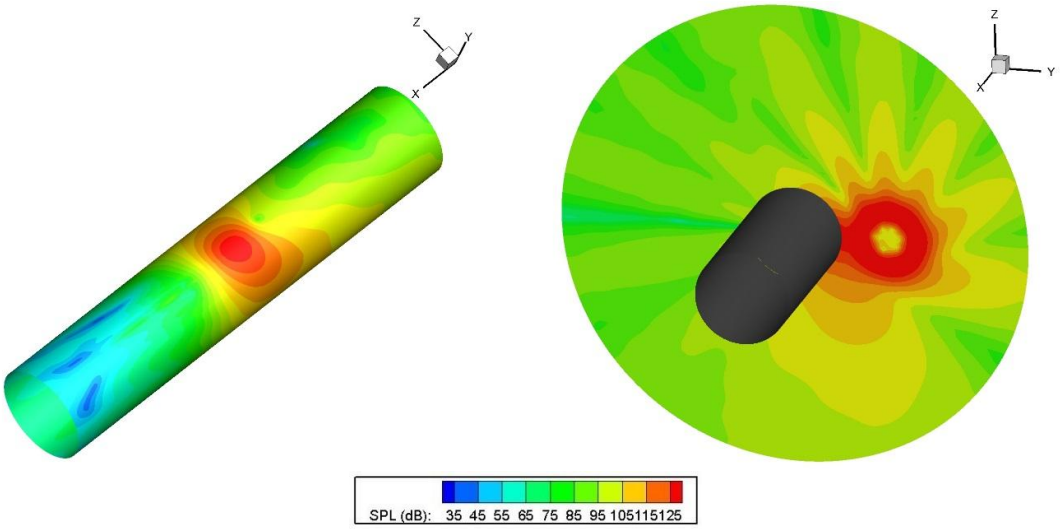
b) Acoustic pressure on the ring plane.



c) Acoustic pressure at plane $y = 4$.

d) Acoustic pressure at plane $z = -4$.

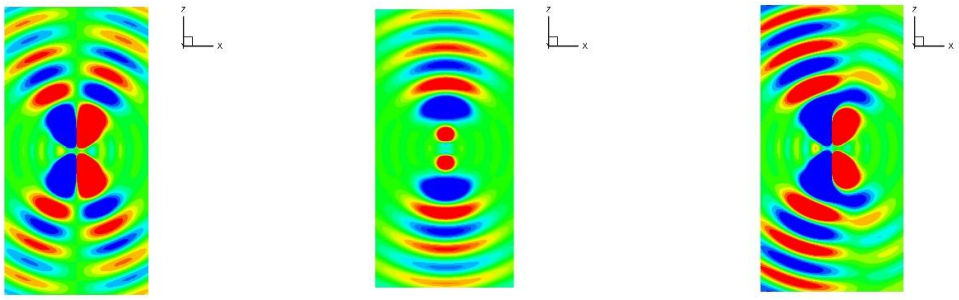
Figure 5.17 Instantaneous pressure contours simulated by LEE in uniform flow.



a) SPL on the cylinder wall. b) SPL on the ring plane.

Figure 5.18 SPL contours simulated by LEE in uniform flow.

The acoustic pressure is propagating spirally in Figure 5.17 b), which illustrates the rotation of the sound sources. It can be observed in Figure 5.17 a) and c) that acoustic pressure radiated by the spinning dipoles mainly propagates in the ring plane and upstream, which is quite different from the propagation of spinning monopoles. The propagation in the ring plane mainly comes from the contribution of the drag of the blade section, whereas the propagation upstream is attributed to the resultant contribution of the thrust normal to the ring plane and the drag. This conclusion is demonstrated by the Figure 5.19 below:



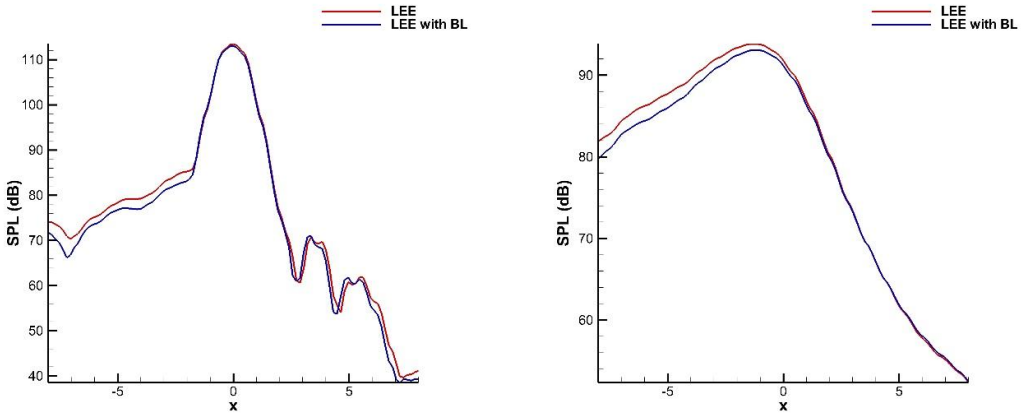
a) Dipole *(F_x, 0, 0)*. b) Dipole *(0, 0, F_ϑ)*. c) Dipole *(F_x, 0, F_ϑ)*.

Figure 5.19 Radiation pattern by three kinds of spinning dipoles.

The contributions of the lift, drag and the resultant forces of a ring of the spinning dipoles are depicted in Figure 5.19 a), b) and c), respectively. The ring of spinning dipoles *(F_x, 0, 0)* radiates the

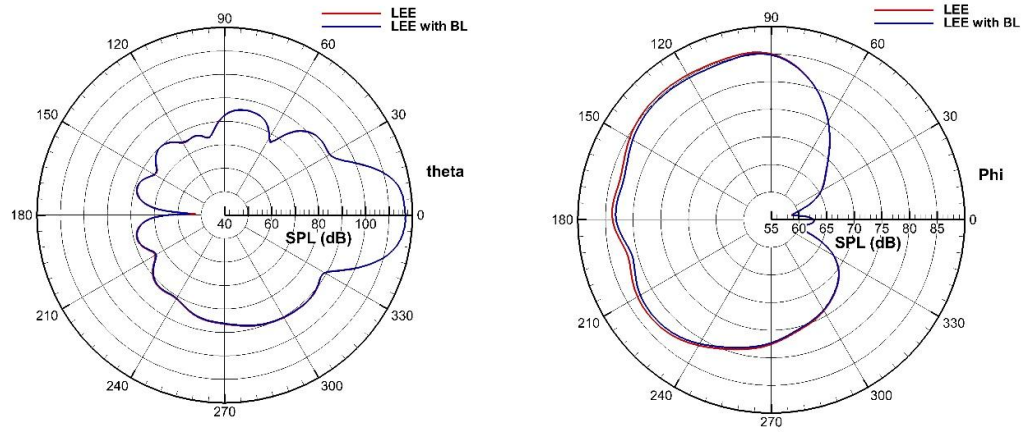
acoustic waves upstream and downstream, whereas the spinning dipoles *(0, 0, F_ϑ)* mainly radiates the acoustic waves on the ring plane. The ring of resultant spinning dipoles *(F_x, 0, F_ϑ)* radiates near and upstream of the ring plane. The acoustic waves radiated by the ring of spinning dipoles *(0, 0, F_ϑ)* enhance the acoustic wave upstream radiated by the ring of spinning dipoles *(F_x, 0, 0)* and weaken that downstream. The result of the ring of spinning dipoles *(F_x, 0, F_ϑ)* is the resultant interference of those of spinning dipoles *(F_x, 0, 0)* and *(0, 0, F_ϑ)*.

The sound pressure contour under the cylinder is shown in Figure 5.17 d). Unlike the monopole case, the radiation of the spinning dipoles to the ground does not concentrate at the RHS of the cylinder but mainly radiates upstream. This is due to the upstream acoustic propagation of spinning dipoles. The pressure and SPL of acoustic waves on the ring plane are shown in Figure 5.17 b) andFigure 5.18 b). It shows a strong directivity on the ring plane and multiple peaks and valleys of SPL distribute due to the scattering from the cylinder. To evaluate the refraction effect of the BL, quantitative comparisons of the SPL on the cylinder wall and far-field directivities are given in Figure 5.20 andFigure 5.21 shown below:



a) SPL on the cylinder wall, ϑ = 0. b) SPL on the cylinder wall, ϑ = 1.5π.

Figure 5.20 SPL values on the cylinder wall along the stream-wise direction.



a) SPL directivity at *x* = 0, *r* = 5.2. b) SPL directivity at *z* = -4 m, *r* = 5.2.

Figure 5.21 Far-field SPL directivities.

On the cylinder wall with $\vartheta = 0$ in Figure 5.20 a), the loading noise also concentrates on the ring plane and behaves like a Gaussian distribution along the stream-wise direction. However, the SPL decreases more slowly upstream than downstream when the observers are the same distance away from the ring plane due to the upstream propagation. The difference between the peaks of the SPL values obtained from LEE and LEE with BL is less than 0.5 dB. When observed on the line $\vartheta = 1.5\pi$, which corresponds to the bottom of the cylinder wall in Fig. 5.18 b), the peaks of SPL are 93.8 dB for LEE and 93.1 dB for LEE with BL. The SPL on the line $\vartheta = 1.5\pi$ drops much more sharply downstream than upstream. The maximum difference of SPL values upstream achieves 1.8 dB on the cylinder wall. In contrast, the difference downstream is small and negligible. The boundary layer reduces the SPL on the cylinder wall upstream, which is consistent to the conclusion obtained in the analytical methods [113-117, 119].

At the far-field as it is shown in Figure 5.21 a) and b), the two same rings sketched in Figure 5.7 are used as observers to compute the SPL. In Figure 5.21 a), the directivity behaves like a Gaussian distribution with respect to ϑ as well. The peak achieves 117.1 dB at $\vartheta = 0$ for both LEE and LEE with BL. The SPL directivities of LEE and LEE with BL coincide, which denotes the boundary layer does not change the SPL directivities on the ring plane. The maximum difference is 0.4 dB.

When observed from the ring under the cylinder (from the ground) in Figure 5.21 b), it is found that the directivity mainly concentrates upstream with a relatively smooth peak region due to the upstream propagation. The maximum peaks of LEE and LEE with BL are 88.5 dB and 88.1 dB respectively. The valleys are below 60 dB. In addition, the directivities of LEE and LEE with BL almost coincide. The refraction effect of the boundary layer reduces the SPL upstream at the far-field by a maximum of 0.8 dB. The maximum difference of SPL downstream is 0.2 dB and can be ignored.

In conclusion, the SPL behaves like a Gaussian distribution on the cylinder wall and concentrates near the ring plane. The refraction effect of the boundary layer on the loading noise on the cylinder wall with low Mach number flow can be ignored. On the ring plane, the boundary layer does not contribute to the SPL directivities, whereas the boundary layer reduces the SPL upstream. However, the reduction is smaller than 2 dB on the cylinder wall and smaller than 1 dB at the far-field upstream. It is assumed to be important when the difference is larger than 3 dB. The refraction effect of boundary layer does not play an important role in the far-field SPL directivity for spinning dipoles.

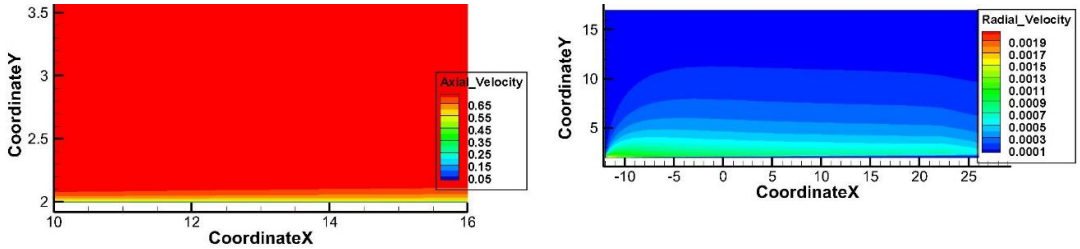
5.3 Cruise Mach Number Setup

The setup of cruise Mach number case is similar to that of low Mach number case. For example, the sound source and the physical domain with boundary conditions are similar. The main

differences between the cruise Mach number and low Mach number cases are the mean flow field, CAA mesh and the properties of the ambient air. The computational setup is introduced in brief below.

5.3.1 Case Setup

The standard air properties at an altitude of 10,668 meters are used in the problem setup. In LEE implementations, the acoustic sources, physical domain together with the boundary conditions remain the same as those in the low Mach number case. The PPW of the mesh is set to 7. Due to the strong convection effect, the size of the mesh upstream along the stream-wise direction is compressed into a quarter of that in the low Mach number case, whereas the size of mesh downstream is stretched into 1.75 times in the flow direction. The mesh sizes in the radial and azimuthal directions are kept the same. The total amount of mesh points is 2.9 million. Non-dimensional variables are used in the computation. The reference length scale L is 1 m, speed scale, c_∞, is 296.6 m/s, and density scale, ρ_∞ is 0.38 kg/m^3. The Mach number of free stream is 0.75. The mean flow field obtained from RANS is shown below:



a) Mach number of axial velocity. b) Mach number of radial velocity.

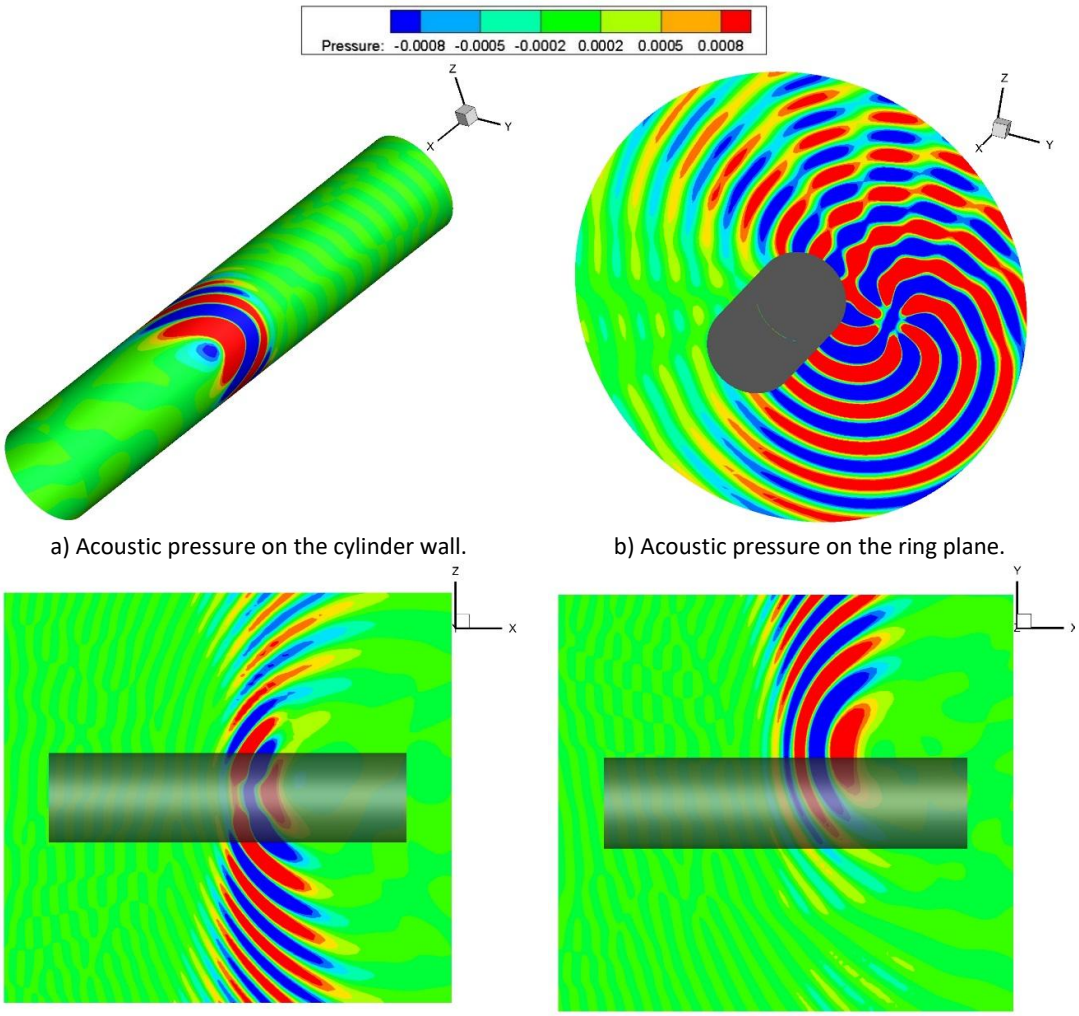
Figure 5.22 Mean flow field obtained from RANS.

The estimated 99% turbulence boundary layer thickness on the end of cylinder surface is 0.186 m, which is thin in comparison to the acoustic wavelength of 2.47 m.

5.3.2 Results of Thickness Noise

The profiles of instantaneous sound pressure field are shown in Figure 5.23 and those of SPL are shown in Figure 5.24. The spiral propagation of sound pressure illustrates the rotation of the sound sources as shown in Figure 5.23 b). In comparison to the low Mach number case, as shown in Figure 5.10 c), the acoustic wavelength upstream is shortened significantly in Figure 5.23 c) due to the strong convection effect of the mean flow-field. The acoustic waves also concentrate near the ring plane and at the RHS of the cylinder where the propeller is located. The concentration of acoustic wave propagation near the ring plane is also demonstrated by Figure 5.23 a) andFigure 5.24 a). The

SPL contour on the ring plane is shown in Figure 5.24 b). It shows a strong directivity on the ring plane and multiple peaks and valleys of SPL distribute due to the reflection of the cylinder. Quantitative comparisons of the SPL on the cylinder wall and far-field directivity are given to evaluate the refraction effect, as shown in Figure 5.25 below:



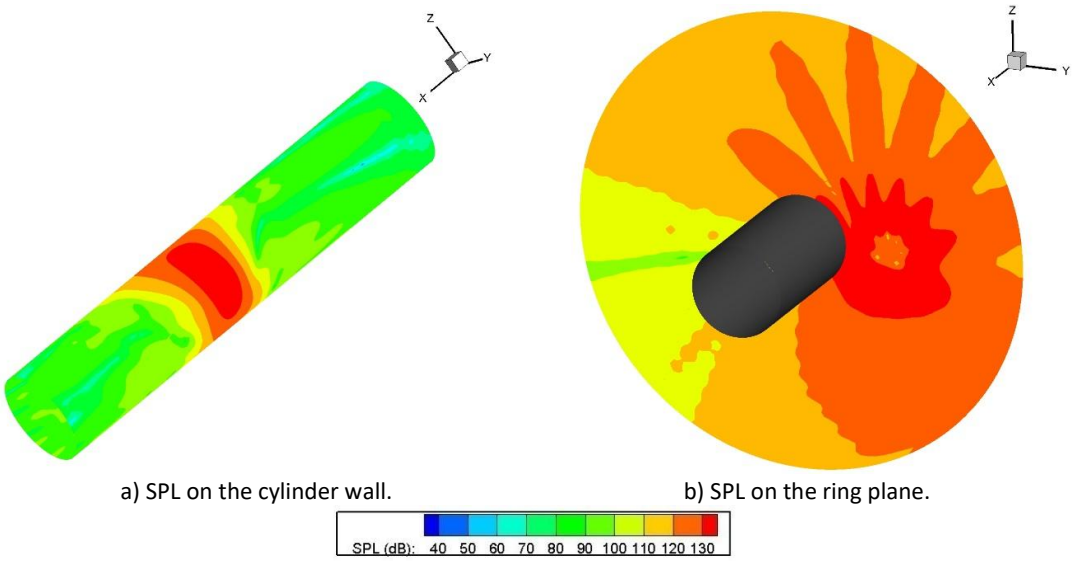
a) Acoustic pressure on the cylinder wall.

b) Acoustic pressure on the ring plane.

c) Acoustic pressure on the plane with y = 4.

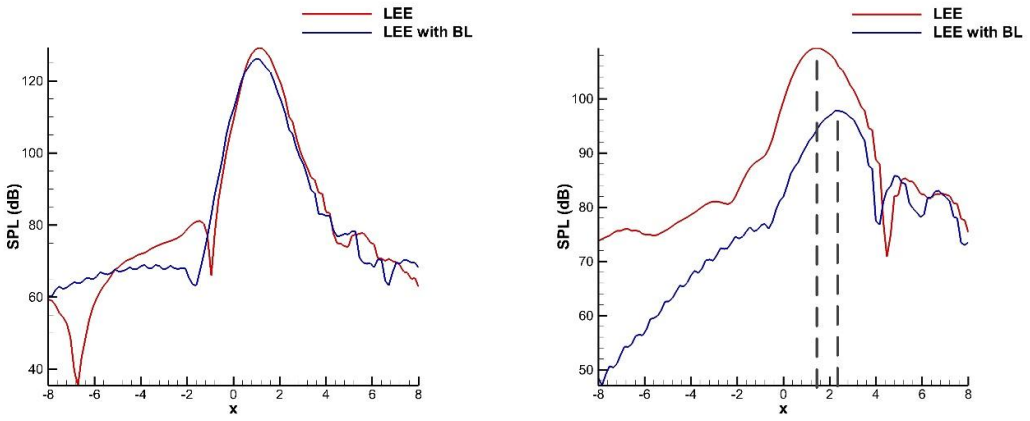
d) Acoustic pressure on the plane with z = -4.

Figure 5.23 Instantaneous pressure contours simulated by LEE in uniform flow.



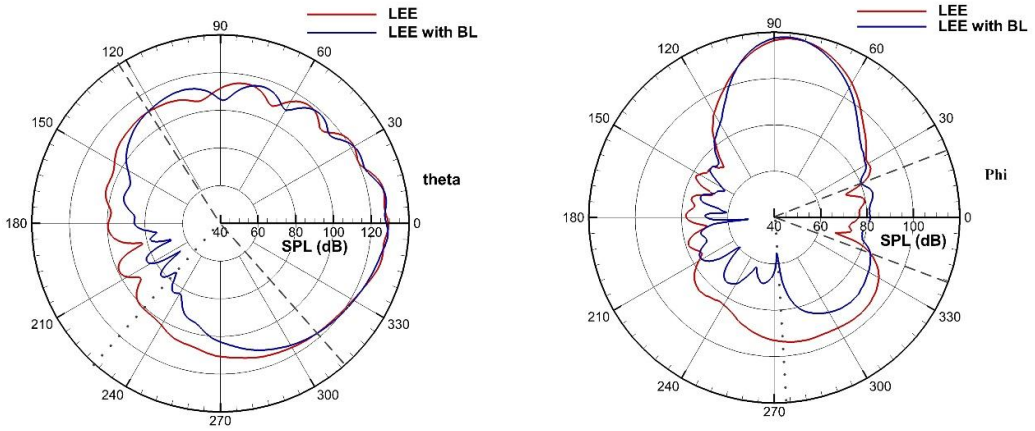
a) SPL on the cylinder wall. b) SPL on the ring plane.

Figure 5.24 SPL contours simulated by LEE in uniform flow.



a) SPL on the cylinder wall, $\vartheta = 0$. b) SPL on the cylinder wall, $\vartheta = 1.5\pi$.

Figure 5.25 SPL on the cylinder wall.



a) SPL directivity at $x = 1$, $r = 5.2$. b) SPL directivity at $z = -4$, $r = 5.2$.

Figure 5.26 Far-field SPL directivities.

The SPL still concentrates on the ring plane and behaves like a Gaussian distribution along the stream-wise direction as shown in Figure 5.25 a), which is the same as that in the low Mach number case. However, the difference of SPL between the peaks is larger, 3.0 dB. The peaks are 129.0 dB and 126.1 dB respectively at x = 1.0 for LEE and LEE with BL. The SPL drops sharply when the observer is far away from the ring plane. The refraction effect of boundary layer increases SPL on the cylinder wall upstream and decreases SPL downstream, which is consistent to the results obtained in analytical methods [113-117, 119]. In comparison to the low Mach number case, the refraction effect of boundary layer in the cruise Mach number case is much stronger. When observed from the line with $\vartheta = 1.5\pi$ at the bottom of the cylinder shown in Figure 5.25 b), the maximum SPL in LEE and LEE with BL is 109.3 dB and 97.8 dB occurring at x = 1.44 and 2.23 respectively. The peak is moved downstream by the strong convection of the mean flow field [113-117, 119]. When the boundary layer is considered, the SPL in LEE with BL is roughly 15.0 dB less than that in LEE upstream and the maximum difference of 10.1 dB occurs downstream. The refraction effect of boundary layer is more significant upstream than downstream.

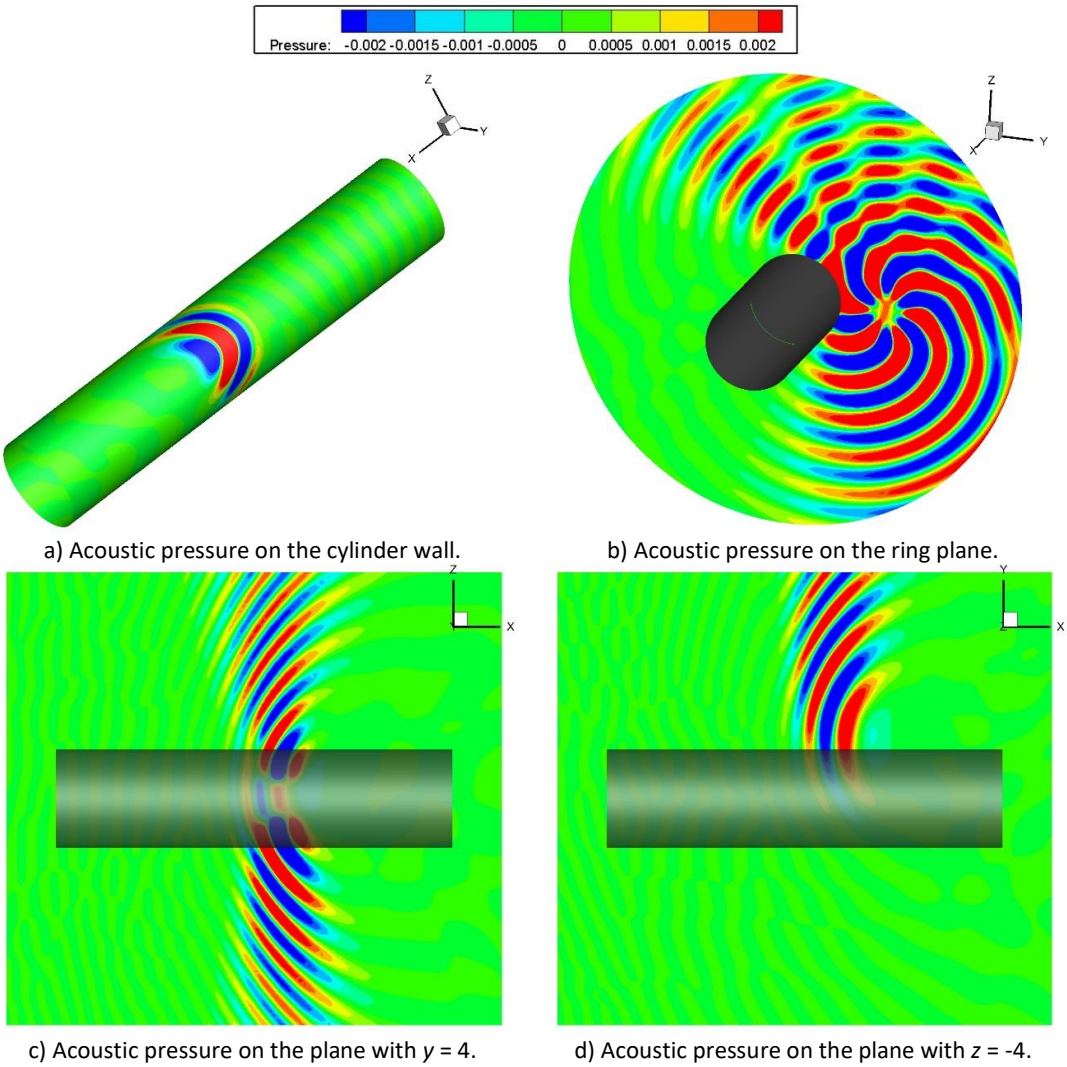
At the far-field, the same rings employed in the low Mach number case as shown in Figure 5.7 are used to monitor the directivities as shown in Figure 5.26. The boundary layer mainly reduces SPL by more than 20 dB in the region where the valleys distribute. In the region with $123.3^o < \vartheta < 310.6^o$ which is denoted by the dash line, the boundary layer always reduces the far-field SPL. The maximum reduction is over 25 dB. When observed from the ring with z = -4 under the cylinder (from the ground), the boundary layer reduces the SPL in most of regions. The reduction achieves 20 dB ~ 30 dB in regions where the SPL valleys distribute. In addition, the boundary layer only increases SPL downstream in a small region with $-20^o < \varphi < 21^o$ which is denoted by the dash line. The boundary layer reduces the SPL upstream by about 8 dB and increases the SPL downstream by roughly 5 dB.

In conclusion, the boundary layer mainly reduces significantly the SPL where the SPL valleys distribute since the propagation path is longer. In addition, the SPL is increased upstream and is reduced downstream at the far-field by the boundary layer. The refraction effect of the boundary layer plays an important role in the cruise Mach number case.

5.3.3 Results of Loading Noise

The similar analysis to that of the scattering of the ring of monopoles is applied to the dipole results. The profiles of the instantaneous sound pressure field are first shown in Figure 5.27 whereas those of SPL are shown in Figure 5.28. The spiral propagation of sound pressure is shown in Figure 5.27 b). In comparison to the low Mach number case as shown in Figure 5.17 a) and c), the acoustic

pressure radiated by the spinning dipoles travelling in the flow field with a cruise Mach number mainly concentrates near the ring plane. The acoustic wavelength upstream is shortened significantly. The concentration is also illustrated by Figure 5.28 a). Another noticeable difference is that the acoustic wave concentrates at the RHS in the flow field with a cruise Mach number as shown in Figure 5.28 d) rather than the upstream direction in the low Mach number case. This phenomenon is similar to that of the monopole case in high Mach number due to the strong convection effect of the mean flow-field. The pressure and SPL of acoustic waves on the ring plane are shown in Figure 5.27 b) and 5.28 b). A strong directivity on the ring plane occurs again and multiple peaks and valleys of SPL distribute, which is consistent to that in the low Mach number case. Quantitative comparisons of the SPL on the cylinder wall and far-field directivity are given to evaluate the refraction effect of the boundary layer, as shown in Figure 5.29 andFigure 5.30 below:



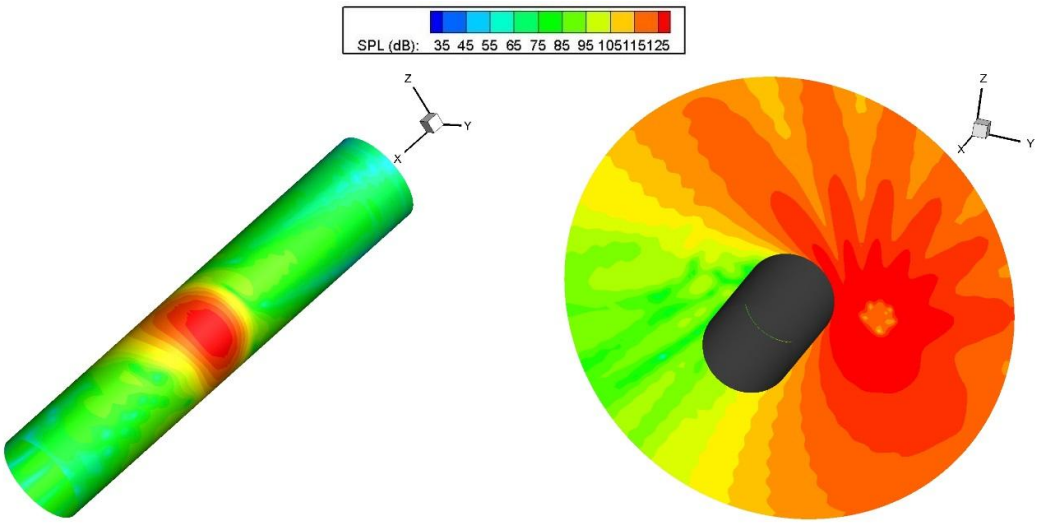
a) Acoustic pressure on the cylinder wall.

b) Acoustic pressure on the ring plane.

c) Acoustic pressure on the plane with $y = 4$.

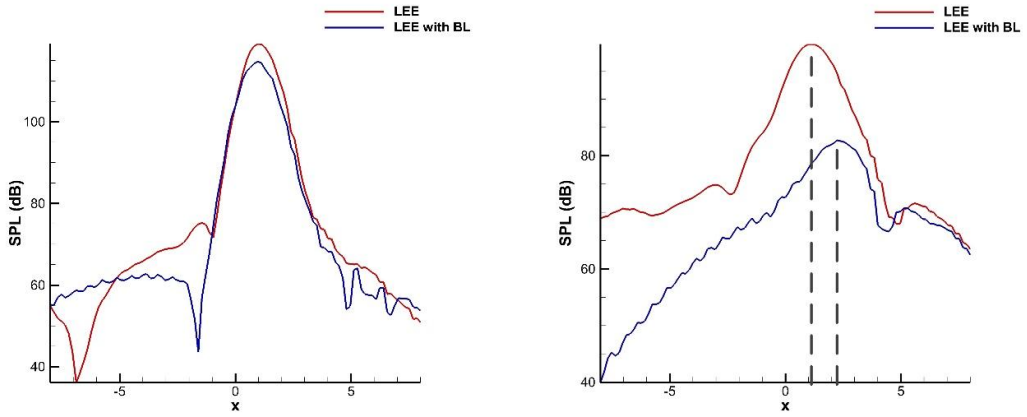
d) Acoustic pressure on the plane with $z = -4$.

Figure 5.27 Instantaneous pressure contours simulated by LEE in uniform flow.



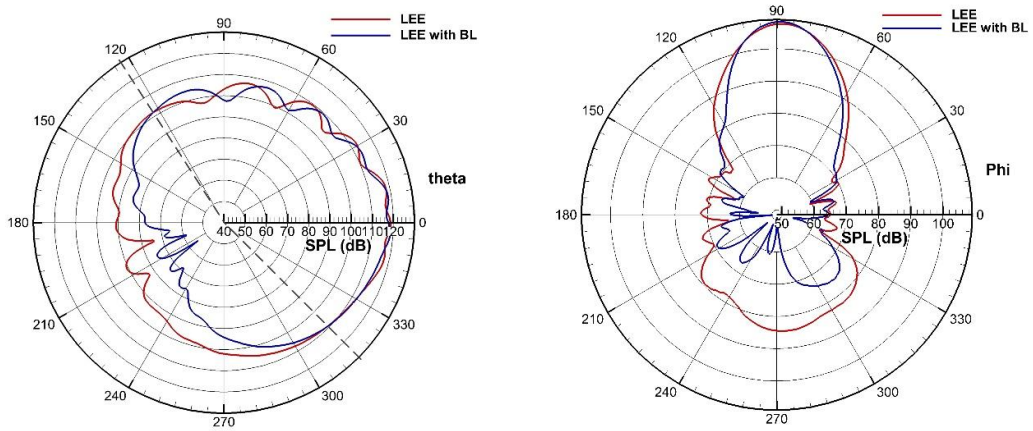
a) SPL on the cylinder wall. b) SPL on the ring plane.

Figure 5.28 SPL contours simulated by LEE in uniform flow.



a) SPL on the cylinder wall, $\vartheta = 0$. b) SPL on the cylinder wall, $\vartheta = 1.5\pi$.

Figure 5.29 SPL on the cylinder wall.



a) SPL directivity at $x = 1$, $r = 5.2$. b) SPL directivity at $z = -4$, $r = 5.2$.

Figure 5.30 Far-field SPL directivities.

The SPL peak obtained from LEE with BL is 4.4 dB less than that obtained from LEE as shown in Figure 5.29 a). When observed from the line $\vartheta = 1.5\pi$ at the bottom of the cylinder from Figure 5.29 b), the maximum values in LEE and LEE with BL are 99.7 dB and 82.7 dB which occur at $x = 1.11$, and 2.26 respectively. The SPL peak simulated by LEE with BL moves downstream. When the boundary layer is considered, the SPL in LEE with BL is roughly 23.1 dB less than that in LEE upstream at $x = 0.63$ and the maximum difference of 11.0 dB occurs at $x = 1.11$ downstream. The boundary layer significantly reduces the SPL both upstream and downstream on the cylinder wall. The refraction effect of boundary layer is more significant upstream than downstream.

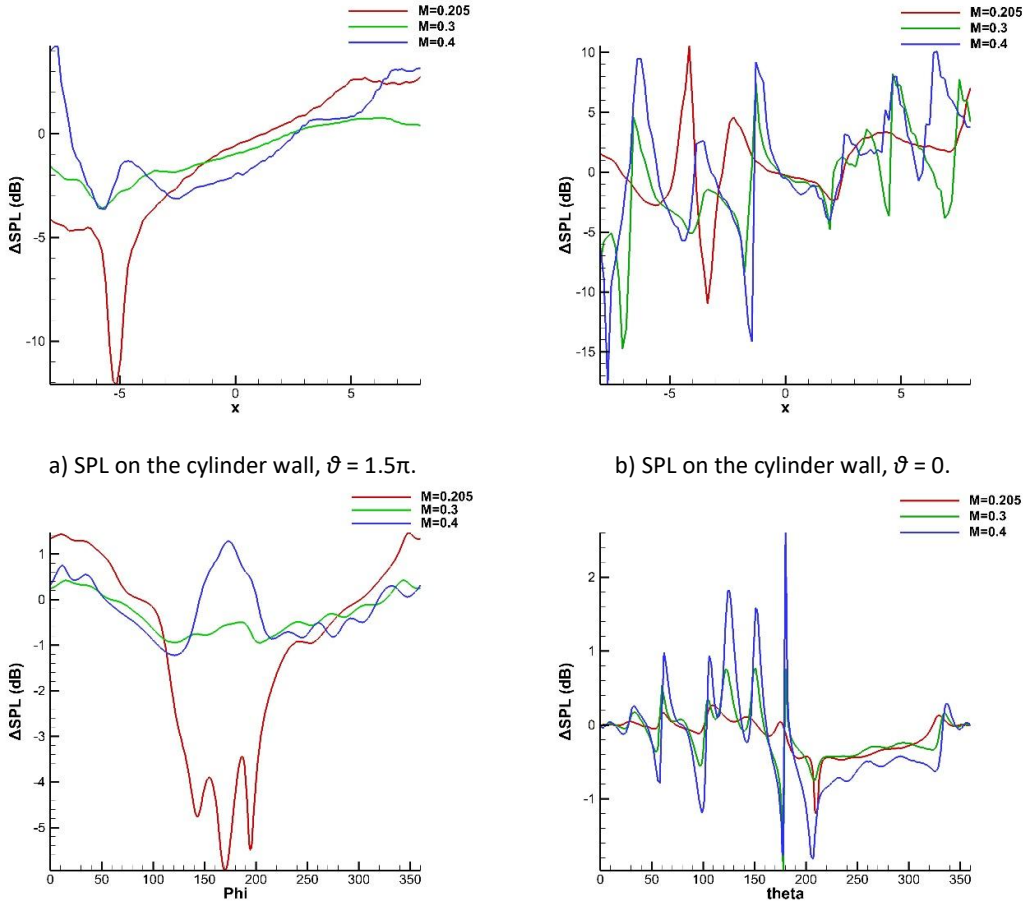
At far-field, the two same rings employed in the low Mach number case sketched in Fig. 5.7 are used as observers to compute the SPL directivities as shown in Figure 5.30 a). The SPL in LEE in the region with $122.6^o < \vartheta < 314.5^o$ which is denoted by the dash line is always larger than that in LEE with BL. The maximum difference over 25 dB is observed on the directivity between LEE and LEE with BL at the valley for LEE with BL at $\vartheta = 215.6^o$. Out of this region, the SPL in LEE is comparable to that for LEE with BL. The boundary layer reduces the SPL significantly in the region where the valleys distribute. When observed from the ring with $z = -4$ under the cylinder (from the ground) in Figure 5.30 b), it is found that the SPL mainly concentrates at the RHS of the cylinder where the propeller is located, which is quite different from the low Mach number case. The maximum difference of SPL between LEE and LEE with BL is 34.5 dB at $\varphi = 270.0^o$. The boundary layer reduces the directivity at the far-field in the region opposite to the propeller side whereas the SPL of LEE with BL is comparable to that of LEE at the propeller side in Figure 5.30 b).

In conclusion, the boundary layer mainly reduces the SPL significantly where the SPL valleys distribute. In addition, the SPL upstream is decreased significantly both on the cylinder wall and at the far-field by boundary layer. The refraction effect of boundary layer plays an important role in the cruise Mach number case.

5.4 Effect of Mach Numbers on Refraction Effect of Boundary Layer

It is of interest to explore trends in refraction effect of BL with variations in flight speed. As shown in Sections 5.2, the refraction effect is weak at $M = 0.205$ whereas it is strong at $M = 0.75$. An interval of Mach number exists between 0.205 and 0.75. To explore at which Mach number the refraction effect starts to be negligible, the low Mach number case is set up at $M = 0.3$ and $M = 0.4$ respectively. Mach numbers above 0.4 are considered to belong to the cruise Mach numbers. At cruise Mach numbers, boundary layer is significant according to the research performed by Spence [113]. At $M = 0.3$ and $M = 0.4$, the same source model, boundary conditions and properties of ambient air as those in the low Mach number case are kept whereas different CAA mesh and background mean

flow-field are used. The acoustic wavelength decreases upstream due to the convection effect of the mean flow-field while the wavelength increases downstream. Consequently, the mesh is adjusted to keep the PPW upstream and downstream of the ring source still 7. Due to the reflection of BL and the fuselage, a strong directivity exists, which is different from the free-field. In addition, the convection of the mean flow-field at different Mach numbers causes different distribution of the SPL lobes on the cylinder wall and directivities at far-field. A direct comparison of directivities between different Mach numbers does not show the refraction effect of boundary layer directly. Consequently, a comparison of the difference between the SPL on the cylinder wall and directivities at far-field at different Mach numbers are shown. The differences of SPL on the solid wall and at the far-field by boundary layer are shown below:



a) SPL on the cylinder wall, ϑ = 1.5π.

b) SPL on the cylinder wall, ϑ = 0.

c) SPL directivity at z = -4, r = 5.2.

d) SPL directivity at x = 0, r = 5.2.

Figure 5.31 Difference of SPL caused by boundary layer.

When boundary layer is considered, the reduction of SPL on the cylinder wall occurs near the ring plane at x = 0 and upstream as shown in Figure 5.31 a). The attenuation of SPL is strongest at M = 0.4 near the ring plane whereas the attenuation is stronger at M = 0.205 when the distance is far

away upstream of the ring plane. The maximum reduction is over 3 dB for M = 0.4. At x = -5, a strong reduction of SPL appears on the cylinder wall for M = 0.205. The cause is attributed to the different distribution of the SPL lobes on the cylinder wall. As explained by Hanson [117] and Belyaev [167], an explanation is that the boundary layer changes the directivity of the source. Due to the existence of cylinder, the propeller source has a directivity in space which is different from that in free space. The directivity in free space is homogeneous in the ring plane. The propagation in the boundary layer results in a change in directivity as illustrated in Figure 5.32:

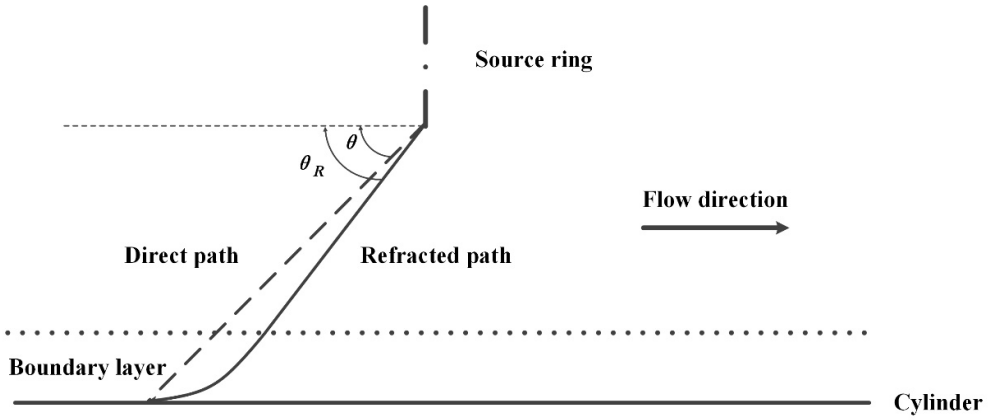


Figure 5.32 Acoustic wave propagation through boundary layer.

In the absence of boundary layer, the acoustic wave propagates to the cylinder in a straight line through the direct path whereas the refracted path is followed when the boundary layer exists. The angle of the direct path ϑ, is different from that of refracted path ϑ_R. When the Mach number increases, the velocity gap between the boundary layer and free field becomes larger. Consequently, the difference between ϑ and ϑ_R becomes larger. In addition, peaks and valleys of the source directivity exists in the streamwise direction. When the refraction effect is strong, the positions of peaks and valleys are moved in comparison to those in the absence of boundary layer. Different lobes of SPL occur on the same locations, which results in a different change of directivity depending on Mach numbers.

The changes of the source directivity by the boundary layer is also illustrated in Figure 5.31 b). A shift of the directivity by the boundary layer along the streamwise direction is significant on the cylinder wall and the strongest change happens when M = 0.4. The shifts of directivity on the cylinder wall for M = 0.3 and 0.4 are similar. The changes are quite different from those for M = 0.205. The different pattern of directivity change between M = 0.205 and M = 0.3, 0.4 is illustrated by Figure 5.31 c) and d) at the far-field. It can be presumed that the refraction effect of the boundary layer is stronger if the Mach number is higher. An explanation is that when the Mach number is

above 0.3, a strong change on the source directivity by the boundary layer happens in comparison to $M \leq 0.205$, which causes a different reduction of SPL by the boundary layer. Different lobes of directivity occur at the same positions and observers at the far-field. At $M = 0.3$ and 0.4, the lobes of directivity changed by the boundary layer at the far-field are distributed in a similar manner, which is different from those at $M = 0.205$. Therefore, it can be concluded that the refraction effect of boundary layer plays an important role when the Mach number is higher than 0.3. When the Mach number is above 0.3, the source directivity is significantly changed by the boundary layer in comparison to that in the absence of the boundary layer.

5.5 Performance

In these cases, the performance running on two Tesla K20M GPUs is compared to those implemented on four cores and on ten cores of Intel Xeon E5-2670 CPU with 2.6 GHz. Due to the size of mesh blocks and loading balance, the maximum available and appropriate amount of MPI process is 10. The performance is summarized below:

Table 5-1 Wall-clock time comparisons in seconds.

Case name	Mesh Size	Wall-clock time (in seconds)		
		2 GPUs	4 CPU cores with speedup	10 CPU cores with speedup
Low Mach monopoles	1.15 M	508.2	5435.1 (21)	2490.1 (24.5)
Low Mach dipoles	1.15 M	534.8	6257.2 (23)	2781.0 (26)
Cruise Mach monopoles	2.87 M	1266.4	18046.2 (29)	8021.2 (32)
Cruise Mach dipoles	2.87 M	1392.2	20187.1 (29)	9033.8 (32)

Note that values in the parentheses denote the speed-ups.

In conclusion, 9 minutes are necessary for the cylinder cases on two GPUs at the low Mach number, whereas 24 minutes must be spent at the cruise Mach number since the total mesh size is larger. A speed-up of about 21~26 is achieved in the low Mach number cases running on two GPUs whereas the speed-up is roughly 30 in the cruise Mach number cases.

5.6 Summary

In this section, the acoustic scattering of a ring of spinning monopoles and dipoles off a cylinder has been investigated by using the CAA method for the first time. A simplified geometry was used to verify the numerical methods, acoustic source models and investigate acoustic refraction effect of the boundary layer in this chapter. A single cylinder is utilized as an ideal fuselage. The propeller noise sources are approximated by a ring of monopoles and a ring of dipoles respectively to investigate the thickness noise and the loading noise.

When compared to the analytical solution in free space, it was found that the PPW of the mesh at the acoustic source region plays a key role in the amplitude of the acoustic propagation. The PPW of the mesh at the source region must meet the requirement of PPW of the numerical schemes to mitigate dispersion error. The amplitude of the acoustic wave must be modulated to be calibrated to the analytical solution. After the calibration, the numerical solution corresponds well with the analytical solution.

Before CAA simulations, the contribution of each harmonic can be identified by the analytical solution. Consequently, the highest order of the harmonic that should be resolved can be identified. The mesh size in the numerical simulation can be determined basically. A validation was firstly performed between the results obtained by LEE and those from CESM. The good agreement between the results validated the source model, numerical methods in SotonLEE and PPW of the mesh.

On the cylinder wall, both the thickness noise and loading noise behave like a Gaussian distribution along the stream-wise direction. The propagation of thickness noise concentrates on the ring plane whereas that of loading noise inclines upstream. The incline upstream of the propagation of loading noise is attributed to the resultant lift and torque of the propeller. At far-field, the directivity contains many peaks and valleys due to the reflection of the cylinder wall. The directivity on the ring plane and on the ground behaves like a Gaussian distribution with respect to the observer position with the peak occurring at the same side as the propeller.

At low Mach number case, the refraction effect of the boundary layer was found negligible whereas the refraction effect is significant at Cruise Mach number computationally. The boundary layer reduces the SPL on the cylinder and far-field directivity upstream of the ring plane. Extensions of computation at M = 0.3 and 0.4 have been performed to determine at which Mach number the refraction effect of boundary layer becomes important. The difference between results in the absence of boundary layer and with a realistic boundary layer showed that the refraction effect of the boundary layer starts to be important when Mach number is larger than 0.3.

On performance, 9 minutes are necessary for the cylinder cases on two GPUs at the low Mach number, whereas 24 minutes must be spent at the cruise Mach number since the total mesh size is larger. A speed-up of between 21 and 26 is achieved in the low Mach number cases running on two GPUs whereas the speed-up is roughly 30 in the cruise Mach number cases.

Chapter 6: Scattering of Propeller Noise off a Wing body

In chapter 5, the acoustic scattering of a ring of spinning monopoles and spinning dipoles off a cylinder was investigated. The CAA methods used in the current solver has been validated and verified. In addition, the refraction effect of the boundary layer was investigated computationally. Recall that the geometries of the cases in Chapters 4 and 5 are relatively simple and the computational scale is small. In the following chapter, the case is extended with a complex geometry, an ATR-72-like wing body with a full scale. The research focus is not placed on the analysis and explanation on physical phenomena since the engineering geometry is complex and the analysis is difficult. The numerical methods and sound sources in the following chapter are kept the same as those in Chapter 5. The current case with a complex geometry is mainly devoted to demonstrate the application to a large-scale engineering case and evaluate the true performance of the current solver in engineering problems. In addition, the weak refraction effect of the non-uniform flow field at M = 0.205 around the wing body is demonstrated computationally. Lastly, the speed-up is recorded to evaluate the true speed-up of the current solver in engineering problems. The causes of the relatively low performance of the current solver in this case with complex geometry in comparison to the cases in Chapters 4 and 5 is also analysed in detail.

6.1 Sound Source

In this case, both propellers rotate in the counter clock-wise direction as viewed by an observer downstream. The sound sources are two rings of six spinning monopoles and dipoles distributed evenly as shown in Figure 6.1, which are similar to those in the cylinder case in chapter 5. The radius of the ring is 1.965 m and the distance between the centres of the two rings is 8.1 m [104-106]. Viewed by an observer downstream, both rings rotate in the counter-clockwise direction at 1200 RPM. The first harmonic of BPF is 120 Hz. The phases of all the monopoles and dipoles are set to 0 since only the steady noise source is considered [10, 49]. The non-dimensional amplitude is set to 0.01 to ensure a linear propagation. The ratio of the lift to drag of the blade section is assumed to be 4 in the dipole case [10, 166].

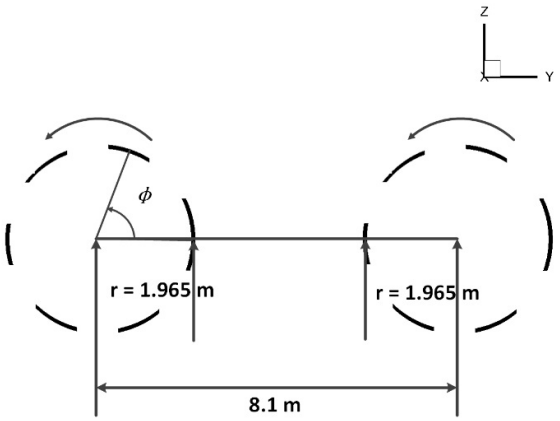


Figure 6.1 Sketch of the spinning sources.

6.2 Analytical Solution

As in the cylinder case in chapter 5, the analytical solution in free space is first computed to determine which BPFs must be resolved. The PSDs of the first three harmonics of the BPF for a single propeller in the free space are shown in Figure 6.2 below:

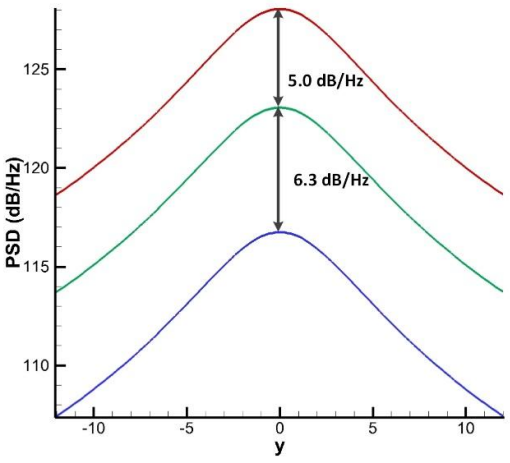


Figure 6.2 PSDs of a single ring of spinning monopoles along the line at *x* = 0, *z* = 5 in free space. The centre of the ring is placed at (0, 0, 0).

It can be observed that the difference of PSDs between the acoustic harmonics of the first BPF and the second BPF is 5.0 dB/Hz, whereas the difference between the second BPF and the third BPF is 6.3 dB/Hz. Therefore, the contribution from harmonics which have a higher frequency than the second BPF is ignored. In other words, the mesh in this setup resolves the acoustic waves with a frequency up to the second BPF.

In this chapter, the main effort does not focus on the comparison between the solution from the uniform mean flow-field and that from RANS mean flow-field. However, it focuses on the demonstration of application of SotonLEE_GPU to the complex geometry, large-scale computation and calculation of the speed-ups.

6.3 Geometry

The investigated configuration is a generic turbo-prop aircraft that has a similar geometry to the ATR-72 500 wing body [104-106]. The aircraft is a simplification of a real aircraft at full scale. The geometries of high-lift devices, engines, horizontal tail and vertical tail are not included. The geometry is used to demonstrate the application of SotonLEE_GPU to real engineering cases. No experimental results are available for comparison. The geometry is depicted below in Figure 6.3:

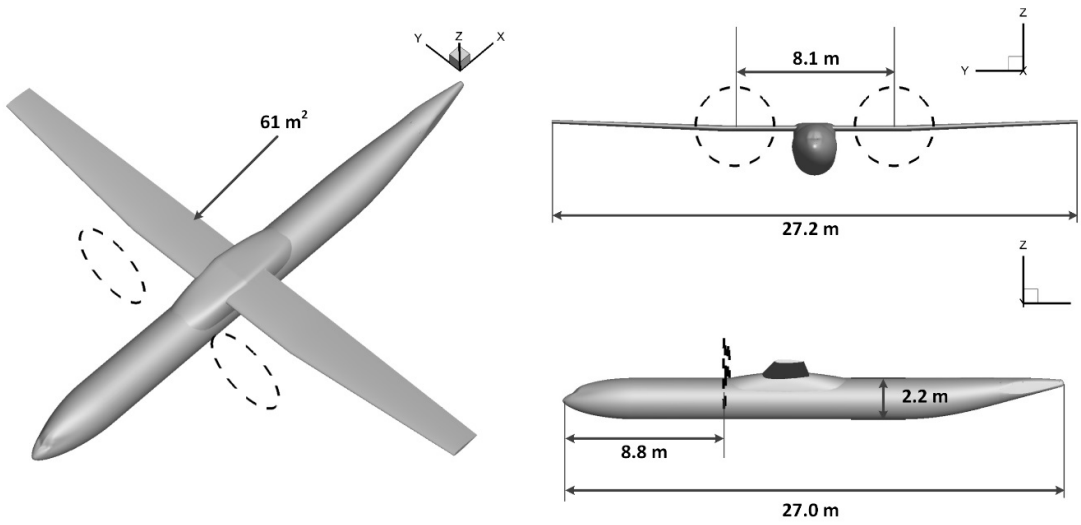
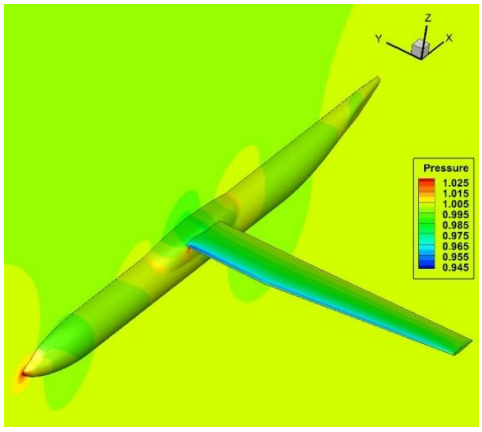


Figure 6.3 Dimensions of the wing-body.

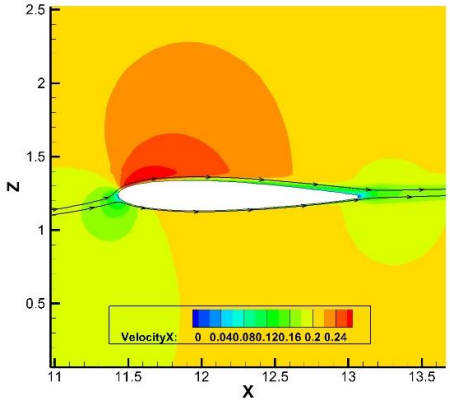
As shown in Figure 6.3, the wing span is 27.2 m with a wing area of 61 m^2. The propeller is 8.8 m away from the fuselage nose. The fuselage is 27.0 m long with a maximum diameter of 2.2 m.

6.4 Mean Flow

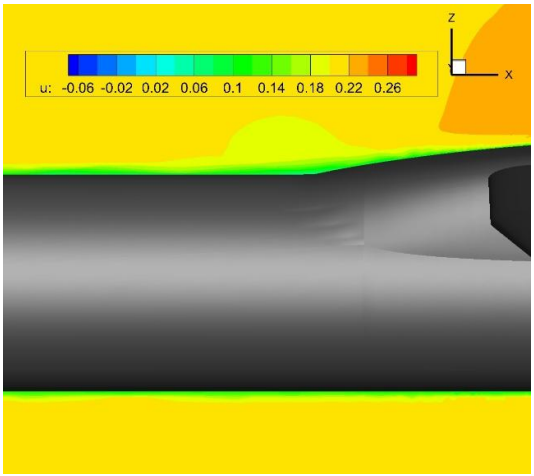
This study focuses on the initial climbing of flight. The aircraft climbs at a Mach number of 0.205, at the sea level, with an angle of attack of 5° of the fuselage. The background mean flow field is obtained by a RANS simulation. The estimated 99% turbulence boundary layer thickness at the aft of the fuselage is 0.25 m, which is quite thin in comparison to the acoustic wavelengths of 2.83 m of the first harmonic and 1.42 m of the second harmonic. The mean flow field is shown in Figure 6.4 below:



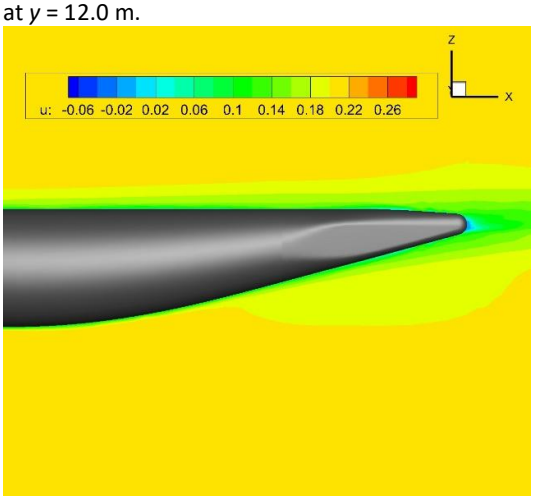
a) Pressure contours.



b) Mach number in the *x* direction at wing section at *y* = 12.0 m.



c) Mach number in the *x* direction near the ring plane.



d) Mach number in the *x* direction near the aft of the fuselage.

Figure 6.4 Mean flow field around the wing body.

6.5 Computation Setup

The physical domain is a box with the dimension of (40, 40, 26) m in the *x*, *y* and *z* directions. On the solid wall, an inviscid wall boundary condition is used, while the stretched mesh is employed to

attenuate the acoustic waves at the far-field. The dimension of the CAA domain is shown below in Figure 6.5:

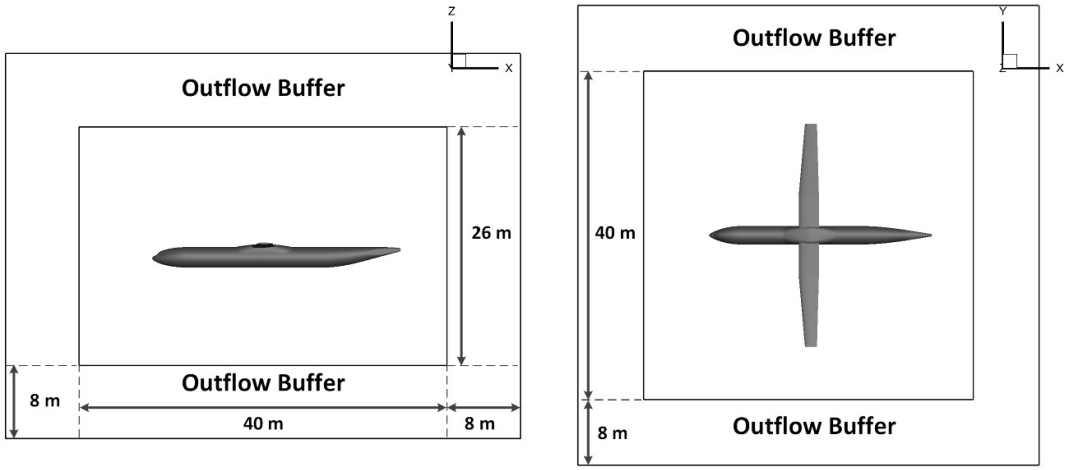


Figure 6.5 Dimensions of physical domain.

At far-field, four rings of observers are used to show the SPL directivities as shown in Figure 6.6. Two rings are located on the ring plane with a radius of 10 m and 1000 m respectively with the origin at (8.8, 0, 0.94) m, which is the middle of two propeller centres. Two other rings are placed under the aircraft with a radius of 10 m and the origin at (8.8, 0, -9.06) m and with a radius of 1000 m and the origin at (8.8, 0, -999.06) m respectively. The SPL on the two rings, with radii of 10 m, is computed directly by LEE during the simulation, while the SPL on the two rings with radii of 1000 m is obtained by using an FW-H solver [100, 161]. The box in grey is the FW-H integration surface, which is used to collect the information of perturbations. The dimension of the box is (28, 28, 8) m. The dimension of the FW-H integration surface is as small as possible to reduce the mesh amount on the control surface and computational cost but sufficiently large to cover the whole geometry and all the acoustic sources at the same time. In this research, the dimension of the FW-H control surface cannot be reduced to the minimum extent by placing it on the solid wall because the acoustic sources of propellers are in the flow field. Otherwise, the most important acoustic contributions would come from the outside of the FW-H control surface, which would render a false result.

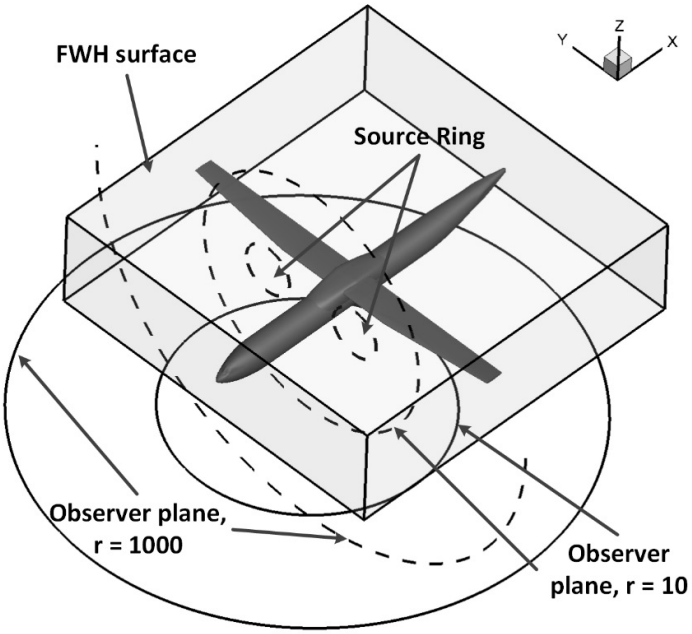


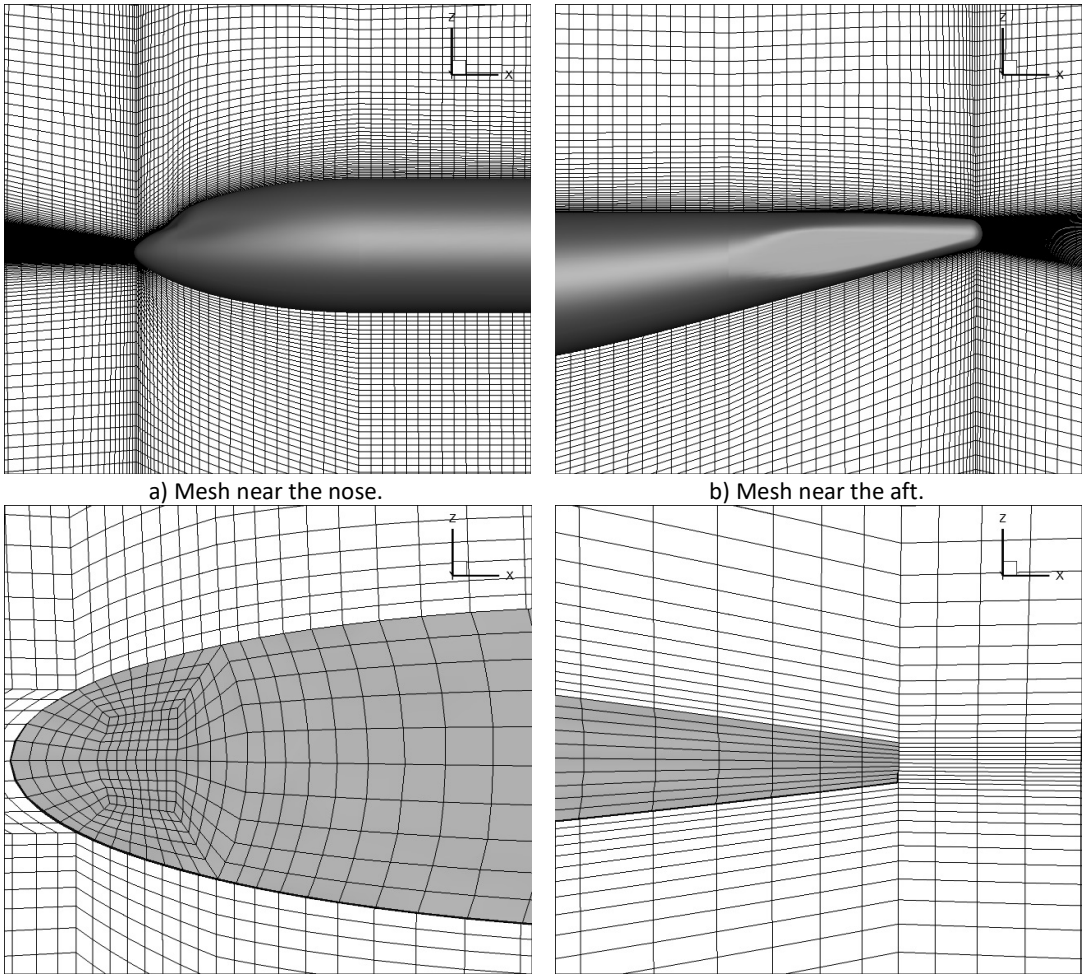
Figure 6.6 Sketch of the observer rings and FW-H control surface.

The computation is in a non-dimensional format. The standard air property at the sea-level is used. The reference length L_∞ is 1 m with the reference velocity 340.2 m/s. The reference density of 1.225 kg/m^3 and reference temperature of 288.2 K are utilized.

6.6 CAA Mesh

For a given CAA solver, such as a commercial solver, the most important and time-consuming part in an application with a complex geometry is the generation of the smooth, high-quality structured mesh. As in chapter 5, the PPW of the CAA mesh is set to 7. In total, there are 1,704 mesh blocks and 23 million mesh points in the physical domain. The largest mesh block contains 78 k mesh points. 16 Tesla K20M GPUs, each with a memory volume capacity of about 4.5 GB, are necessary for performing the computation. In this study, 16 GPUs are employed.

Since high-order schemes are used and the current solver is found not stable enough if a 3D O-mesh is used, much attention and time is spent on the improvement of the quality of mesh. Otherwise, spurious parasite waves close to the solid wall mentioned in Section 2.6.3 are easily excited. Some important local details of the mesh are shown below:



a) Mesh near the nose. b) Mesh near the aft.

c) Mesh at the leading edge at the wing tip. d) Mesh at the trailing edge.

Figure 6.7 Mesh details of important locations.

It can be observed that the mesh near the nose in Figure 6.7 a) and near the aft in Figure 6.7 b) is smooth and the growth rate is 1.1. At the wing tip, a 2D O-mesh stretched straight to the far-field boundary is utilized to improve the mesh at the leading edge as shown in Figure 6.7 c). The mesh density is high at the trailing edge, which makes the overall amount of mesh points large. In the mesh around a complex geometry, the mesh size is more dependent on the minimum mesh points allowed by the numerical schemes and some important locations of the geometry. For example, the thickness of the trailing edge at the wing tip is 0.02 m which is only 0.7% of the wavelength of the first harmonic. However, due to the requirement of the explicit filters, there must be at least 7 points along the trailing edge. A smooth increase on the mesh size also must be ensured near the trailing edge. Consequently, the mesh size is greatly increased due to the locations with small geometric size but important geometric features.

In addition, the most common way to improve the quality of the structured mesh is using the O-mesh close to the solid wall. However, the application using a 3D O-mesh always diverges near the mesh interface close to the solid wall in the current solver. Many attempts have been made to improve the mesh quality and it is found a 2D O-mesh at the wingtip straight to the far-field boundary is allowed as shown in Figure 6.7 c). The cause is attributed to the smoothness and quality of the mesh near the block interfaces close to the solid wall [168], which makes the numerical flux through the interface non-conservative. Since smoothness is not good enough and the grid metrics are discontinuous near the interface of the O-mesh, the parasite waves are excited. The parasite waves are very short (grid to grid oscillations), and cannot be dampened and removed by spatial filtering effectively. This also poses many challenges to mesh generation and increases the difficulty.

6.7 Results

The same procedure of analysis used in the cylinder cases in chapter 5 is employed here. First, the result of spinning monopoles is shown. Then, the investigation is performed on dipole results.

6.7.1 Results of Thickness Noise

The profile of instantaneous sound pressure field on the surface of the wing body is shown in Figure 6.8. First, it can be observed in Figure 6.8 that the pressure contours on the wing body are asymmetric with respect to the symmetry plane of the wing body when both propellers rotate in the counter clock-wise direction viewed by an observer downstream. The acoustic waves concentrate more on the wing at the LHS than at the RHS. This is also demonstrated by the SPL contours in Figure 6.9 a) and b). In addition, the acoustic waves concentrate on the ring plane as shown in Figure 6.9 a) and b), which is the same as that in the cylinder case.

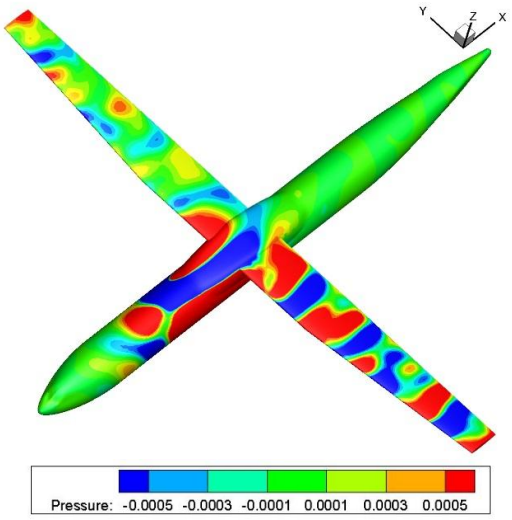
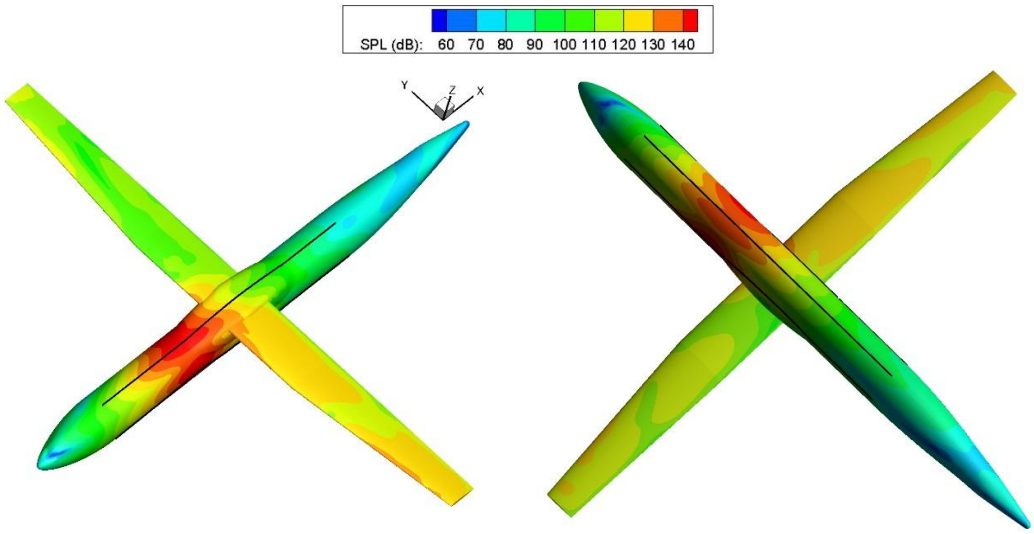


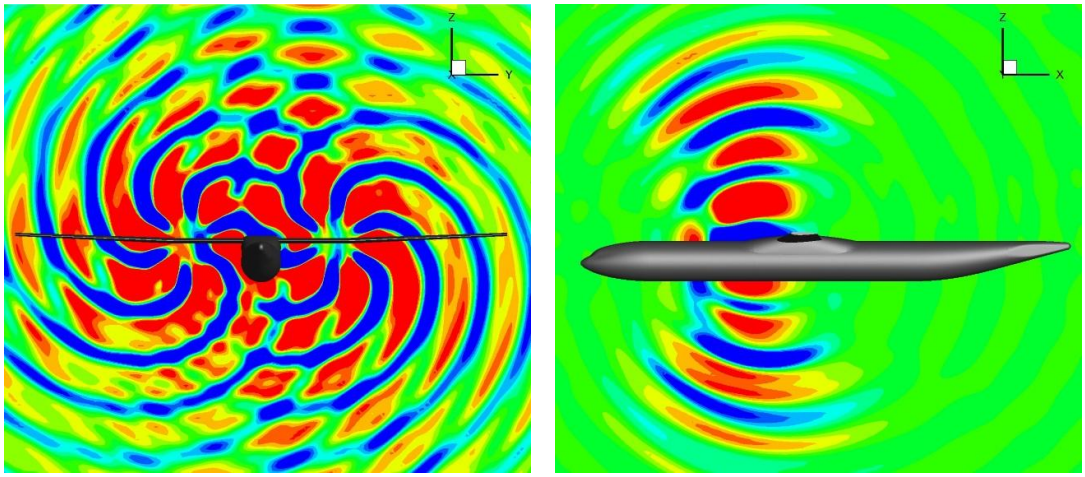
Figure 6.8 Sound pressure contours on the solid wall.



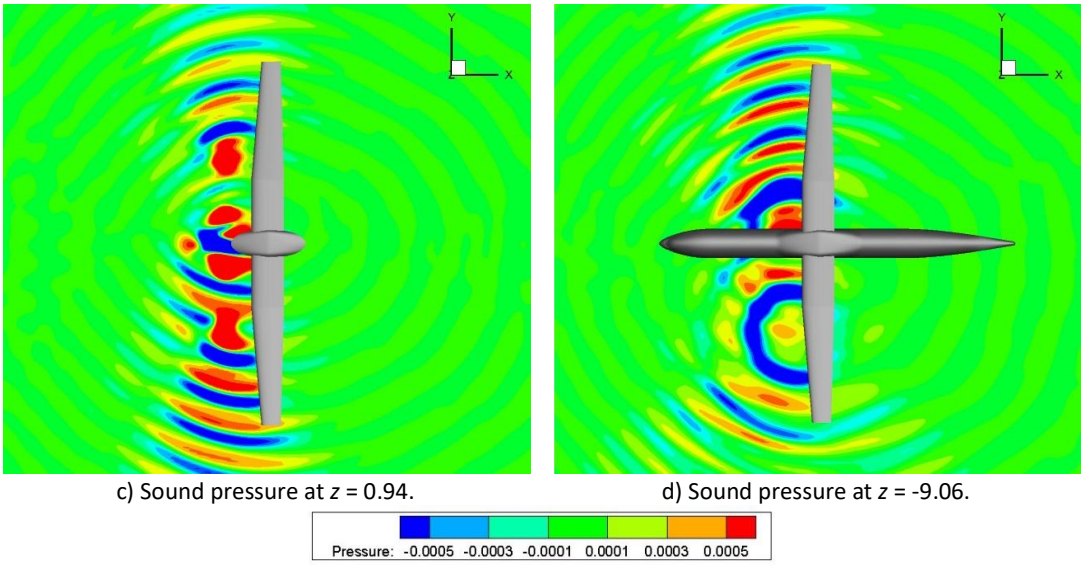
a) SPL on the upper surface in RANS flow. b) SPL on the lower surface in RANS flow.

Figure 6.9 SPL contours on the solid wall.

The contours of instantaneous acoustic pressure field around the wing body in the RANS mean flow are shown in Figure 6.10 below. The acoustic pressure on the ring plane is shown in Figure 6.10 a). The sound pressure is propagating spirally which denotes that the rotation of both rings is in the counter clock-wise direction. In Figure 6.10 b), the acoustic pressure on the symmetry plane is shown and the propagation of acoustic waves near the ring plane is illustrated. The propagation of acoustic wave is inclined slightly upstream due to the convection effect of the mean flow field. The acoustic waves on the planes parallel to the ground are shown in Figure 6.10 c) and d). The plane goes through the ring centres at z = 0.94 in Figure 6.10 c), while the plane is 10 lower than the ring centre in Figure 6.10 d).



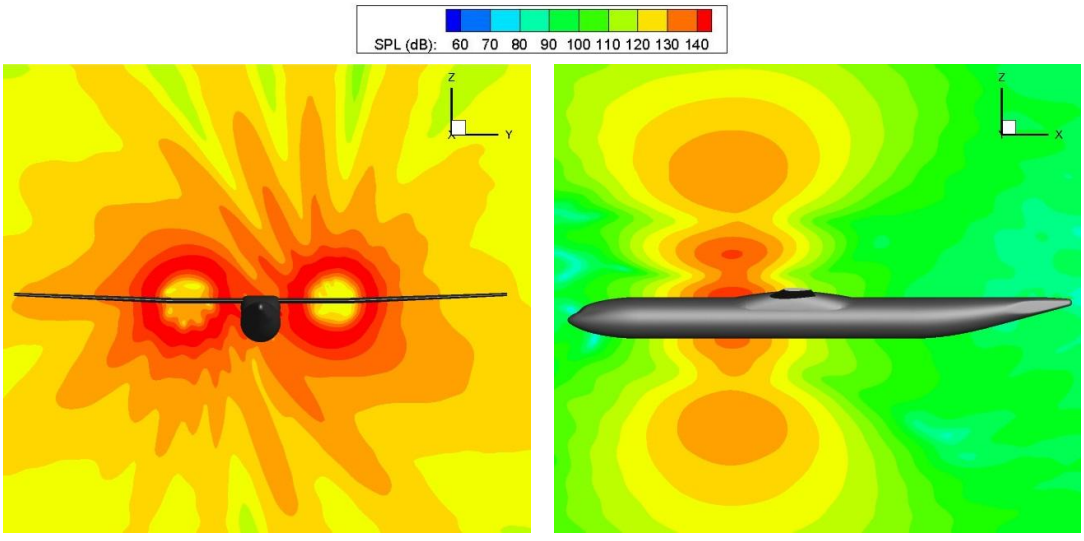
a) Sound pressure on the ring plane (x = 8.8). b) Sound pressure on the symmetry plane (y = 0).



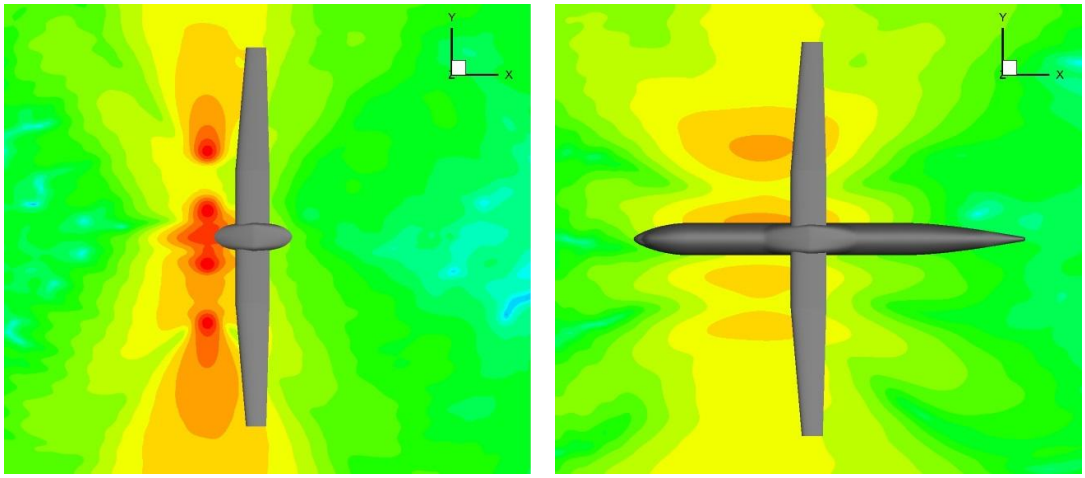
c) Sound pressure at z = 0.94. d) Sound pressure at z = -9.06.

Figure 6.10 Instantaneous pressure contours in RANS mean flow.

The SPL contours captured on the same planes as those shown in Figure 6.10 are shown in Figure 6.11. Multiple peaks and valleys of the SPL distribute on the ring plane as shown in Figure 6.11 a). The directivity is strong due to the existence of the wing body. In addition, difference of SPL occurs between the LHS and RHS with respect to the symmetry plane. The SPL contours at the symmetry plane are shown in Figure 6.11 b). The propagation of acoustic waves concentrates near the ring plane, which are also demonstrated in Figure 6.11 c) and d).



a) SPL on the ring plane (x = 8.8). b) SPL on the symmetry plane (y = 0).



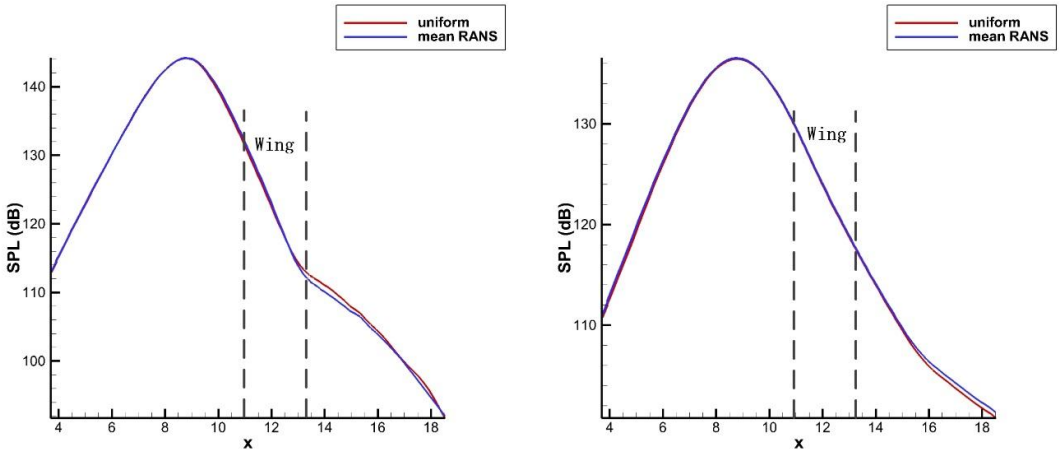
c) SPL at z = 0.94. d) SPL at z = -9.06.

Figure 6.11 SPL contours in RANS mean flow.

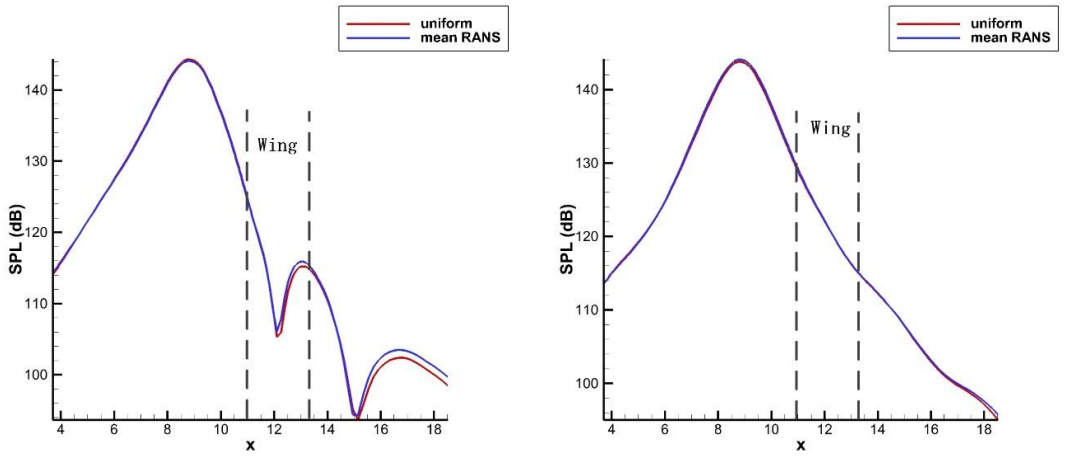
6.7.1.1 Refraction Effect of RANS Mean Flow

6.7.1.1.1 SPL on Wing Body

As mentioned in chapter 5, the sound propagations in the uniform flow and in RANS flow are similar at M = 0.205. To demonstrate the weak refraction effect of the mean flow, four curves on the fuselage are used to collect the SPL, which are shown as the black curves in Figure 6.9. The curves are at the top, the bottom, the LHS and the RHS of the fuselage, respectively. A quantitative comparison of the SPL is made on the fuselage shown in Figure 6.12 below:



a) On the top line. b) On the bottom line.



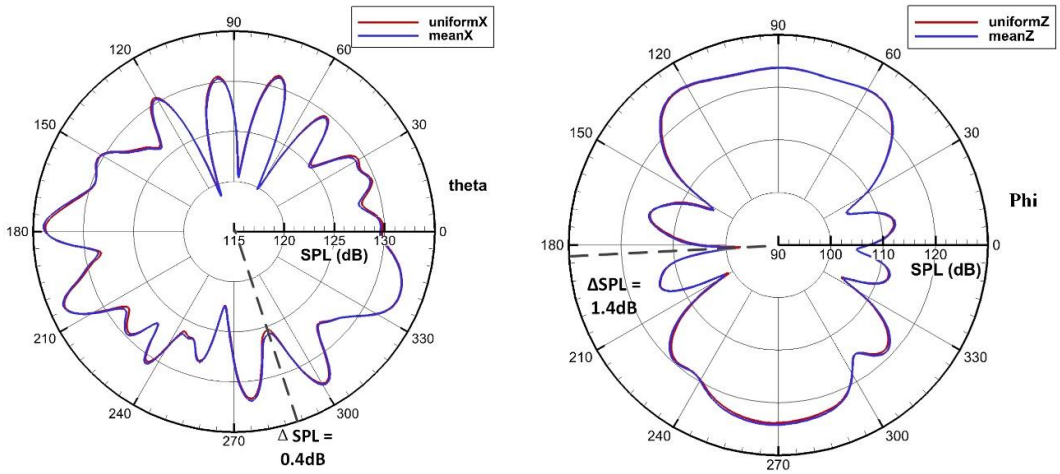
c) On the LHS line. d) On the RHS line.

Figure 6.12 Comparison of SPL on the fuselage in uniform flow and in RANS flow.

The difference between the SPL in the uniform and that in RANS flow on the fuselage wall is small, especially near the peak region as shown in Figure 6.12 a), b), c) and d). The maximum difference between the SPL in the uniform and that in then RANS flow is 1.0 dB on the top curve, 0.5 dB on the bottom curve, 1.0 dB on the LHS and 0.3 dB on the RHS curves respectively. In conclusion, the refraction effect of non-uniform flow on the fuselage is weak.

6.7.1.1.2 Directivities

At the near-field, the observer rings with radii of 10 m, shown in Figure 6.6, are used to compute the SPL around the wing body. The SPL on the ring with the origin at (8.8, 0, 0.94) is shown in Figure 6.13 a). The definitions of the angles used in the directivity are the same as those in the cylinder case as shown in Figure 5.7. The SPL no longer behaves like a Gaussian distribution with respect to ϑ when both propellers are rotating, which occurs when a single propeller rotates. Instead, many peaks and valleys are distributed on the ring plane. In addition, the SPL from LEE coincides well with that in RANS flow. The difference is small and the maximum difference is 0.4 dB denoted by the dotted line in Figure 6.13 a).



a) SPL on the ring plane with the origin at (8.8, 0, 0.94) and a radius of 10.

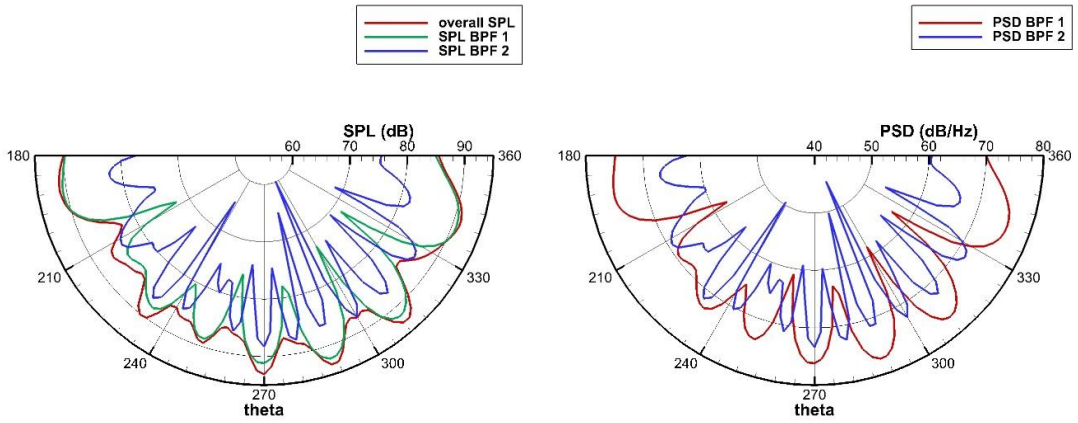
b) SPL under the fuselage with the origin at (8.8, 0, -9.06) and a radius of 10.

Figure 6.13 Near-field SPL directivities.

When observed from the ring with the origin at (8.8, 0, -9.06), the behaviour of SPL like a Gaussian distribution with respect to φ occurs again but with two relatively flat peaks at $\varphi = 90^o$ and at $\varphi = 270^o$ respectively. The peak at $\varphi = 90^o$ corresponds to the RHS of the symmetry plane while the peak at $\varphi = 270^o$ denotes the LHS. Two valleys occur at $\varphi = 0^o$ and $\varphi = 180^o$ which denote the upstream and downstream direction respectively. Still, concentration on the ring plane is the basic feature of the directivity. The maximum difference between the SPL in uniform flow and that in RANS flow is 1.4 dB. The difference in other regions is small and can be ignored.

6.7.1.2 Contributions of Harmonics on Directivities

At the far-field, a half ring under the fuselage with the origin at (8.8, 0, 0.94) and a radius of 1000 as shown in Figure 6.6 is utilized to show the far-field directivity on the ring plane. In addition, another ring with the origin at (8.8, 0, -999.06) under the fuselage is also used to show the far-field directivity. The far-field directivities of the overall SPL and PSDs of the first and second harmonics in the uniform flow are shown in Figure 6.14 below:



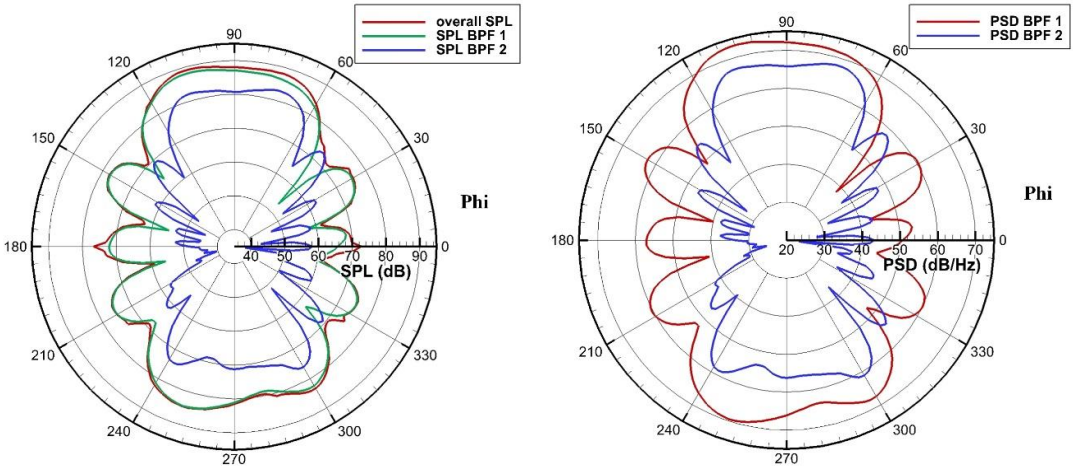
a) SPL on the half ring. b) PSDs on the half ring.

Figure 6.14 Far-field SPL and PSDs in RANS mean flow on the half ring with the origin at (8.8, 0, 0.94) and a radius of 1000.

No behaviour of the Gaussian distribution on the ring plane occurs at the far-field, which is similar to that at the near-field as shown in Figure 6.14 a). Multiple peaks and valleys distribute over the half ring. The peak of the overall SPL occurs at $\vartheta = 270.1^{o}$ with the peak value of 92.8 dB, whereas the value of valley is 80.2 dB at $\vartheta = 207.9^{o}$. A gap of 12.6 dB exists between the peak and the valley, which shows a strong directivity.

A difference of 1.8 dB occurs between the peaks of the overall SPL and the first harmonic whereas the difference between the overall SPL and the second harmonic is 7.4 dB. The peaks and valleys of the overall SPL occur together with those of the first harmonic at the same positions. It denotes that the shape of the overall SPL is dictated by the shape of the first harmonic. However, big difference exists between the valleys of the overall SPL and the first harmonic. This mainly results from the contribution of the second harmonic as shown in Figure 6.14 a). It can be observed that there is always a peak of the second harmonic when a valley of the overall SPL or the first harmonic occurs. The valleys of the first harmonic are compensated for by the peaks of the second harmonic. The second harmonic dominates the value of the valleys of the overall SPL. In addition, the amount of the peaks and valleys of the second harmonic is twice of those of the first harmonic and overall SPL, which is also demonstrated by the PSD shown in Figure 6.14 b).

Qualitative comparisons of the far-field directivities of the overall SPL and the PSDs of the first and second harmonics on the ring with the origin at (8.8, 0, -999.06) and a radius of 1000 under the fuselage in the uniform flow are shown in Figure 6.15 below:



a) SPL on the ring. b) PSDs on the ring.

Figure 6.15 Far-field SPL and PSDs in RANS mean flow on the ring with the origin at (8.8, 0, -999.06) and a radius of 1000 under the wing body.

The behaviour of Gaussian distribution occurs again on the ring under the fuselage as shown in Figure 6.15 a), which is similar to that at the near-field shown in Figure 6.13 b). A gap of roughly 30 dB exists between the peak and valley of the overall SPL. The peaks of the overall SPL and the SPL of the first harmonic coincide well, which denotes that the first harmonic dominate the directivities. The main differences occur at the valley regions where the second harmonic plays an important role in the overall SPL. This behaviour is also demonstrated by the PSDs shown in Figure 6.15 b).

6.7.2 Results of Loading Noise

The analysis similar to that on thickness noise is made on the result of loading noise. The profile of instantaneous sound pressure field on the wing body is shown in Figure 6.17. Three features which are the same as that in the thickness noise case are found. First, the pressure contours on the wall, especially on the wing, are asymmetric with respect to the symmetry plane of the wing body when both propellers rotate in the counter clock-wise direction viewed by an observer downstream as shown in Figure 6.16. The propagation of acoustic waves concentrates more at the LHS than at the RHS. This is also demonstrated by the SPL contours in Figure 6.17 a) and b). In addition, the acoustic waves concentrate near the ring plane as shown in Figure 6.17 a) and b).

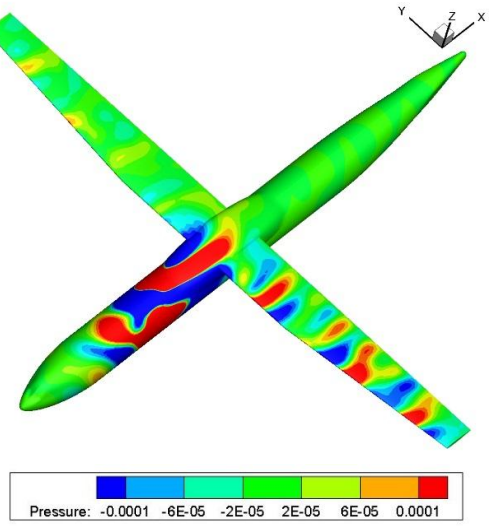
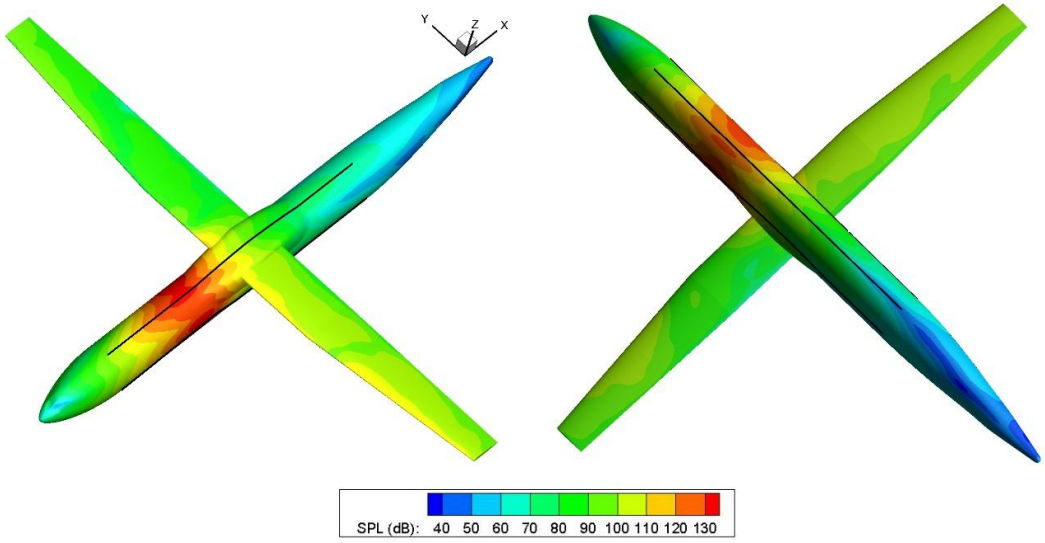


Figure 6.16 Sound pressure contours on the wing body.

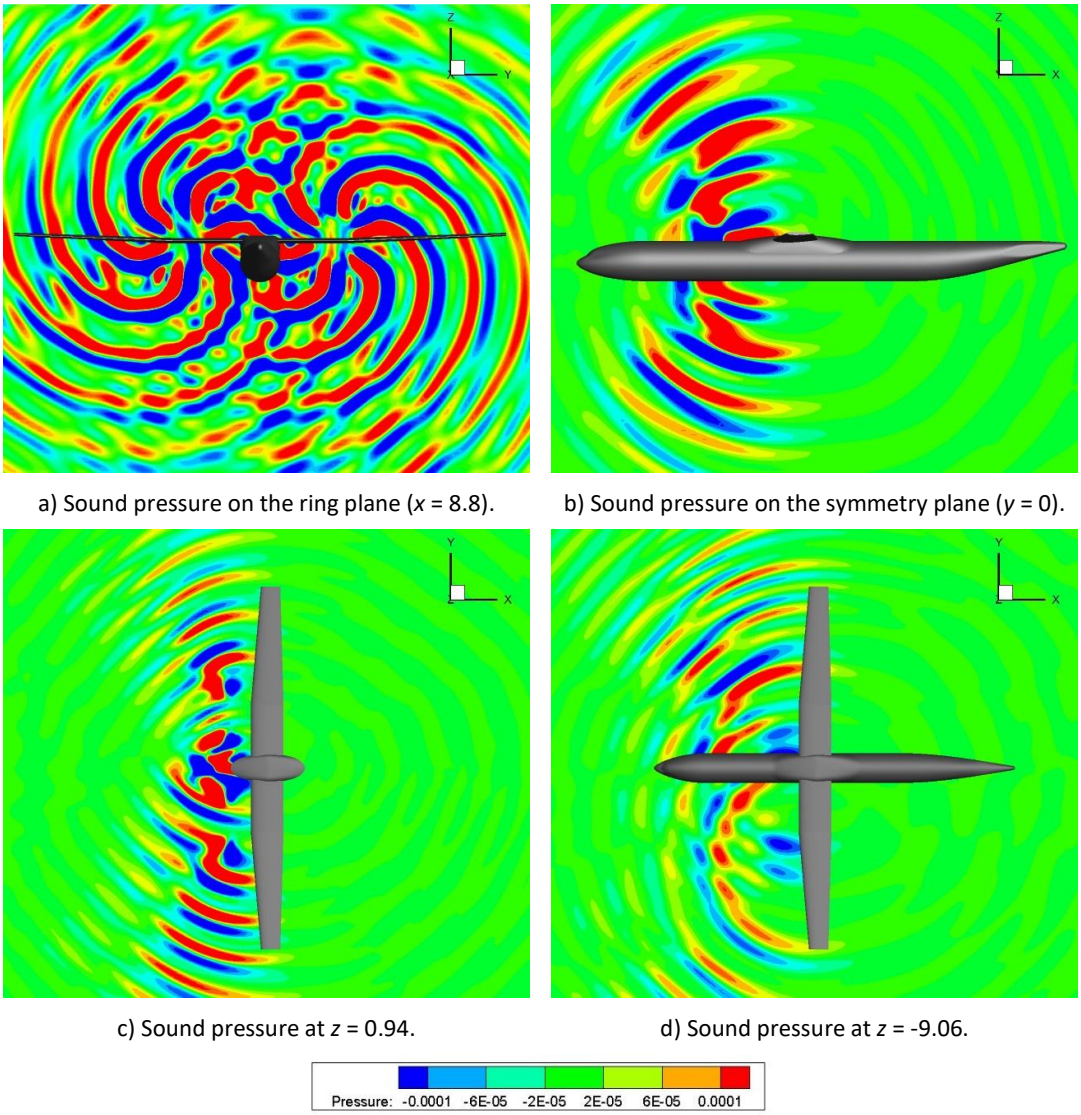


a) SPL on the upper surface in RANS flow. b) SPL on the lower surface in RANS flow.

Figure 6.17 SPL contours on the wing body.

The contours of instantaneous acoustic pressure field around the wing body in the RANS mean flow are shown in Figure 6.18. The acoustic pressure on the ring plane is shown in Figure 6.18 a). The spiral propagation of sound pressure illustrates the rotation of both rings in the counter clock-wise direction. In Figure 6.18 b), the acoustic pressure on the symmetry plane is shown and the concentration of acoustic waves near the ring plane can be observed. The propagation of acoustic wave is inclined upstream as that in the cylinder case. Furthermore, the incline of the acoustic wave is also illustrated in Figure 6.18 c) and d). The acoustic waves on the planes parallel to the ground

are shown in Figure 6.18 c) and d). The plane goes through the ring centres at z = 0.94 in Figure 6.18 c), whereas the cut plane below the wing body is 10 lower than the ring centre in Figure 6.18 d).



a) Sound pressure on the ring plane (x = 8.8).

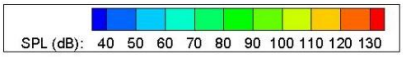
b) Sound pressure on the symmetry plane (y = 0).

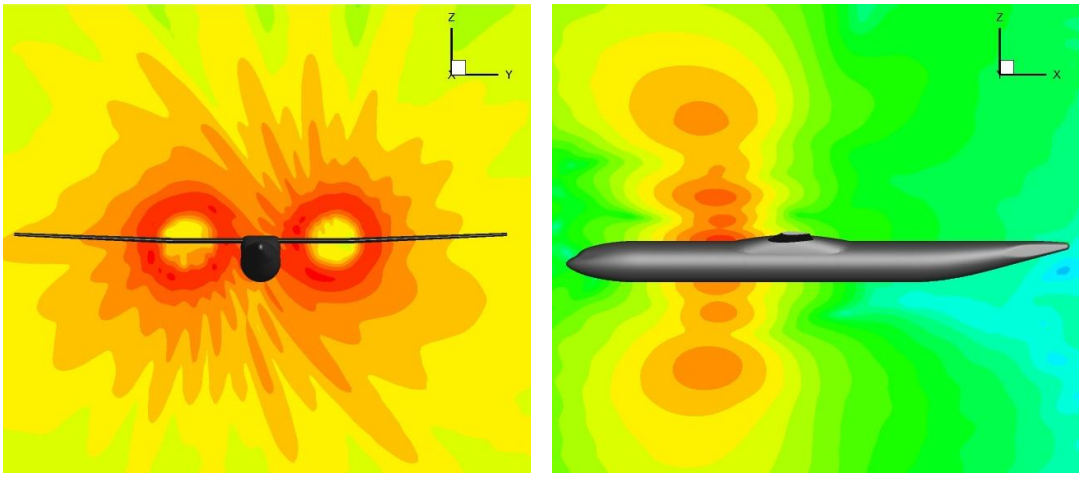
c) Sound pressure at z = 0.94.

d) Sound pressure at z = -9.06.

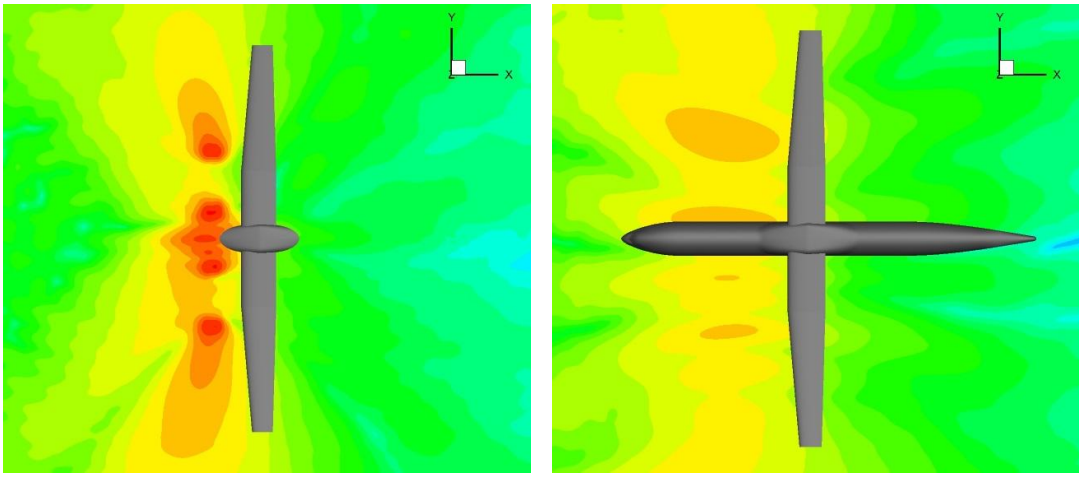
Figure 6.18 Instantaneous acoustic pressure contours around the wing body in RANS mean flow.

The SPL contours captured on the same planes as those shown in Figure 6.18 are shown in Figure 6.19. Multiple peaks and valleys of the SPL distribute on the ring plane shown in Figure 6.19 a). The SPL contours at the symmetry plane are shown in Figure 6.19 b). It can be observed that the propagation concentrates near the ring plane. Furthermore, the incline of SPL upstream is much stronger in comparison to that in the case of thickness noise. The concentration near the ring plane and the incline upstream are also demonstrated in Figure 6.19 c) and d).





a) SPL on the ring plane (x = 8.8). b) SPL on the symmetry plane (y = 0).



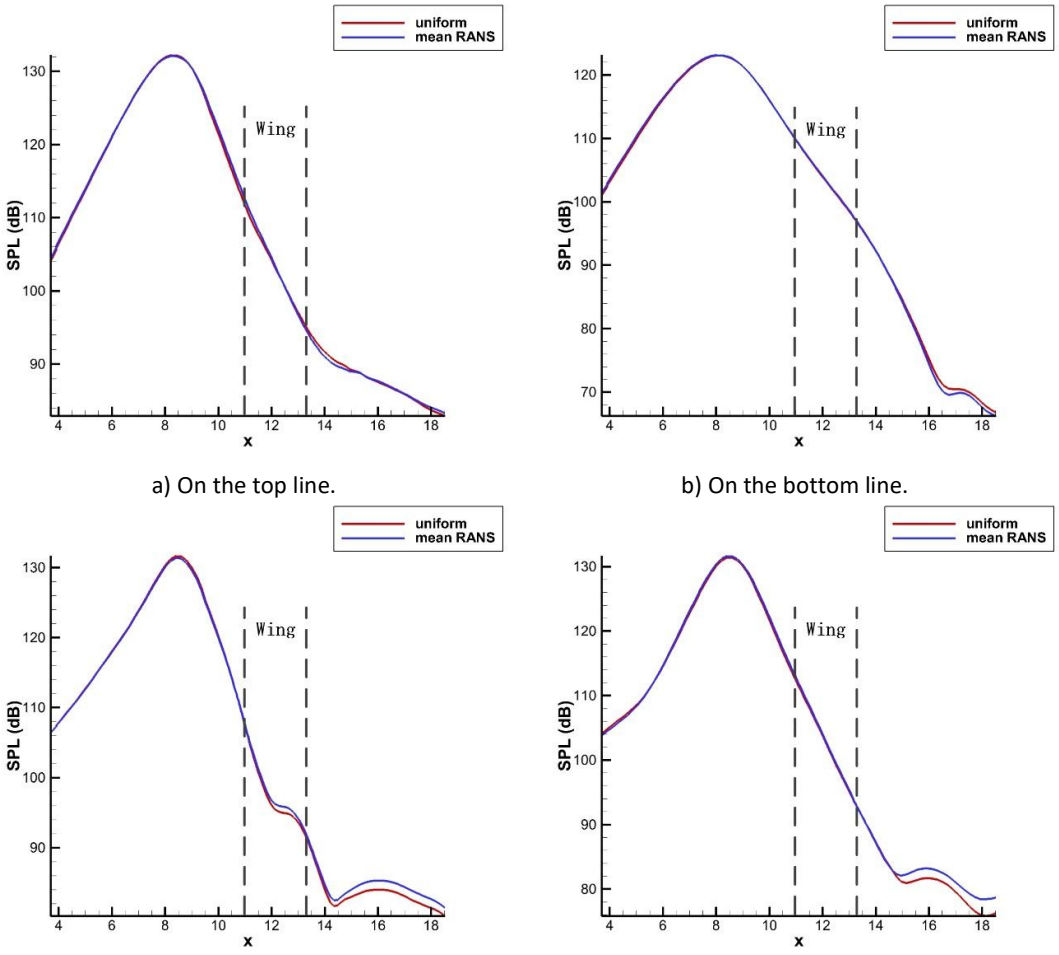
c) SPL at z = 0.94. d) SPL at z = -9.06.

Figure 6.19 SPL contours around the wing body in RANS mean flow.

6.7.2.1 Refraction Effect of RANS Mean Flow

6.7.2.1.1 SPL on Wing Body

It has been known in chapter 5 that the sound propagations in the uniform flow and that in RANS flow are similar at M = 0.205. To demonstrate the weak refraction effect of the mean flow, the same four curves on the fuselage are used to collect the SPL in the monopole case, which are shown as the black lines in Figure 6.17. The lines are at the top, the bottom, the LHS and the RHS of the fuselage, respectively. A quantitative comparison between the SPL in the uniform flow and in the RANS flow is made on the SPL on the fuselage shown in Figure 6.20 below:



a) On the top line. b) On the bottom line.

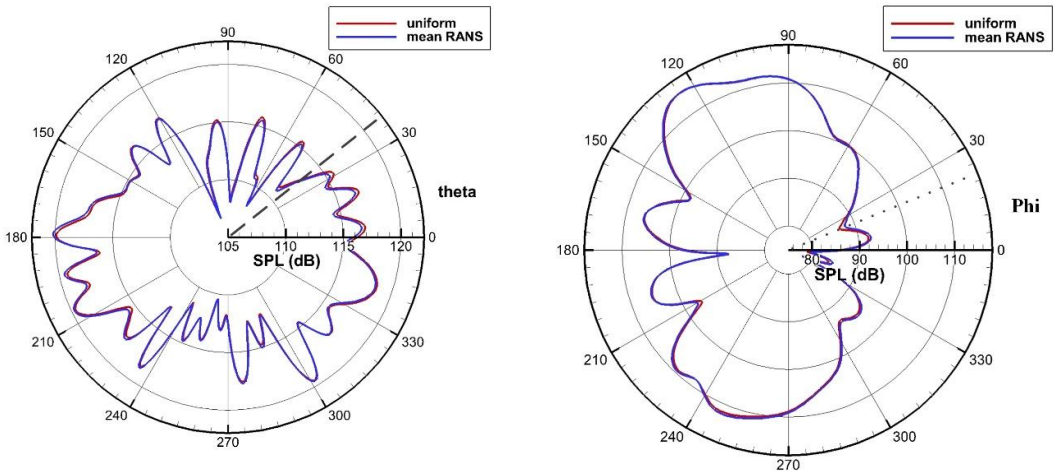
c) On the LHS line. d) On the RHS line.

Figure 6.20 Comparison of SPL on the fuselage in uniform flow and in RANS flow.

It is illustrated in Figure 6.20 a), b) c) and d) that the difference between the SPL in the uniform and that in RANS flow on the fuselage wall is small, especially near the peak region, which is similar to that in the spinning monopole case. The maximum difference between the SPL in the uniform and that in then RANS flow is 0.3 dB for the LHS and the RHS respectively. In conclusion, the refraction effect of non-uniform flow on the fuselage is weak and can be ignored.

6.7.2.1.2 Directivities

At near-field, the observer rings with radii of 10 m shown in Figure 6.6 are used to compute the SPL around the wing body. The SPL on the ring with the origin at (8.8, 0, 0.94) is shown in Figure 6.21 a). As in the monopole case, many comparable peaks and valleys are distributed on the ring plane. In addition, the SPL from LEE coincides well with that in RANS flow. The difference is small and the maximum difference is 0.4 dB denoted by the dash line in Figure 6.21 a). These features are similar to those of the monopole case.



a) SPL on the ring plane with the origin at (8.8, 0, 0.94) and a radius of 10.

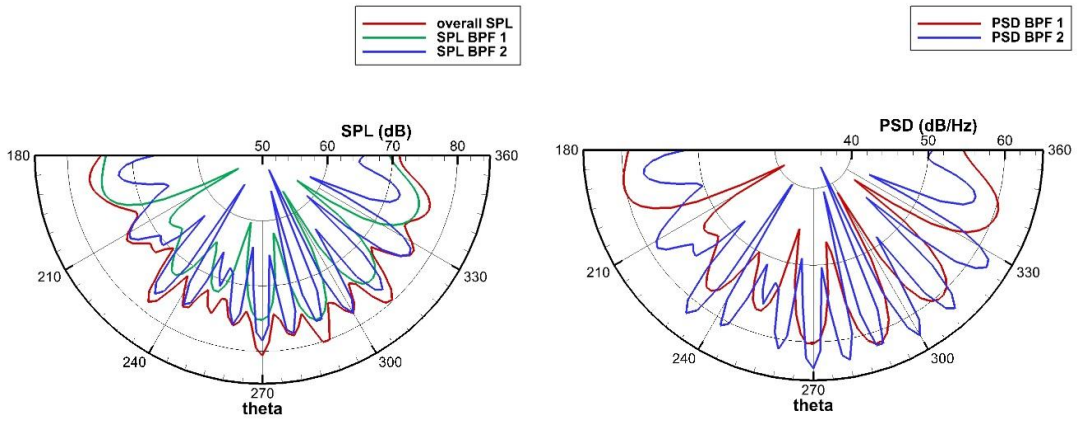
b) SPL under the fuselage with the origin at (8.8, 0, -9.06) and a radius of 10.

Figure 6.21 SPL directivities at the near-field.

When observed from the ring parallel to ground with the origin at (8.8, 0, -9.06), the behaviour like the Gaussian distribution with respect to θ of SPL occurs again but with two relatively flat peaks inclined upstream at φ = 124.3° and at φ = 247.9° respectively, which is different from that in the spinning monopole case. The peak at φ = 124.3° corresponds to the RHS of the symmetry plane, whereas the peak at φ = 247.9° denotes the LHS. In the cases of thickness noise, the concentration on the ring plane is the basic feature of the directivity. In contrast, the concentration inclined upstream of the ring plane is the key behaviour of the directivity. This is due to the interference of the propagation of forces F_x and F_ϑ. The maximum difference between the SPL in uniform flow and that in RANS flow is small, about 1.2 dB denoted by the dotted line shown in Figure 6.21 b). The difference in other region is small and can be ignored.

6.7.2.2 Contributions of Harmonics on Directivities

At far-field, the same rings used in the spinning monopole cases are utilized again. A half ring under the fuselage with the origin at (8.8, 0, 0.94) and a radius of 1000 is utilized to show the far-field directivity on the ring plane. In addition, another ring with the origin at (8.8, 0, -999.06) parallel to ground under the fuselage is also used to show the far-field directivity. The far-field directivities of overall SPL and PSDs of the first and second harmonics in the uniform flow are shown in Figure 6.22 below:



a) SPL on the ring with the origin at (8.8, 0, 0.94) and a radius of 1000.

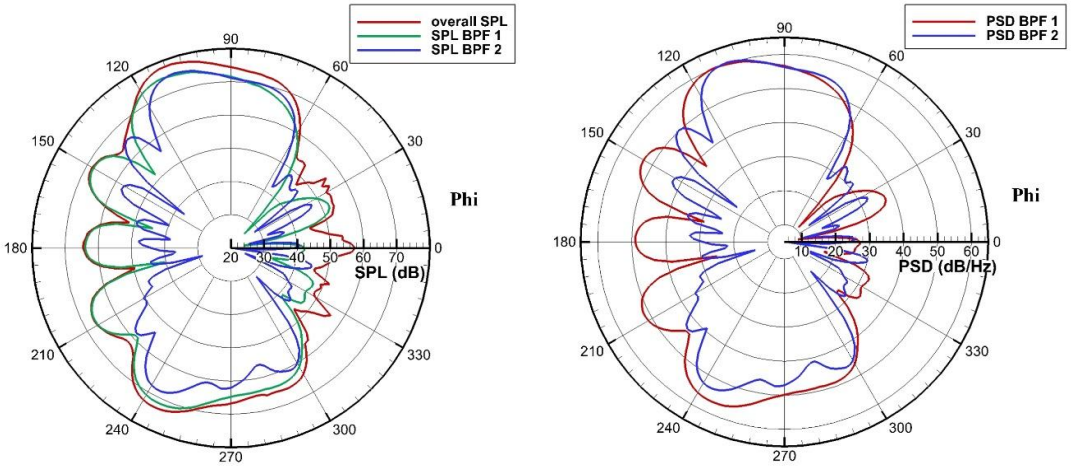
b) SPL under the fuselage with the origin at (8.8, 0, 0.94) and a radius of 1000.

Figure 6.22 Far-field SPL and PSDs in RANS mean flow on the half ring with the origin at (8.8, 0, 0.94) and a radius of 1000 on the ring plane.

The directivity on the ring plane at the far-field shown in Figure 6.22 a) is similar to that at the near-field. Multiple peaks and valleys are distributed on the half ring. The peak of overall SPL occurs at ϑ = 270.0^o with the peak value of 80.7 dB, whereas the value of the valley is 69.7 dB at ϑ = 226.0^o. A gap of 11.0 dB exists between the valley and the peak, which shows a strong directivity.

At ϑ = 270.0^o the peak is 75.2 dB for the first harmonic and 78.4 dB for the second harmonic. A difference of 5.5 dB occurs between the peaks of the overall SPL and the first harmonic, whereas the difference between the overall SPL and the second harmonic is 2.3 dB. It can be observed that the peaks and valleys of the overall SPL occur at the same position as those of the second harmonic. It denotes that the shape of the overall SPL is dictated by the shape of the second harmonic, which is quite different from that in the spinning monopole case. However, big difference exists between the valleys of the overall SPL and the second harmonic. This mainly results from the contribution of the first harmonic as shown in Figure 6.22 a). It can be observed that the first harmonic dominates the overall SPL when a valley of the overall SPL or the second harmonic occurs. The valleys of the second harmonic are compensated for by the first harmonic. In addition, the amount of the peaks and valleys of the first harmonic is half of that of second harmonic and overall SPL, which is also demonstrated by the PSD shown in Figure 6.22 b).

Qualitative comparisons of the far-field directivities of the overall SPL and PSDs of the first and second harmonics on the ring parallel to ground with the origin at (8.8, 0, -999.06) and a radius of 1000 under the fuselage in the RANS mean flow are shown in Figure 6.23 below:



a) SPL on the ring. b) PSD on the ring.

Figure 6.23 Far-field SPL and PSDs in RANS mean flow on the ring with the origin at (8.8, 0, -999.06) and a radius of 1000 below the wing body.

The directivity on the ring under the fuselage is similar to that at the near-field shown in Figure 6.21 b). A gap of roughly 33 dB exists between the peak and valley of the overall SPL. In addition, the peaks of the overall SPL and the SPL of the first harmonic coincide well upstream, whereas difference occurs downstream. In comparison to the directivity on the ring plane, the first harmonic plays a more important role at the far-field under the fuselage. The amount of the peaks and valleys of the overall SPL is determined by that of the first harmonic. The main difference between the overall SPL and the first harmonic occurs at the valley regions where the second harmonic contributes considerably to the overall SPL. This behaviour is also demonstrated by the PSDs shown in Figure 6.23 b). A gap of 10 dB/Hz – 20 dB/Hz is observed between the PSDs of the first harmonic and the second harmonic at the peaks of the first harmonic upstream.

6.8 Performance

The computational performance is an important aspect in this research in addition to the physical phenomena. In the current case, the minimum allowable amount of GPUs is 16 due to the requirement on the memory volume if an output to the FW-H solver is performed. Consequently, the performance is compared on 16 CPU cores and 16 GPU cards for both the monopole case and the dipole case. The performance is summarized below:

Table 6-1. Wall-clock time comparisons in seconds.

Case name	Mesh size	Wall-clock time (in seconds)		
		16 GPUs	16 CPU cores	Speed-up
Thickness noise	22.6 M	40420.7	478260	11.9
Loading noise	22.6 M	58530.7	703478	12.2

As shown in Table 6-1, 11 hours are necessary for the prediction of the thickness noise on 16 GPUs, whereas 16 hours have to be spent on the case of loading noise. In comparison, 133 hours and 195 hours are necessary on 16 CPU cores. However, the speed-up on GPUs is not high, about 12 for both cases. Four causes are analysed here:

1) Though the overall mesh size in the current case is large, the maximum size of the mesh block is small, which reduces the overall performance. Recall that, as mentioned in section 4.6, the performance on the GPU increases with the mesh size and the performance on the maximum mesh block dominates the overall performance. On the other hand, a total of 1704 mesh blocks exist, whereas the largest mesh block contains only 78 K mesh points in this application as mentioned in section 6.6. The performance of these small mesh blocks on GPUs is relatively low in comparison to the 3D cases in section 4.6. The overall mesh is distributed across many small mesh blocks since strong controls have to be made on the mesh quality via many small mesh blocks, which decreases the overall performance.
2) The overall performance is reduced by the data transfers among CPU cores though good loading balance is achieved. In this case, all the mesh blocks are distributed evenly among GPUs, each of which is distributed roughly 107 mesh blocks and 1.4 million grid points. However, the connection amount is high and a total 4261 connections exist between all the mesh blocks. Although these connections are not all the connections between the mesh blocks distributed on different GPUs, they contribute considerably to the overhead of data transfer and index transformation, and result in the final performance penalty.
3) The performance gap between the Tesla K20m GPU and the Intel Xeon E5–2670 is smaller as tabulated in Table 1-2. In Table 1-2, the gap of Flops in double precision between the later GPU and CPU drops from 13.4 to 7 which is nearly the half of the gap between the older GPU and CPU. Consequently, the speed-up also decreases.
4) The cache size plays a more important role in the speedup of the codes on CPUs over GPUs. Recall that the performance of the codes with small mesh blocks is high in section 4.6.6. Data access to the cache, fast access memory on a CPU, is implicitly managed and optimized by compilers. Therefore, the effect of the cache size on the performance of the codes is high. The performance on the small mesh blocks is high on CPUs. On a GPU, the shared memory, fast access memory on a GPU, is explicitly managed by the codes. The performance on the small mesh blocks is relatively low on GPUs. As a result, the speedup of the codes on GPUs over CPUs decreases.

In conclusion, though the speed-up decreases, the GPU still offers a feasible way to fast predict the large-scale engineering cases. The speed-up can be even higher if the high-order code, SotonLEE, is more stable on a structured mesh since the size in the mesh blocks can be larger.

6.9 Summary

In this section, the case of propeller noise scattering off an aircraft has been predicted by CAA methods and the speedup has been evaluated. The numerical methods and sound sources in the following chapter were kept the same as those in Chapter 5. The research focus was not placed on the analysis and explanation on physical phenomena but was mainly devoted to demonstrate the application to a large-scale engineering case and evaluate the true performance of the current solver in engineering problems.

For the thickness noise radiated by the ring of spinning monopoles, the key findings were:

- The thickness noise is asymmetric with respect to the symmetry plane both around and on the wing body.
- The thickness noise concentrates on the ring plane of the wing body.
- The refraction effect of the RANS flow field around the wing body at M = 0.205 is weak and illustrated again in the current case.
- At far-field, two peaks of the directivity occur corresponding to the propeller positions.
- At far-field, the first harmonic dominates the basic directivity of the overall SPL, whereas the second harmonic offers important contribution to the valleys of the overall SPL. The peaks of the overall SPL are dictated by those of the first harmonic. In addition, the amount of the peaks and valleys of the overall SPL is determined by the amount of those of the first harmonic.

For loading noise radiated by the ring of spinning dipoles, the key findings are:

- The loading noise is also asymmetric with respect to the symmetry plane both around and on the wing body.
- The loading noise concentrates upstream the ring plane of the wing body.
- A substantial difference occurs on the ring plane between far-field directivity in the dipole case and that in the monopole case. It is the second harmonic which dominates the basic directivity of the overall SPL, whereas the first harmonic plays an important role in the valleys of the overall SPL. The peaks of the overall SPL are dictated by those of the second harmonic. In addition, the number of the peaks and valleys of the overall SPL is determined by the number of those of the second harmonic.

- At the far-field under the wing body, the first harmonic dominates the basic directivity of the overall SPL again, whereas the second harmonic offers important contribution at the valleys of the overall SPL. The peaks of the overall SPL are dictated by those of the first harmonic. In addition, the number of the peaks and valleys of the overall SPL is determined by that of the first harmonic.

A speed-up of about 12 was achieved based on the implementations on 16 GPUs and those on 16 CPU cores in the current cases. 11 hours were necessary for the thickness noise computation on 16 GPUs, while 16 hours must be spent on the simulation of the dipole case, which offers an acceptable wall-clock time for computation of propeller noise scattering off a wing-body at a full scale. The relative low speed-up in the current case in comparison to those in Chapter 4 has been summarized. The causes include:

- The relatively small scale of the maximum mesh block size in the current case decreases the overall performance.
- The overall performance is reduced by the data transfers among small mesh blocks on different CPU cores.
- The hardware is different, which results in a smaller performance gap between the CPUs and GPUs. Moreover, the performance of the applications implemented on CPUs benefits more from the larger cache size on the current CPU chip. Consequently, the speed-up decreases.

Chapter 7: Conclusions and Future Work

7.1 Conclusions

Noise generated by aircrafts has become an increasingly important issue with the development of civil aircraft design requirements. With the increasing demand on the physics of aerodynamic sound as well as the development of computational power of computers and numerical techniques, CAA has been increasingly employed to study airframe/engine noise. However, the high computational cost and the long run-time prevent the widespread CAA application to real engineering problems. A hierarchy of methods has been developed and utilized to accelerate the implementation of CAA simulations, including the CAA hybrid methods on the physical aspect, the CAA numerical schemes on the numerical aspect and the HPC method on the implementation.

The aim of this research was to accelerate an existing high-order, CAA scattering solver, based on finite difference method, on multiple GPUs and investigate the refraction effect of the boundary layer on propeller noise computationally, and finally apply it to a large-scale engineering case. Consequently, this research was divided into three phases: the development of the CAA programme the investigation of the refraction effect of boundary layer computationally, and the application on engineering problems. First, this research employed the GPU to reduce the wall-clock time of an available CAA scattering solver which features the CAA hybrid method and the numerical schemes with high-order accuracy, low-dissipation and low-dispersion. Second, the propeller noise scattering off a cylinder with boundary layer was investigated computationally using the new CAA scattering solver. Finally, the propeller noise scattering off a full-scaled wing body was predicted and the computational performance was analysed.

7.1.1 Numerical Methods

At the development phase, the research focused on the development of efficient algorithms and strategies of implementation of the existing high-order CAA solver, SotonLEE, on multiple GPUs. First, it was found that the computation of prefactored compact scheme is the most computing-intensive part and dominates the overall performance of the current CAA solver. The other subroutines do not contribute considerably to the overall performance. Consequently, the key to the efficient solution of the current CAA solver on a GPU is to find an efficient solution of the optimized prefactored compact schemes. Second, the optimized prefactored compact schemes give rise to bidiagonal matrices. As a result, the key to the efficient solution of optimized prefactored compact schemes on a GPU is to find an efficient solution of bidiagonal matrices. Three

methods were employed for the first time to solve the bidiagonal matrix on the GPU, namely Natural method, PCR method and MatMul method. The PCR method is the most efficient in 2D computations and in large-scale 3D computations. The MatMul method is efficient in small mesh block size in 3D computations whereas the Natural method achieves poor performance. Third, in the development of algorithms, it was found that the memory access pattern plays a key role in the performance of an application implemented on the GPU. The memory access pattern contains coalesced memory access and redundant memory access. The coalesced memory access results in high performance whereas the redundant memory access causes performance penalty. Fourth, the anisotropic memory access pattern was investigated for the first time. The poor performance of the Natural method results from the redundant memory access in the *x* direction whereas the PCR and MatMul methods achieve good performance with the aid of coalesced memory access pattern in the *x* direction. However, the redundant memory access in the *y* and *z* directions still exists for the PCR method. Finally, a hybrid method has been proposed in terms of anisotropic memory access pattern. It combines the best performance of different algorithms and achieves the highest performance in 2D and 3D computations. Accordingly, the solving strategy has been also formulated dependent on the mesh sizes.

In terms of different memory access pattern, subroutines in SotonLEE were categorized into five types: implicit stencil type, explicit stencil type, point-wise type, unstructured gather type and reduction type. Accordingly, different types of subroutines have been applied with different parallel strategies. The “tiling method” was applied to the explicit stencil type whereas the unstructured gather type was paid much attention on the index transformation. The reduction type was investigated by using Harris Kernel 3. In addition, the cost of data transfer between multiple GPUs has been mitigated by using the GPUDirect. The new solver, SotonLEE_GPU, achieves a series of speed-ups over multiple academic and engineering cases. For 2D computations, the maximum speed-up achieves 78 for the 2D cylinder case whereas the peak speed-up achieves 54 for the 3D engine bifurcation case.

7.1.2 Introduction of Propeller Noise Source into LEE

At the second phase, the study investigated the scattering of the noise of a single propeller off a cylinder by using the CAA method for the first time. In LEE, the thickness noise of a propeller is imitated by a ring of spinning monopoles, introduced into LEE by the term s_1 in the continuity equation or s_5 in the pressure equation. The loading noise is modelled by a ring of spinning dipoles, introduced into LEE by the terms s_2, s_3 and s_4 in the momentum equations. The Dirac Delta function in the monopoles and dipoles is smoothed by the 3D Gaussian distribution in which the parameter σ plays a key role in the numerical implementation. The numerical solution of a ring of spinning

monopoles in the free space was calibrated to analytical solution of Hanson's method in the frequency domain and Farassat's Formulation 1A in the time domain. It was found that the PPW of the mesh at the acoustic source region plays a key role in the amplitude of the acoustic propagation. The PPW of the mesh at the source region must meet the requirement of PPW of the numerical schemes to mitigate dispersion error. The amplitude of the acoustic wave must be modulated to be calibrated to the analytical solution. After the calibration, the numerical solution corresponds well with the analytical solution. Before CAA simulations, the contribution of each harmonic can be identified by the analytical solution. Consequently, the highest order of the harmonic that should be resolved can be identified. The mesh size in the numerical simulation can be determined basically.

7.1.3 Propeller Noise Scattering off a Cylinder with Boundary Layer

In the LEE implementations, the mesh only resolved the first harmonic since the contribution of the second harmonic was much smaller. A validation was firstly performed between the results obtained by LEE and those from CESM. The good agreement between the results validated the source model, numerical methods in SotonLEE and PPW of the mesh. Then, the LEE implementation has been employed to investigate the scattering of propeller noise off a cylinder and the refraction effect of the boundary layer.

On the cylinder wall, both the thickness noise and loading noise behave like a Gaussian distribution along the stream-wise direction. The SPL drops sharply when the observer is some distance away from the ring plane. The propagation of thickness noise concentrates on the ring plane whereas that of loading noise inclines upstream. At far-field, the directivity contains many peaks and valleys due to the reflection of the cylinder wall. The SPL directivity on the ring plane and on the ground behaves like a Gaussian distribution with respect to the observer position with the peak occurring at the same side as the propeller. At low Mach number case, the refraction effect of the boundary layer was found negligible whereas the refraction effect is significant at Cruise Mach number computationally. The boundary layer reduces the SPL on the cylinder and far-field directivity upstream of the ring plane. Extensions of computation at M = 0.3 and 0.4 have been performed to determine at which Mach number the refraction effect of boundary layer becomes important. The difference between results in the absence of boundary layer and with a realistic boundary layer showed that the refraction effect of the boundary layer starts to be important when Mach number is larger than 0.3.

On performance, 9 minutes are necessary for the cylinder cases on two GPUs at the low Mach number, whereas 24 minutes must be spent at the cruise Mach number since the total mesh size is

larger. A speed-up of between 21 and 26 is achieved in the low Mach number cases running on two GPUs whereas the speed-up is roughly 30 in the cruise Mach number cases.

7.1.4 Propeller Noise Scattering off a Wing-Body at a Full Scale

At the application phase, the study predicted the scattering of the double propeller noise off a wing body with a real size. This case is an extension of the cylinder case on the complexity of the CAA mesh and the computational cost. The performance on GPUs and on CPU cores has been recorded and compared.

When both propellers rotate in the counter clock-wise direction viewed by an observer downstream, the acoustic field on the wing body wall and around the wing body is asymmetric with respect to the symmetry plane of the wing body. The propagation of thickness noise concentrates on the ring plane whereas that of loading noise inclines upstream. At far-field, two peaks of the directivity occur corresponding to the propeller positions. On the ring plane, the shape of the overall SPL of thickness noise is dominated by the first harmonic, whereas the second harmonic contributes significantly at the valleys of the overall SPL. The second harmonic dictates the basic shape of the overall SPL of loading noise whereas the first harmonic contributes considerably to the valleys of the overall SPL. On the ground at the far-field, the first harmonic dominates the overall SPL again, which is similar to that in the thickness noise. The second harmonic contributes considerably to the valleys of the overall SPL. In addition, the refraction effect of the non-uniform flow around the wing body is demonstrated weak and can be ignored.

On performance, 11 hours are necessary for the thickness noise computation on 16 GPUs, while 16 hours have to be spent on the simulation of the dipole case, which offered an acceptable wall-clock time for computation of propeller noise scattering off a wing-body at a full scale. A speed-up of about 12 is achieved in the current cases. The causes on the performance have been analysed: including the overall mesh size, the largest mesh block size, the cache size on the CPU and difference between the hardware used in IRIDIS 3 and IRIDIS 4.

7.2 Future Work

In this research, some deficiencies were found in the current CAA solver. Some possible topics are suggested here to make the solver more robust in future work:

- Improvement of the far-field boundary conditions. It was found that the far-field condition was the main bottleneck when the scattering of dipole noise was simulated in LEE. More

sophisticated far-field conditions should be incorporated into the current solver to reduce the dimension of the physical domain and to mitigate the reflection.

- Robust support of 3D O-mesh. It is found that the 3D O-mesh is not well sustained in the current solver when it is close to the solid wall. The simulations using the 3D O-mesh around the solid wall blows up at the block interfaces in the current solver since the smoothness the mesh is not good enough. This causes the numerical flux through the interface non-conservative. Therefore, it makes the mesh generation difficult and time-consuming, and increases the total mesh amount. The new interface conditions [168] can be probably utilized to improve the support of the 3D O-mesh in future work.

Regarding the scattering of the propeller noise, more investigations are suggested below for future work:

- Investigation of distributed source: In this research, only point sources were investigated. The distributed sources must be investigated if further application to the engineering problem is to be performed.
- Investigation of higher harmonics: In this study, only the first and second harmonics were investigated. The higher harmonics were proved to be unimportant. However, the higher harmonics might contribute, to some extent, at the valleys of the overall SPL. In addition, the refraction effect of the non-uniform flow is stronger when the acoustic wavelength is shorter. The refraction effect of the boundary layer on higher harmonics should be investigated.
- Reduced investigated geometry downstream from the ring plane: In this study it was found that the propagation of the thickness and the loading noise mainly concentrates near the ring plane and upstream. The SPL some distance away from the ring plane downstream is much smaller in comparison to that on the ring plane. This implies that the geometry some distance downstream from the ring plane might be removed to reduce the total number of mesh points and the computational cost in future work.

Glossary of Terms

Access Latency

The time between a request for an access to the RAM being issued, and the data becoming available

Anisotropic memory access pattern

Different memory access strides in the *x*, *y* and *z* direction. The memory access stride refers to the interval between two elements in an array accessed by two adjacent loops.

Array padding

Allocating dummy additional elements at the end of an array to match the size of a cache and to improve the cache use

Data race

Multiple write operations or write and read operations occurring simultaneously on the same RAM

Load balancing

Workloads are divided into many small portions which are distributed across working tasks. Load balancing aims to distribute the workloads as evenly as possible.

Thread

A process is an instance of a program implemented on a computer. It may contain multiple concurrent threads. A thread is a component of a process. Multiple threads share the instructions and some variables of a process.

Bibliography

References

1. N. E. Antoine, and I. M. Kroo. "Aircraft optimization for minimal environmental impact," *Journal of aircraft,* vol, **41**, No. 4, pp. 790-797, 2004.

2. P. Argüelles, M. Bischoff, P. Busquin, B. Droste, S. R. Evans, W. Kröll, J. Lagardere, A. Lina, J. Lumsden, and D. Ranque. "European Aeronautics: A vision for 2020," European Commission, 2001.

3. "Flightpath 2050 Europe's Vision for Aviation," *Report of the High Level Group on Aviation Research*, European Commission, 2011.

4. A. Filippone. "Aircraft noise prediction," *Progress in Aerospace Sciences,* vol, **68**, pp. 27-63, 2014.

5. X. Zhang. "Aircraft noise and its nearfield propagation computations," *Acta Mechanica Sinica,* vol, **28**, No. 4, pp. 960-977, 2012.

6. X.-d. Li, M. Jiang, J.-h. Gao, D.-k. Lin, L. Liu, and X.-y. Li. "Recent advances of computational aeroacoustics," *Applied Mathematics and Mechanics,* vol, **36**, No. 1, pp. 131-140, 2015.

7. S. W. Rienstra, and A. Hirschberg. *An introduction to acoustics*. Eindhoven University of Technology, 2003.

8. J. Yin, A. Stuermer, and M. Aversano. "Aerodynamic and Aeroacoustic Analysis of Installed Pusher-Propeller Aircraft Configurations," *Journal of Aircraft,* vol, **49**, No. 5, pp. 1423-1433, 2012.

9. J. Yin, and A. Stuermer. "Fast simulation of noise radiation and reduction from installed pusher propeller aircraft," *CEAS Aeronautical Journal,* vol, **4**, No. 4, pp. 443-458, 2013.

10. D. Juergen, A. A. Rinie, D. Jan, and E. Roland. "Installation Effects of a Propeller Mounted on a Wing with Coanda Flap. Part II: Numerical Investigation and Experimental Validation," *20th AIAA/CEAS Aeroacoustics Conference,* Atlanta Georgia, No. 2014-3189, 2014.

11. C. K. W. Tam, and J. C. Webb. "Dispersion-Relation-Preserving Finite Difference Schemes for Computational Acoustics," *Journal of Computational Physics,* vol, **107**, No. 2, pp. 262-281, 1993.

12. S. K. Lele. "Compact finite difference schemes with spectral-like resolution," *Journal of Computational Physics,* vol, **103**, No. 1, pp. 16-42, 1992.

13. R. Hixon. "Prefactored Small-Stencil Compact Schemes," *Journal of Computational Physics,* vol, **165**, No. 2, pp. 522-541, 2000.

14. G. Ashcroft, and X. Zhang. "Optimized prefactored compact schemes," *Journal of Computational Physics,* vol, **190**, No. 2, pp. 459-477, 2003.

15. W. Gropp, E. Lusk, N. Doss, and A. Skjellum. "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing,* vol, **22**, No. 6, pp. 789-828, 1996.

16. NVIDIA. "NVIDIA CUDA C Progamming Guide 5.5," NVIDIA Corporation, 2013.

17. E. Elsen, P. LeGresley, and E. Darve. "Large calculation of the flow over a hypersonic vehicle using a GPU," *Journal of Computational Physics,* vol, **227**, No. 24, pp. 10148-10161, 2008.

18. J. Cohen, and M. J. Molemaker. "A fast double precision CFD code using CUDA," *Parallel Computational Fluid Dynamics: Recent Advances and Future Directions,* Moffett Field, CA.pp. 414-429, 2009.

19. T. Brandvik, and G. Pullan. "Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science,* vol, **221**, No. 12, pp. 1745-1748, 2007.

20. B. Tobias, and P. Graham. "Acceleration of a 3D Euler Solver Using Commodity Graphics Hardware," *46th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, No. 2008-607, 2008.

21. T. Brandvik, and G. Pullan. "An Accelerated 3D Navier–Stokes Solver for Flows in Turbomachines," *Journal of Turbomachinery,* vol, **133**, No. 2, pp. 021025-021025, 2010.

22. J. Dana, T. Julien, and S. Inanc. "An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters," *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition,* Orlando, Florida, No. 2010-522, 2010.

23. C. J. Webb, and S. Bilbao. "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Prague.pp. 317-320, 2011.

24. C. J. Webb, and S. Bilbao. "Virtual Room Acoustics: A Comparison of Techniques for Computing 3D-FDTD Schemes Using CUDA," *Audio Engineering Society Convention 130,* London, UK, No. 8438, 2011.

25. J. J. López, D. Carnicero, N. Ferrando, and J. Escolano. "Parallelization of the finite-difference time-domain method for room acoustics modelling based on CUDA," *Mathematical and Computer Modelling,* vol, **57**, No. 7–8, pp. 1822-1831, 2013.

26. R. Ewert, and W. Schröder. "Acoustic perturbation equations based on flow decomposition via source filtering," *Journal of Computational Physics,* vol, **188**, No. 2, pp. 365-398, 2003.

27. M. J. Lighthill. "On Sound Generated Aerodynamically. I. General Theory," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* vol, **211**, No. 1107, pp. 564-587, 1952.

28. D. C. Mincu, and E. Manoha. "Numerical and experimental characterization of fan noise installation effects," *AerospaceLab*, pp. 1-11, 2014.

29. S. Malte, D. Jan, and C. Bastien. "Refraction and Scattering in High Mach Number Boundary Layers," *17th AIAA/CEAS Aeroacoustics Conference (32nd AIAA Aeroacoustics Conference),* Portland, Oregon, No. 2011-2847, 2011.

30. D. Juergen, E. Roland, D. Jan, S. Christian, and R. Marco. "The Effect of a Boundary Layer on Engine Noise propagating to the Fuselage at Flight Conditions," *19th AIAA/CEAS Aeroacoustics Conference,* Berlin, Germany, No. 2013-2006, 2013.

31. F. Farassat, and C. Jay. "Towards an Airframe Noise Prediction Methodology: Survey of Current Approaches," *44th AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, No. 2006-210, 2006.

32. R. J. Astley. "Numerical methods for noise propagation in moving flows, with application to turbofan engines," *Acoustical Science and Technology,* vol, **30**, No. 4, pp. 227-239, 2009.

33. M. Dieste, and G. Gabard. "Broadband Interaction Noise Simulations using Synthetic Turbulence," *16th International Conference in Sound and Vibration,* Krakow, Poland, No. 451, 2009.

34. X. Zhang, X. Chen, C. Morfey, and P. Nelson. "Computation of spinning modal radiation from an unflanged duct," *AIAA Journal,* vol, **42**, No. 9, pp. 1795-1801, 2004.

35. S. Richards, X. Chen, X. Huang, and X. Zhang. "Computation of fan noise radiation through an engine exhaust geometry with flow," *International Journal of Aeroacoustics,* vol, **6**, No. 3, pp. 223-241, 2007.

36. X. Huang, X. Chen, Z. Ma, and X. Zhang. "Efficient Computation of Spinning Modal Radiation Through an Engine Bypass Duct," *AIAA Journal,* vol, **46**, No. 6, pp. 1413-1423, 2008.

37. X. Chen, X. Huang, and X. Zhang. "Sound Radiation from a Bypass Duct with Bifurcations," *AIAA Journal,* vol, **47**, No. 2, pp. 429-436, 2009.

38. J. F. Williams, and D. L. Hawkings. "Sound generation by turbulence and surfaces in arbitrary motion," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences,* vol, **264**, No. 1151, pp. 321-342, 1969.

39. J. W. Kim, and D. J. Lee. "Optimized compact finite difference schemes with maximum resolution," *AIAA journal,* vol, **34**, No. 5, pp. 887-893, 1996.

40. F. Q. Hu, M. Y. Hussaini, and J. Manthey. "Low-dissipation and low-dispersion Runge–Kutta schemes for computational acoustics," *Journal of Computational Physics,* vol, **124**, No. 1, pp. 177-191, 1996.

41. C. A. Kennedy, and M. H. Carpenter. "Several new numerical methods for compressible shear-layer simulations," *Applied Numerical Mathematics,* vol, **14**, No. 4, pp. 397-433, 1994.

42. C. A. Kennedy, and M. H. Carpenter. "Comparison of several numerical methods for simulation of compressible shear layers," NASA-TP-3484, NASA Langley Research Center, Hampton, VA, 1997.

43. O. V. Vasilyev, T. S. Lund, and P. Moin. "A general class of commutative filters for LES in complex geometries," *Journal of Computational Physics,* vol, **146**, No. 1, pp. 82-104, 1998.

44. R. L. Manuel, S. Abhishek, R. B. Jonathan, D. E. Thomas, R. Joshua, E. W. Jerry, M. W. David, P. Francisco, J. Antony, and E. M. David. "Verification and Validation of HiFiLES: a High-Order LES unstructured solver on multi-GPU platforms," *32nd AIAA Applied Aerodynamics Conference,* Atlanta, GA, No. 2014-3168, 2014.

45. Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. "High-order

CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids,* vol, **72**, No. 8, pp. 811-845, 2013.

46. J. Antony. "Advances in Bringing High-Order Methods to Practical Applications in Computational Fluid Dynamics," *20th AIAA Computational Fluid Dynamics Conference,* Honolulu, Hawaii, No. 2011-3226, 2011.

47. B. Landmann. "A parallel discontinuous Galerkin code for the Navier-Stokes and Reynolds-averaged Navier-Stokes equations," **PhD** thesis, *Institution of Aerodynamics and Gasdynamics, University of Stuttgart,* 2008.

48. J. A. Ekaterinaris. "High-order accurate, low numerical diffusion methods for aerodynamics," *Progress in Aerospace Sciences,* vol, **41**, No. 3–4, pp. 192-300, 2005.

49. G. Thomas Le, and R. Gabriel. "Computational AeroAcoustics of Counter Rotating Open Rotor Model on rear full scale airplane in cruise condition," *18th AIAA/CEAS Aeroacoustics Conference (33rd AIAA Aeroacoustics Conference),* Colorado Springs, CO, No. 2012-2125, 2012.

50. S. Redonnet, G. Desquesnes, E. Manoha, and C. Parzani. "Numerical Study of Acoustic Installation Effects with a Computational Aeroacoustics Method," *AIAA Journal,* vol, **48**, No. 5, pp. 929-937, 2010.

51. Y. Liu, M. Vinokur, and Z. J. Wang. "Spectral difference method for unstructured grids I: Basic formulation," *Journal of Computational Physics,* vol, **216**, No. 2, pp. 780-801, 2006.

52. Z. J. Wang, Y. Liu, G. May, and A. Jameson. "Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations," *Journal of Scientific Computing,* vol, **32**, No. 1, pp. 45-71, 2007.

53. S. Richards, X. Zhang, X. Chen, and P. Nelson. "The evaluation of non-reflecting boundary conditions for duct acoustic computation," *Journal of Sound and Vibration,* vol, **270**, No. 3, pp. 539-557, 2004.

54. B. Chapman, G. Jost, and R. V. D. Pas. "Using OpenMP: portable shared memory parallel programming," The MIT Press, Cambridge, Massachusetts, 2008.

55. B. Barney. "Message Passing Interface (MPI)," https://computing.llnl.gov/tutorials/mpi/, Last accessed. 15/04/2013.

56. E. Lusk, and A. Chan. "Early Experiments with the OpenMP/MPI Hybrid Programming Model," *OpenMP in a New Era of Parallelism: 4th International Workshop, IWOMP 2008 West Lafayette*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 36-47.

57. R. Rabenseifner, G. Hager, and G. Jost. "Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes," *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, No. 1066-6192.pp. 427-436, 2009.

58. F. Wolf, and B. Mohr. "Automatic performance analysis of hybrid MPI/OpenMP applications," *Journal of Systems Architecture,* vol, **49**, No. 10–11, pp. 421-439, 2003.

59. "Top 500," https://www.top500.org/, Last accessed. 25/04, 2015.

60. U. o. Southampton. "Iridis," http://cmg.soton.ac.uk/iridis, Last accessed. 01.05, 2013.

61. J. Jeffers. "Intel® Xeon Phi™ Coprocessors," *Modern Accelerator Technologies for Geographic Information Science*. Springer US, Boston, MA, 2013, pp. 25-39.

62. M. Daga, A. M. Aji, and W. c. Feng. "On the Efficacy of a Fused CPU+GPU Processor (or APU) for Parallel Computing," *2011 Symposium on Application Accelerators in High-Performance Computing*, No. 2166-5133.pp. 141-149, 2011.

63. A. Munshi. "The OpenCL specification," *2009 IEEE Hot Chips 21 Symposium (HCS)*.pp. 1-314, 2009.

64. J. E. Stone, D. Gohara, and G. Shi. "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *Computing in Science & Engineering,* vol, **12**, No. 3, pp. 66-73, 2010.

65. A. Hart, R. Ansaloni, and A. Gray. "Porting and scaling OpenACC applications on massively-parallel, GPU-accelerated supercomputers," *The European Physical Journal Special Topics,* vol, **210**, No. 1, pp. 5-16, 2012.

66. T. Hoshino, N. Maruyama, S. Matsuoka, and R. Takaki. "CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application," *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*.pp. 136-143, 2013.

67. B. Behzad, J. M. Andrew, and J. R. Christopher. "Multilevel Parallelism for CFD Codes on Heterogeneous Platforms," *46th AIAA Fluid Dynamics Conference,* Washington, D.C., No. 2016-3329, 2016.

68. L. Lixiang, R. E. Jack, and L. Hong. "Performance Assessment of Multi-block LES Simulations using Directive-based GPU Computation in a Cluster Environment," *52nd Aerospace Sciences Meeting,* National Harbor, Maryland, No. 2014-1130, 2014.

69. "CUDA FORTRAN programming guide and reference," The Portland Group, 2012.

70. P. Micikevicius. "3D finite difference computation on GPUs using CUDA," *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*. ACM, 2009, pp. 79-84.

71. J. A. Hay, C. Richter, L. Panek, and F. Thiele. "Acceleration of CAA methodology for liner applications using graphics processors," *Proceedings of the Seventeenth International Congress on Sound and Vibration, Cairo, Egypt*. 2010.

72. Q. H. Fang. "An efficient solution of time domain boundary integral equations for acoustic scattering and its acceleration by Graphics Processing Units," *19th AIAA/CEAS Aeroacoustics Conference,* Berlin, Germany, No. 2013-2018, 2013.

73. C. Andrew, K. Kailas, L. Junhui, R. Ravi, S. Douglas, and D. Johann. "A Hybrid Grid Compressible Flow Solver for Large-Scale Supersonic Jet Noise Simulations on Multi-GPU Clusters," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition,* Nashville, Tennessee, No. 2012-0564, 2012.

74. X. Yidong, L. Lixiang, L. Hong, L. Jialin, R. E. Jack, and M. Frank. "On the Multi-GPU Computing of a Reconstructed Discontinuous Galerkin Method for Compressible Flows on 3D Hybrid Grids," *7th AIAA Theoretical Fluid Mechanics Conference,* Atlanta, GA, No. 2014-0381, 2014.

75. E. W. Jerry, R. Joshua, and J. Antony. "Multi-GPU, Implicit Time Stepping for High-order Methods on Unstructured Grids," *46th AIAA Fluid Dynamics Conference,* Washington, D.C., No. 2016-3965, 2016.

76. W. Dawes, D. Caleb, and K. William. "Acceleration of an Unstructured Hybrid Mesh RANS Solver by Porting to GPU Architectures," *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition,* Orlando, Florida, No. 2011-944, 2011.

77. L. Jialin, X. Yidong, L. Lixiang, L. Hong, R. E. Jack, and M. Frank. "OpenACC-based GPU Acceleration of a *p*-multigrid Discontinuous Galerkin Method for Compressible Flows on 3D Unstructured Grids," *53rd AIAA Aerospace Sciences Meeting,* Kissimmee, Florida, No. 2015-0822, 2015.

78. L. Lixiang, R. E. Jack, L. Hong, and M. Frank. "GPU Port of A Parallel Incompressible Navier-Stokes Solver based on OpenACC and MVAPICH2," *7th AIAA Theoretical Fluid Mechanics Conference,* Atlanta, GA, No. 2014-3083, 2014.

79. L. Jialin, X. Yidong, L. Lixiang, L. Hong, R. E. Jack, and M. Frank. "OpenACC directive-based GPU acceleration of an implicit reconstructed discontinuous Galerkin method for compressible flows on 3D unstructured grids," *54th AIAA Aerospace Sciences Meeting,* San Diego, California, No. 2016-1815, 2016.

80. C. Dominic, S. Jayanaryanan, and M. Dimitri. "CU++ET: An Object Oriented Tool for Accelerating Computational Fluid Dynamics codes using Graphical Processing Units," *20th AIAA Computational Fluid Dynamics Conference,* Honolulu, Hawaii, No. 2011-3222, 2011.

81. V. Peter, D. W. Freddie, M. F. Antony, N. George, C. V. Brian, S. P. Jin, and S. I. Arvind. "PyFR: Next-Generation High-Order Computational Fluid Dynamics on Many-Core Hardware (Invited)," *22nd AIAA Computational Fluid Dynamics Conference,* Dallas, TX, No. 2015-3050, 2015.

82. Nvidia. "CUBLAS," http://docs.nvidia.com/cuda/cublas/index.html, Last accessed. 20/02, 2012.

83. "Portable, Extensible Toolkit for Scientific Computation," https://www.mcs.anl.gov/petsc/, Last accessed. 01/06, 2016.

84. "LAPACK — Linear Algebra PACKage," http://www.netlib.org/lapack/#_users_guide, Last accessed. 15/02, 2012.

85. Y. Zhang, J. Cohen, and J. D. Owens. "Fast tridiagonal solvers on the GPU," *ACM Sigplan Notices,* vol, **45**, No. 5, pp. 127-136, 2010.

86. H. S. Stone. "An efficient parallel algorithm for the solution of a tridiagonal linear system of equations," *Journal of the ACM (JACM),* vol, **20**, No. 1, pp. 27-38, 1973.

87. D. Egloff. "High performance finite difference PDE solvers on GPUs," *QuantAlea GmbH,,* Zurich, Switzerland, 2010.

88. J. Krger, and d. Westermann. "Linear algebra operators for GPU implementation of numerical algorithms," *ACM SIGGRAPH 2005 Courses,* Los Angeles, California.pp. 234, 2005.

89. J. Bolz, I. Farmer, E. Grinspun, and P. Schrder. "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," *ACM SIGGRAPH 2005 Courses,* Los Angeles, California.pp. 171, 2005.

90. B. Tutkun, and F. O. Edis. "A GPU application for high-order compact finite difference scheme," *Computers & Fluids,* vol, **55**, pp. 29-35, 2012.

91. P. Micikevicius. "3D finite difference computation on GPUs using CUDA," *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units,* Washington, D.C., USA.pp. 79-84, 2009.

92. S. Miao, X. Zhang, O. G. Parchment, and X. Chen. "A fast GPU based bidiagonal solver for computational aeroacoustics," *Computer Methods in Applied Mechanics and Engineering,* vol, **286**, pp. 22-39, 2015.

93. S. Miao, X. Zhang, O. G. Parchment, and X. Chen. "A Fast GPU Based Bidiagonal Solver for Computational Aeroacoustics," *20th AIAA/CEAS Aeroacoustics Conference,* Atlanta, GA, No. 2014-2317, 2014.

94. J. L. Gustafson. "Reevaluating Amdahl's law," *Commun. ACM,* vol, **31**, No. 5, pp. 532-533, 1988.

95. H. H. Hubbard. "Aeroacoustics of flight vehicles: Theory and practice. volume 1. noise sources," NASA-RP-1258-VOL-1, NASA Langley Research Center; Hampton, VA, 1991.

96. F. Farassat. "Linear Acoustic Formulas for Calculation of Rotating Blade Noise," *AIAA Journal,* vol, **19**, No. 9, pp. 1122-1130, 1981.

97. F. Farassat, D. Mark, T. Ana, and N. Douglas. "Open Rotor Noise Prediction Methods at NASA Langley: A Technology Review," *15th AIAA/CEAS Aeroacoustics Conference (30th AIAA Aeroacoustics Conference),* Miami, Florida, No. 2009-3133, 2009.

98. F. B. Metzger. "A review of propeller noise prediction methodology: 1919-1994," NASA-CR-198156, Metzger Technology Services; Simsbury, CT, United States, 1995.

99. D. B. Hanson, and D. J. Parzych. "Theory for noise of propellers in angular inflow with parametric studies and experimental verification," NASA-CR-4499, NASA. Lewis Research Center, Washington, United States, 1993.

100. P. di Francescantonio. "A NEW BOUNDARY INTEGRAL FORMULATION FOR THE PREDICTION OF SOUND RADIATION," *Journal of Sound and Vibration,* vol, **202**, No. 4, pp. 491-509, 1997.

101. F. Farassat. "Derivation of Formulations 1 and 1A of Farassat," NASA/TM-2007-214853, NASA Langley Research Center; Hampton, VA, United States, 2007.

102. A. A. Rinie, P.-P. Michael, B. Heino, D. Jan, and A. Daniela. "Installation Effects of a Propeller Mounted on a High-Lift Wing with a Coanda Flap. Part I: Aeroacoustic Experiments," *20th AIAA/CEAS Aeroacoustics Conference,* Atlanta, GA, No. 2014-3191, 2014.

103. J. E. Marte, and D. W. Kurtz. "A review of aerodynamic noise from propellers, rotors, and lift fans," Jet Propulsion Laboratory, California Institute of Technology, 1970.

104. M. Niţă. "Aircraft design studies based on the ATR 72," **PhD** thesis*, Hamburg University of Applied Sciences,* 2008.

105. K. Seeckt. "Application of PreSTo: Aircraft preliminary sizing and data export to CEASIOM," Technical report, Royal Institute of Technology, 2011.

106. K. Seeckt, and D. Scholz. "Application of the aircraft preliminary sizing tool PreSTo to kerosene and liquid hydrogen fueled regional freighter aircraft," *Proceedings: Deutscher Luft- und Raumfahrtkongress 2010*, 2010.

107. S. Miao, Y. Hou, and X. Zhang. "Evaluation AND COMPARISON OF LINEARIZED EULER EQUATIONS AND EQUIVALENT SOURCE METHOD," *The 22nd International Congress on Sound and Vibration,* Florence, Italy, No. 169, 2015.

108. H. Yu, Z. Xin, and A. David. "A Complex Equivalent Source Method for Scattering Effect of Aircraft Noise," *20th AIAA/CEAS Aeroacoustics Conference,* Atlanta, GA, No. 2014-3302, 2014.

109. Y. J. Gounot, and R. E. Musafir. "Simulation of scattered fields: Some guidelines for the equivalent source method," *Journal of Sound and Vibration,* vol, **330**, No. 15, pp. 3698-3709, 2011.

110. M. R. Bai. "Application of BEM (boundary element method)‐based acoustic holography to radiation analysis of sound sources with arbitrarily shaped geometries," *The Journal of the Acoustical Society of America,* vol, **92**, No. 1, pp. 533-549, 1992.

111. M. Eric, J. Xavier, and R. Francois. "Numerical Simulation of Aircraft Engine Installation Acoustic Effects," *11th AIAA/CEAS Aeroacoustics Conference,* Monterey, California, No. 2005-2920, 2005.

112. R. Stéphane, P. Céline, M. Eric, and L. Delphine. "Numerical Study of 3D Acoustic Installation Effects Through a Hybrid Euler/BEM Method," *13th AIAA/CEAS Aeroacoustics Conference (28th AIAA Aeroacoustics Conference),* Rome, Italy, No. 2007-3500, 2007.

113. P. L. Spence. "Effects of fuselage boundary layer on noise propagation from advanced propellers," *Journal of Aircraft,* vol, **29**, No. 6, pp. 1005-1011, 1992.

114. D. Hanson. "Shielding of Prof-Fan cabin noise by the fuselage boundary layer," *Journal of Sound and Vibration,* vol, **92**, No. 4, pp. 591-598, 1984.

115. M. G, and J. R. J. Rawls. "Effects of boundary layer refraction and fuselage scattering on fuselage surface noise from advanced turboprop propellers," *22nd Aerospace Sciences Meeting,* Reno,NV, No. 1984-249, 1984.

116. H. Y. Lu. "Fuselage boundary-layer effects on sound propagation and scattering," *AIAA Journal,* vol, **28**, No. 7, pp. 1180-1186, 1990.

117. D. Hanson, and B. Magliozzi. "Propagation of propeller tone noise through a fuselage boundary layer," *Journal of Aircraft,* vol, **22**, No. 1, pp. 63-70, 1985.

118. S. M, T. A. M. C, B. R, and G. J. "An experimental and theoretical investigation of the propagation of sound waves through a turbulent boundary layer," *10th Aeroacoustics Conference,* Seattle,WA, No. 1986-1968, 1986.

119. G. McAninch. "A note on propagation through a realistic boundary layer," *Journal of Sound and Vibration,* vol, **88**, No. 2, pp. 271-274, 1983.

120. V. Miguel, and G. Datta. "Computation of aeroacoustic fields on general geometries using compact differencing and filtering schemes," *30th Fluid Dynamics Conference,* Norfolk,VA, No. 1999-3706, 1999.

121. M. Visbal, and D. Rizzetta. "Large-eddy simulation on general geometries using compact differencing and filtering schemes," *40th AIAA Aerospace Sciences Meeting & Exhibit,* Reno,NV, No. 2002-288, 2002.

122. R. Hixon. "Prefactored compact filters for computational aeroacoustics," *37th Aerospace Sciences Meeting and Exhibit,* Reno,NV, No. 1999-0358, 1999.

123. A. Najafi-Yazdi, and L. Mongeau. "A low-dispersion and low-dissipation implicit Runge–Kutta scheme," *Journal of Computational Physics,* vol, **233**, pp. 315-323, 2013.

124. M. Parsani, G. Ghorbaniasl, C. Lacor, and E. Turkel. "An implicit high-order spectral difference approach for large eddy simulation," *Journal of Computational Physics,* vol, **229**, No. 14, pp. 5373-5393, 2010.

125. R. Bulirsch, and J. Stoer. "Numerical treatment of ordinary differential equations by extrapolation methods," *Numerische Mathematik,* vol, **8**, No. 1, pp. 1-13, 1966.

126. J. A, S. Wolfgang, and T. Eli. "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes," *14th Fluid and Plasma Dynamics Conference,* Palo Alto,CA, No. 1981-1259, 1981.

127. X. Zhang, X. Chen, and C. Morfey. "Acoustic radiation from a semi-infinite duct with a subsonic jet," *International Journal of Aeroacoustics,* vol, **4**, No. 1, pp. 169-184, 2005.

128. J. W. Kim, and D. J. Lee. "Generalized Characteristic Boundary Conditions for Computational Aeroacoustics," *AIAA Journal,* vol, **38**, No. 11, pp. 2040-2049, 2000.

129. J. W. Kim, and D. Joo. "Generalized Characteristic Boundary Conditions for Computational Aeroacoustics, Part 2," *AIAA Journal,* vol, **42**, No. 1, pp. 47-55, 2004.

130. M. L. Caraeni, and L. Fuchs. "Investigation of Nonreflective Boundary Conditions for Computational Aeroacoustics," *AIAA Journal,* vol, **44**, No. 9, pp. 1932-1940, 2006.

131. T. I. M. Colonius, S. K. Lele, and P. Moin. "Boundary conditions for direct computation of aerodynamic sound generation," *AIAA Journal,* vol, **31**, No. 9, pp. 1574-1582, 1993.

132. J. B. Freund. "Proposed Inflow/Outflow Boundary Condition for Direct Computation of Aerodynamic Sound," *AIAA Journal,* vol, **35**, No. 4, pp. 740-742, 1997.

133. R. Hixon, and S. i. R. R. S-Hmankbadi. "Evaluation of boundary conditions for computational aeroacoustics," *33rd Aerospace Sciences Meeting and Exhibit,* Reno,NV, No. 1995-160, 1995.

134. K. Benjamin, and E. Gunilla. "Evaluating stretched grids and introducing black hole layers as alternative non-reflecting buffer zone," *19th AIAA/CEAS Aeroacoustics Conference,* Berlin, Germany, No. 2013-2220, 2013.

135. E. Nathan, and V. Miguel. "A General Buffer Zone-type Non-Reflecting Boundary Condition for Computational Aeroacoustics," *9th AIAA/CEAS Aeroacoustics Conference and Exhibit,* Hilton Head, South Carolina, No. 2003-3300, 2003.

136. T. Kenji, Z. Xin, and N. Philip. "Linearized Euler Simulations of Leading-Edge Slat Flow," *41st Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, No. 2003-364, 2003.

137. F. Q. Hu. "Absorbing Boundary Conditions," *International Journal of Computational Fluid Dynamics,* vol, **18**, No. 6, pp. 513-522, 2004.

138. C. K. W. Tam, and Z. Dong. "Wall boundary conditions for high-order finite-difference schemes in computational aeroacoustics," *Theoretical and Computational Fluid Dynamics,* vol, **6**, No. 5, pp. 303-322, 1994.

139. R. Hixon. "Curvilinear wall boundary conditions for computational aeroacoustics," *35th Joint Propulsion Conference and Exhibit,* Los Angeles,CA, No. 1999-2395, 1999.

140. R. Hixon, S. H. Shih, and R. R. Mankbadi. "Evaluation of Boundary Conditions for the Gust-Cascade Problem," *Journal of Propulsion and Power,* vol, **16**, No. 1, pp. 72-78, 2000.

141. R. Hixon. "Radiation and Wall Boundary Conditions for Computational Aeroacoustics: A Review," *International Journal of Computational Fluid Dynamics,* vol, **18**, No. 6, pp. 523-531, 2004.

142. P. Diane, A. Steven, M. Douglas, S. Matthew, and E. Francis. "The CGNS system," *29th AIAA, Fluid Dynamics Conference,* Albuquerque,NM, No. 1998-3007, 1998.

143. H. S. Kim, S. Wu, L. w. Chang, and W. m. W. Hwu. "A Scalable Tridiagonal Solver for GPUs," *2011 International Conference on Parallel Processing,* Taipei City.pp. 444-453, 2011.

144. H. S. Stone. "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations," *J. ACM,* vol, **20**, No. 1, pp. 27-38, 1973.

145. X. H. Sun, H. Zhang, and L. M. Ni. "Efficient tridiagonal solvers on multicomputers," *IEEE Transactions on Computers,* vol, **41**, No. 3, pp. 286-296, 1992.

146. J. C. Hardin, J. R. Ristorcelli, and C. K. W. Tam. "ICASE/LaRC workshop on benchmark problems in computational aeroacoustics," NASA-CP-3300, NASA Langley Research Center; Hampton, VA, 1995.

147. G. H. Golub, L. Shui-Hong, T. L. Franklin, and R. J. Plemmons. "Cyclic reduction—history and applications," *Scientific Computing, Proceedings of the Workshop, 10 - 12 March 1997, Hong Kong*. Springer Singapore, 1997, pp. 73-85.

148. R. A. Sweet. "A Parallel and Vector Variant of the Cyclic Reduction Algorithm," *SIAM Journal on Scientific and Statistical Computing,* vol, **9**, No. 4, pp. 761-765, 1988.

149. P. Amodio, and N. Mastronardi. "A parallel version of the cyclic reduction algorithm on a hypercube," *Parallel Computing,* vol, **19**, No. 11, pp. 1273-1281, 1993.

150. S. Christopher, D. Earl, Z. Yao, C. David, D. Roger, and O. John. "GPGPU parallel algorithms for structured-grid CFD codes," *20th AIAA Computational Fluid Dynamics Conference,* Honolulu, Hawaii, No. 2011-3221, 2011.

151. M. Harris. "Optimizing Parallel Reduction in CUDA," http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/reduction/doc/reduction.pdf, Last accessed. 26/01, 2007.

152. B. Ravi, A. Don, and S. Tom. "PCI express system architecture," Addison-Wesley Professional, 2003.

153. Nvidia. "Tesla M2050/M2070 GPU Computing Module: Supercomputing at 1/10th the cost," http://www.nvidia.co.uk/docs/IO/43395/NV_DS_Tesla_M2050_M2070_Apr10_LowRes.pdf, Last accessed. 23/01, 2010.

154. G. Shainer, A. Ayoub, P. Lui, T. Liu, M. Kagan, C. R. Trott, G. Scantlen, and P. S. Crozier. "The development of Mellanox/NVIDIA GPUDirect over InfiniBand—a new model for GPU to GPU communications," *Computer Science - Research and Development,* vol, **26**, No. 3, pp. 267-273, 2011.

155. S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, and D. K. Panda. "Efficient Inter-node MPI Communication Using GPUDirect RDMA for InfiniBand Clusters with NVIDIA GPUs," *2013 42nd International Conference on Parallel Processing,* Lyon, France.pp. 80-89, 1-4 Oct. 2013, 2013.

156. H. Wang, S. Potluri, D. Bureddy, C. Rosales, and D. K. Panda. "GPU-Aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation," *IEEE Transactions on Parallel and Distributed Systems,* vol, **25**, No. 10, pp. 2595-2605, 2014.

157. P. Joseph, and C. L. Morfey. "Multimode radiation from an unflanged, semi-infinite circular duct," *The Journal of the Acoustical Society of America,* vol, **105**, No. 5, pp. 2590-2600, 1999.

158. G. F. Homicz, and J. A. Lordi. "A note on the radiative directivity patterns of duct acoustic modes," *Journal of Sound and Vibration,* vol, **41**, No. 3, pp. 283-290, 1975.

159. R. M. Munt. "The interaction of sound with a subsonic jet issuing from a semi-infinite cylindrical pipe," *Journal of Fluid Mechanics,* vol, **83**, No. 04, pp. 609-640, 1977.

160. S. K. Richards, X. X. Chen, X. Huang, and X. Zhang. "Computation of Fan Noise Radiation through an Engine Exhaust Geometry with Flow," *International Journal of Aeroacoustics,* vol, **6**, No. 3, pp. 223-241, 2007.

161. D. Casalino. "An advanced time approach for acoustic analogy predictions," *Journal of Sound and Vibration,* vol, **261**, No. 4, pp. 583-612, 2003.

162. D. B. Hanson. "Direct frequency domain calculation of open rotor noise," *AIAA Journal,* vol, **30**, No. 9, pp. 2334-2337, 1992.

163. Y. J. R. Gounot, and R. E. Musafir. "Simulation of scattered fields: Some guidelines for the equivalent source method," *Journal of Sound and Vibration,* vol, **330**, No. 15, pp. 3698-3709, 2011.

164. D. A. Russell, J. P. Titlow, and Y.-J. Bemmen. "Acoustic monopoles, dipoles, and quadrupoles: An experiment revisited," *American Journal of Physics,* vol, **67**, No. 8, pp. 660-664, 1999.

165. M. J. Crocker. "Handbook of acoustics," USA, Canada: John Wiley & Sons, Inc, 1998.

166. S. Martínez-Aranda, A. L. García-González, L. Parras, J. F. Velázquez-Navarro, and C. d. Pino. "Comparison of the Aerodynamic Characteristics of the NACA0012 Airfoil at Low-to-Moderate Reynolds Numbers for any Aspect Ratio," *International Journal of Aerospace Sciences,* vol, **4**, No. 1, pp. 1-8, 2016.

167. I. V. Belyaev. "The effect of an aircraft's boundary layer on propeller noise," *Acoustical Physics,* vol, **58**, No. 4, pp. 387-395, 2012.

168. J. Gao. "A block interface flux reconstruction method for numerical simulation with high order finite difference scheme," *Journal of Computational Physics,* vol, **241**, pp. 1-17, 2013.